TSPS No. 1:

Stored Program Control No. 1A

By G. R. DURNEY, H. W. KETTLER, E. M. PRELL,
G. RIDDELL and W. B. ROHN

(Manuscript received August 17, 1970)

The Stored Program Control No. 1A processor complex was conceived as a program sequenced control for the Traffic Service Position System No. 1 and for the Electronic Translator Systems. It was designed to be a highly reliable complex using a conservative discrete component hardware design with excellent fault detecting capability and with exceptionally good automatic recovery of call processing when faced with hardware faults.

The design aims, order structure, hardware features, fault detection, diagnostic, and recovery aspects of the system are described. Stress is placed on those features which are felt to be improvements over previous program controlled systems.

I. INTRODUCTION

New telephone services are regularly being conceived which place large demands upon existing switching facilities—demands which are increasingly difficult to satisfy with electromechanical techniques. The use of large high-speed memory and stored-program logic permits the modernization of existing switching functions[1] and implemenation of new services more effectively and at lower cost than by electromechanical means. The stored program control (SPC) No. 1A has been developed as a general purpose stored program electronic processing system to provide a flexible control for implementing new or modernized telephone services. It is a system which is independent of application but with generalized interfaces to which hardware and software may be readily applied in the development of specific application systems.

Bell System electronic switching machines must work reliably in an environment in which substantial noise and temperature variations may be encountered. System processing capacity must economically meet traffic handling needs over the projected life of the system. However, this requirement is generally met more readily with state-of-theart technology than the reliability objectives. Thus, the SPC No. 1A represents a conservative design in device technology and processing speed, but an advanced and sophisticated organization for automatic trouble recovery. The emphasis in this article is on the hardware and software organization for maintaining system operation with particular attention given to the maintenance features which depart from those used in other Bell System stored-program processors.

The SPC was developed concurrently with the traffic service position system (TSPS) No. 1, which is designed to improve operator assistance facilities.[2] The SPC is also used in the electronic translator system to replace the present electromechanical call routing translators in the 4A toll crossbar system.

II. A COMMON SYSTEM

2.1 *Hardware Organization*

The basic concept of the SPC is that of an electronic data processor operating with a stored program to control all functions of its associated application system on a time-shared basis. The general organization of the SPC complex is illustrated in Fig. 1. The complex consists of the arithmetic-logic unit known as the processor, the memory system using "piggyback twistor" stores,[3] several peripheral units required for maintenance of the processor-memory system, and a master control center complex for man-machine interaction.

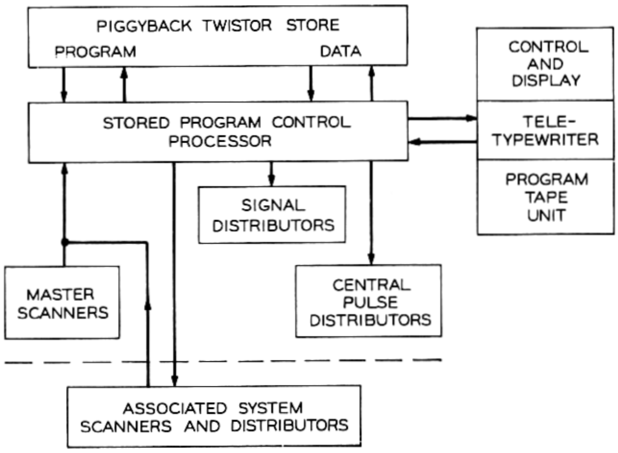


Fig. 1—Basic SPC complex.

To permit the widest application for the SPC, careful consideration was given to the structure and definition of the instruction set. The final choice consisted of 19 basic processing operations and 14 maintenance operations. These represent a compact but powerful order set that is well-balanced for either logical data processing or for control of peripheral hardware. This was particularly important for the SPC which in its inception was not aimed specifically at either one of these fields of application.

The first two application systems using the SPC, the electronic translator system and TSPS, are examples of these two quite distinct processing orientations: logical data processing and peripheral hardware control.

For basic data manipulation, memory access is organized for both word and character. In addition to access to full or half words, individual memory access instructions can, through an option field, specify the reading or writing of any byte. (A byte is an item of i bits positioned j bits from the least significant bit of any half word.) Seven processor index registers are fully flexible, that is, any may be specified for the functions of accumulator, return address, and so on. Transfer tests may be made on the total data or any specified bit of any register. The basic hardware oriented orders are scan and distribute orders. Special purpose orders provide combined operations to permit real-time efficiency for highly repetitive input-output functions.

Data transfer to and from memory always consists of a full 40-bit word plus seven bits of error detection and correction code. The stores are nondestructive readout memories which are used to store both semipermanent and temporary data. Semipermanent data locations are protected from inadvertent writing by lockout circuitry. Protection is defined in increments of 1/16th of a store through wiring straps which may be easily changed in the field. The lockout function may be overridden by program control to permit updating or loading the semipermanent data.

Since the stores are electrically alterable, they provide a single-memory-device system for the generic programs, parameter data, office translation data, and the temporary data scratch pad.

Maintenance of the central processor requires the ability to distribute control signals (that is, to set flip-flops or operate relays). This dictates the need for a central pulse distributor and a signal distributor.[4] To maintain units of the SPC, the system must also perform scan operations to read the states of internal points which requires the use of a master scanner.[4] Points in the matrices of these units which are not required for the SPC are available for use in the application system. Communication to these units is via private SPC peripheral and central pulse distributor address bus systems.

As a common system, the SPC design does not attempt to anticipate the peripheral addressing requirements of the application systems. Thus, provision is made for a standard binary output to an application system peripheral unit address bus. It is assumed that each application system requiring peripheral unit communication will provide an external translator for any desired translation function.[5] Application system scanner type units share the scanner answer bus system with the SPC master scanner.

The master control center consists of a control and display panel, a teletypewriter and a program tape unit. The control and display provides visual and audible indications of major and minor alarm conditions, as well as lamp displays of the status of the various equipment and buses that make up the SPC complex. It also provides keys for data input to the system and for control of the system configuration of on-line equipment units when manually aided recovery is required. These lamps and keys represent additional functions of the SPC that are implemented through the central pulse distributor, signal distributor, and master scanner.

The teletypewriter is the device by which the maintenance craftsman receives detailed information concerning trouble detection and isolation and through which he instructs the system to perform specific tests.

The program tape unit serves as a secondary memory system that is used initially to load the piggyback twistor memory and subsequently to make large changes in the generic program or office translation data. The capacity of the tape memory is such that many application system functions requiring bulk storage with slow access can be accommodated. For example, it can be used to collect dumps of office data for off-line verification and to retain the original office data during changes. It thereby serves as a backup should restoration be required. It can also be used to contain a library of infrequently used application routines, such as growth testing routines.

Flexibility for growth is quite important. In the SPC complex, the impact of growth centers primarily on the stores and central pulse distributors. As previously mentioned, the signal distributor and master scanner are on a private SPC address bus system and contain adequate matrix capacity for all SPC functions up to a maximum size. Assignments in the signal distributor and master scanner have been made to allow for maximum growth in the SPC area before allowing assignment points to be used by the application system. Growth for stores is considerably simplified by providing full bus connectorization.[6]

2.2 *Software Organization*

The SPC generic program provides the necessary maintenance functions for recovery from troubles and for fault isolation in the various equipment that comprises the SPC hardware complex. (Generic program refers to the set of instructions common to all SPC systems.)

Since maintenance of the SPC hardware involves a man-machine interface, the programs which relate to the control and display, teletypewriter, and program tape unit are also part of the generic program. The maintenance test programs and input-output programs for the SPC equipment function as an integral part of the surrounding software-environment and make very specific assumptions regarding this environment. This environment consists of job administration and certain common service programs. These provide not only a cohesive SPC maintenance system but also the framework for executing a broad range of application programs.

Figure 2 is a general block diagram of the SPC generic program and its interfaces with the application system program. The blocks are arranged from top to bottom to show descending priority levels of program execution. Certain blocks are labeled SPC APPLICATION. This indicates that for this class of program functions generalized interfaces have been established that readily allow integration of application system programs with SPC programs.

Of highest priority in the automatic handling of troubles are the fault recognition programs which are entered by interrupts generated when trouble detection circuits recognize a system malfunction. These programs are required to distinguish non-repeating from repeating circuit malfunctions and, in the latter instances, to remove the faulty unit from service, establishing a working system configuration so that the system's normal tasks may be resumed. In priority ranking, they are concerned with processor, store, and peripheral unit malfunctions. The peripheral unit fault recognition programs at this priority level are concerned with symptoms of hardware trouble which appear immediately after signals are sent to the peripheral system. Failure to receive an enable-verify signal or an all-seems-well signal from the peripheral unit indicates such trouble.

Hardware emergency action is another high priority program that is required to facilitate recovery from circuit malfunctions. It functions

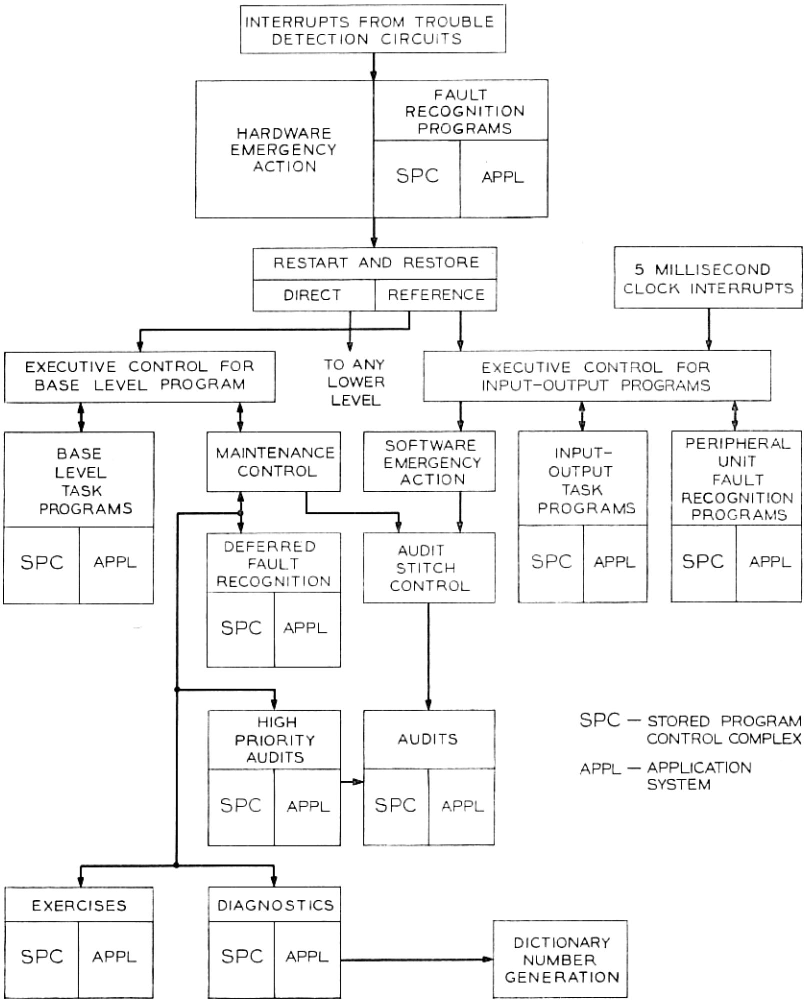

INTERRUPTS FROM TROUBLE DETECTION CIRCUITS
HARDWARE EMERGENCY ACTION
FAULT RECOGNITION PROGRAMS
SPC
APPL
RESTART AND RESTORE
DIRECT
REFERENCE
5 MILLISECOND CLOCK INTERRUPTS
EXECUTIVE CONTROL FOR BASE LEVEL PROGRAM
TO ANY LOWER LEVEL
EXECUTIVE CONTROL FOR INPUT-OUTPUT PROGRAMS
BASE LEVEL TASK PROGRAMS
MAINTENANCE CONTROL
SOFTWARE EMERGENCY ACTION
INPUT-OUTPUT TASK PROGRAMS
PERIPHERAL UNIT FAULT RECOGNITION PROGRAMS
DEFERRED FAULT RECOGNITION
AUDIT STITCH CONTROL
HIGH PRIORITY AUDITS
AUDITS
SPC — STORED PROGRAM CONTROL COMPLEX
APPL — APPLICATION SYSTEM
EXERCISES
DIAGNOSTICS
DICTIONARY NUMBER GENERATION

Fig. 2—SPC generic program.

in conjunction with a wired logic hardware sequencer to establish and test new combinations of active processor and active base store until a combination is found which is trouble free. (The base store is the store unit containing the emergency action recovery programs.) It is essential to cope with troubles that destroy sanity of the active control processor system first.

All maintenance interrupt programs return to normal call processing via the restart and restore program. In most cases, the return is made to pick up processing at the point at which the interruption occurred. However, under some conditions this is not safe or feasible; the return is then made to an appropriate restart or reference point in the job administration stream where carryover of index register data is not required. Several of these reference points are defined in the two executive control programs.

An executive control program for input-output work is entered periodically from interrupts generated by a 5 ms clock.[7] This represents the next lower major priority class of work. Both SPC and application system input-output tasks are scheduled by the executive control. The SPC tasks include teletypewriter and program tape unit input-output, execution of signal distributor orders to operate lamps and alarms, and scanning for alarm conditions.

Another program activity scheduled at the input-output priority level is an extension of peripheral unit fault recognition. These programs look for peripheral controllers that have failed to reset to an idle state in the normal time interval. The SPC program of this class is required to handle signal distributor controller troubles.

A software emergency action is also scheduled by the executive control for input-output programs. It makes several tests for the sanity of system processing to detect degradation that could have been caused by data mutilation in unprotected memory. The SPC system and any appropriate application programs normally detect and correct such data errors by frequently running a series of memory audit programs. However, in some cases, mutilated data will produce an avalanche effect that requires immediate and drastic action. The emergency action program is designed to detect these cases and immediately execute a series of memory audit and initialization routines to restore normal operation.

The lowest priority class is that consisting of all the programs executed at the base level of processing, that is, the noninterrupt level of processing. These are scheduled by an executive control for base level programs which has flexible arrangements for assigning priority and scheduling task programs. These administrative functions are developed through the use of parameter tables which provide a simple interface for application programs.

In general, base level task programs process data collected by input programs and buffer appropriate data to be subsequently distributed by output programs. Return address linking is maintained in the output buffer tables so that the executive control programs can arrange the stages of processing in sequence for each call while providing time-shared processing of many calls.

At the lowest priority level in the base level of processing, the executive control enters the maintenance control program. This program arranges in sequence all maintenance task programs whose work may be performed on low priority relative to other system tasks. These deferrable maintenance tasks are executed by maintenance control according to priority.

The highest priority group consists of the deferred fault recognition programs. These programs are intended to bring as many units as possible into operation, since the interrupt level fault recognition program (because of constraints on interrupt time) may have to put together a minimum working configuration. The next level consists of high priority memory auditing routines designed to check the integrity and consistency of the data used in the various bookkeeping operations of the program system. The routines are called in at this priority level by programs encountering processing anomalies or hardware failures which might lead to data mutilation.

The diagnostic programs are next in the priority structure. They perform exhaustive tests on the units removed from service by the fault recognition programs and save the pass-fail data for the tests. These data are processed by the dictionary number generation program to produce a number printed on the maintenance teletypewriter. The number is used to enter an appropriate trouble location manual which will identify a circuit pack or packs suspected of causing the trouble. The lowest priority of maintenance programs are the exercises, which include routines that respond to manual requests, routines that periodically test circuit functions which are not in continuous use, and memory audits which are run when all high priority work has been completed. The tables used by the maintenance control program make generous allowance for adding application routines in each of the various priority categories.

Another important function of the maintenance control program is known as audit stitch control. This mode of system operation is requested by the software emergency action program when it detects symptoms of software insanity. Under the audit stitch control, a series of memory audit and software-hardware initialization routines are executed (that is, "stitched") to restore sanity. All call processing activity is suspended during this operation.

Figure 3 depicts the program control plan specifying the functional assignments for the maintenance interrupts (A through G plus K),

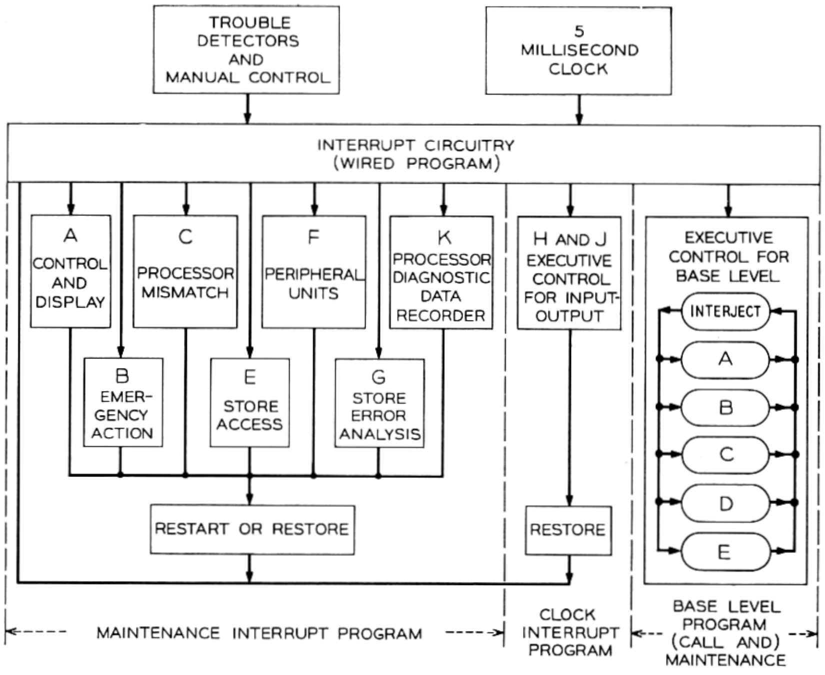

TROUBLE DETECTORS AND MANUAL CONTROL
5 MILLISECOND CLOCK
INTERRUPT CIRCUITRY (WIRED PROGRAM)
A CONTROL AND DISPLAY
C PROCESSOR MISMATCH
F PERIPHERAL UNITS
K PROCESSOR DIAGNOSTIC DATA RECORDER
H AND J EXECUTIVE CONTROL FOR INPUT-OUTPUT
EXECUTIVE CONTROL FOR BASE LEVEL
INTERJECT
A
B
C
D
E
B EMERGENCY ACTION
E STORE ACCESS
G STORE ERROR ANALYSIS
RESTART OR RESTORE
RESTORE
MAINTENANCE INTERRUPT PROGRAM
CLOCK INTERRUPT PROGRAM
BASE LEVEL PROGRAM (CALL AND) MAINTENANCE

Fig. 3—Program control plan.

the input-output interrupts (*H* and *J*) entering the executive control for input-output, and the executive control for base level with its several priority classes of work. Most of the blocks in this figure have been referred to generally in the preceding discussion. However, a few further comments are required to clarify some points.

The maintenance interrupt blocks are designated in terms of the hardware subsystem whose trouble detection circuits generate the particular interrupt level. An appropriate fault recognition program for a given hardware subsystem is entered from the interrupt level associated with the control and display panel where facilities exist for the craftsman to configure the system when automatic reconfiguration is unsuccessful. The *G* and *K* levels are associated with specialized functions discussed in detail later.

All base level task programs are assigned to one of the five priority classes (*A* through *E*). A priority class of work known as "interject" represents an intermediate priority between interrupt and noninterrupt (base-level) programs. Section IV says more about this class. The lowest priority base-level class (*E*) is primarily dedicated to maintenance control and the programs which it schedules.

Several techniques have been used extensively to provide a clean interface between the SPC programs and the application system. One approach in the administrative programs is heavy reliance on table control. The tables of these programs allow very flexible task program scheduling and priority assignment. The SPC and application task programs may be mixed together in the tables with SPC assignments ordinarily remaining fixed.

Transfer vector tables (transfer orders to link interprogram communication) are used as a fixed interface for certain necessary linking of SPC programs and application programs. Since the application programs may be located differently in the various systems, this technique avoids consequent variations in the SPC programs.

These techniques make it possible to provide a standard SPC load tape. This is of considerable value to development organizations that use the SPC in an application system development, since they can freely alter the structure of the application programs without having to reassemble SPC programs.

III. HARDWARE

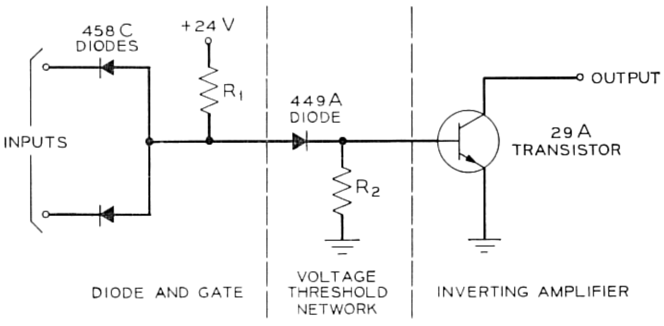
3.1 *General*

To obtain economic advantages from standardized hardware it was decided early in the development of the SPC complex to use low level logic and the existing ESS No. 1 circuit packs wherever possible.[8] Only when there was no existing circuit pack was one designed. Where possible, entire circuits and circuit functions were used from ESS No. 1 with the result that several units are nearly identical.[9] Because of this decision, the central pulse distributor, signal distributor, master scanner, teletypewriter circuits, and communication bus systems are so similar to ESS No. 1 circuits that only minimal descriptions are given here because detailed descriptions appear in Ref. 10.

3.2 *Basic Logic Gate*

Throughout the entire system the basic No. 1 ESS "and not" low level logic gate was used (see Fig. 4). The characteristics are as shown. Worst case circuit design techniques were used in order to insure reliability over working voltage and temperature ranges during the life of the office.

For a moderate sized SPC complex with 20 piggyback twistor stores,



(a)

PARAMETER	RANGE	TYPICAL VALUE
SIGNAL LEVELS		
HIGH (1)	4.0 – 5.1 V	4.6 V
LOW (0)	0 1 – 0.5 V	0.25 V
FAN – OUT	1 – 8	4
FAN – IN	1 – 20	4
TURN – ON DELAY	10 – 65 nSEC	35 nSEC
TURN – OFF DELAY	10 – 65 nSEC	35 nSEC
NOISE MARGINS		
HIGH	1.6 – 3.6 V	2.5 V
LOW	1.0 – 1.5 V	1.2 V
CIRCUIT MARGINS		
RESISTOR TOLERANCE	± 20 %	
VOLTAGE TOLERANCE	± 10 %	
POWER CONSUMPTION	37 – 140 mW	76 mW
OPERATING TEMPERATURE RANGE	0 – 55° C	

(b)

Fig. 4—Basic low level logic gate (a) and characteristics (b).

about 50,000 low level logic gates are required. Forty-two percent are used in the stores, 40 percent in the duplicated processors and the other 18 percent in the rest of the SPC complex. In addition to low level logic gates there are many circuit packs requiring higher powered transistors for power supplies, memory drivers, etc.

3.3 *Processor Description*

3.3.1 *General*

The processor, which provides the control for the system, is a computer-like circuit which executes program instructions received from the piggyback twistor stores. The processor cycle is 6.3 μ long. Execution of an instruction requires from 1 to 10 processor cycles. During the last cycle of each instruction the processor reads memory to obtain the next instruction to be executed. Figure 5 depicts the essential portions of the processor which are required for this instruction fetching. The memory address of the next instruction is contained in the 19-bit program address register.

The five most significant bits specify the name of the store to be approached, and the remaining 14 bits indicate the internal address within that store. The processor is thus capable of addressing any one of 32 stores and approaching any of 16,384 locations in that store. The total memory capacity for the system exceeds one-half million directly accessible 47-bit words.

The address image register is a 20-bit register which temporarily retains the address of the most recent communication with the store. If an error occurs during the store communication, the address to

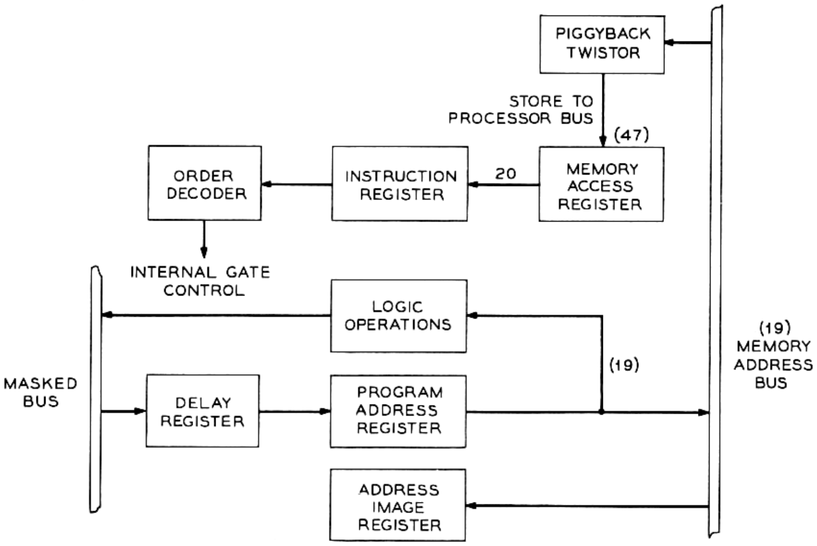


Fig. 5—Program fetching.

which readdressing is required is available in the address image register. The least significant bit of the address image register is used internally by the processor to select a half word on some instructions.

The processor performs logic with a 20-bit combinational parallel subtractor which is also capable of addition and the logical operations AND, OR, and EXCLUSIVE-OR.

The result of a logic operation may be gated to the delay register via a 20-bit bus called the masked bus. The delay register provides temporary storage for instructions which require gating of information from the program address register (or an index register) through the logic operation and back to the program address register (or index register).

The memory access register is a 47-bit register which accepts information from the piggyback twistor stores. The 47 bits consist of a 40-bit word generally containing a 20-bit operation code and a 20-bit address or data field as well as a 6-bit Hamming error correction code and an overall parity bit. The instruction register is a 20-bit register which stores the operation code for the instruction to be executed, and the order decoder controls the gating and information signals necessary for its execution.

In memory fetching, the contents of the program address register are gated onto the memory address bus, and a read instruction is sent to a store. While the processor is waiting for the store response, the program address register contents are gated to the logic circuits where they are incremented to the next higher store address to prepare for reading the next program instruction. The resultant address is gated to the masked bus and into the delay register.

When the store has completed its reading cycle, it gates 47 bits of information onto the store-to-processor bus and subsequently into the memory access register. The most significant 20 bits are gated into the instruction register where they provide inputs to the order decoder until the instruction is completed. During the beginning of the execution of the instruction, the updated program address will be gated from the delay register to the program address register in preparation for the next fetch.

There are five main classes of SPC instructions which make up the SPC program repertoire: internal data manipulation, transfer, memory reading, memory writing, and peripheral input-output orders. These instructions are designed to provide a powerful set of orders for use in a broad variety of application programs. To accomplish this objective, virtually every instruction can be executed with any of the internal index registers. Any index register may be used for operations such as address indexing, return address storage, logic operations, and peripheral unit enables. Special circuits were designed to permit some of the frequently performed tasks to be accomplished efficiently. For example, circuitry was added to provide access to a particular bit or set of adjacent bits (that is, a byte) with just one instruction. A corresponding packing option is available through another circuit for memory writing orders.

3.3.2 *Internal Data Manipulation Instruction*

An internal data manipulating instruction is used for operations in which data from an internal register are placed into a register after some arithmetic or logic operations are performed on the data. There are seven general purpose registers each consisting of 20 bits which provide storage for information manipulation. For convenience and efficiency all data handling inside the processor is done in the half word size of 20 bits. Two 20 lead digital converted bus systems provide easy access to and from these registers.

Figure 6 indicates the arrangement of the registers and the buses. Access from the registers to the logic operation circuit is provided by the unmasked bus and the argument bus. With this arrangement it is possible to perform a logic operation on any two of the registers by gating one on the unmasked bus and the other on the argument bus and requesting the desired function from the logic circuitry. The results are gated onto the masked bus and into the delay register. From there the result may be transferred to any of the registers.

Although the data words are 20 bits long, operational data themselves normally consist of much smaller bytes. A byte is a quantity defined by a programmer which can vary between 1 and 20 bits. Several of these bytes are then assembled into a 20-bit word for efficiency in storage, and a common task is to unpack a particular byte. The byte of interest is isolated by performing an ANDing operation of the 20-bit word with a register.

Figure 6 also indicates circuits which have been added to permit unpacking of the most commonly used bytes without using a register. The shift and rotate is a combinational circuit which permits shifting or rotating data to the left or the right by any number of bit positions from 0 to 19. The wired mask inhibits transmission of all information except a selected group of right adjusted bits. There are 16 different mask sizes which may be selected. These mask sizes are 1–12, 14–16, and 20 bits. Unpacking is accomplished by gating contents of a register

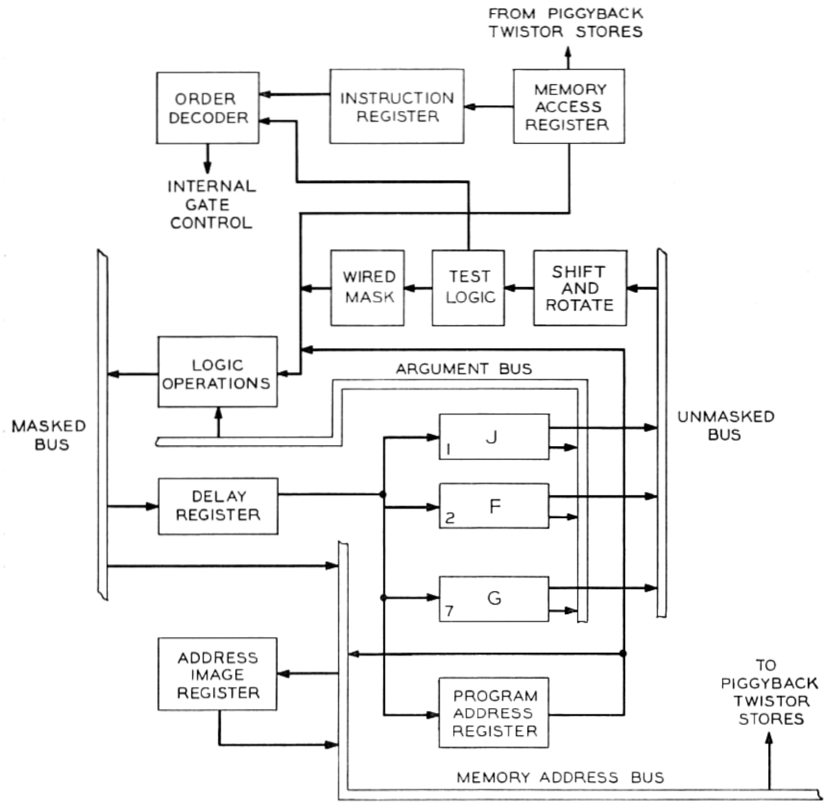


Fig. 6—Internal data manipulation and transfer instruction information flow.

onto the unmasked bus into the shift and rotate where the data is rotated until the byte to be isolated is adjusted to the right. The byte can then be passed through the wired mask and gated into one of the general purpose registers. Some instructions of this type have sufficient bits in the instruction code to specify a logic operation with a second register after the wired unpacking has taken place.

3.3.3 *Transfer Instructions*

Transfer instructions require a single machine cycle, and are basically fetching operations. If the transfer is conditional the address at which the fetching occurs depends on whether the transfer conditions are satisfied or not.

The least significant 20 bits of the memory access register contain the address to which program control is to be transferred if the transfer is consummated. This address may be augmented by adding to it the contents of any one of the seven general purpose registers in an operation called indexing. Indexing is permissable on all orders which involve store or peripheral communication.

An address is indexed by gating the right half of the memory access register to the logic operations and at the same time placing the contents of the selected general purpose register on the argument bus. These two quantities are added in the logic operations, the sum is gated onto the masked bus for subsequent gating to the memory address bus, and to the piggyback twistor stores. When a transfer is executed, the present address may be gated into any one of the general purpose registers and later be used as a link back into the present program.

When a conditional transfer instruction is executed on contents of a register, the register is tested in test logic at the same time as the indexed address is being formed. The result of the test is transmitted to the order decoder where the proper gating is established to transfer program control to the new address or to perform a normal fetch for the next sequential instruction.

3.3.4 *Memory Reading Instructions*

Figure 7 depicts the flow of data associated with a half-word memory read instruction. In a fetching instruction the 20 least significant bits in the memory access register contain the basic store address. During the first part of the first cycle this address can be combined with the contents of any general purpose register to initiate a reading sequence in the store selected.

The five most significant bits in the indexed address specify the name of the store to be approached. Name code 00000 is permanently assigned to 22 sets of registers within the processor circuit known as buffer bus registers. A particular register is specified by the remaining bits of the address. These registers are read by the maintenance programs in monitoring the operation of the processor. These registers contain as many as 24 flip-flops.

The reply from the store or buffer bus register arrives in the memory access register near the end of cycle 1. The least significant bit in the address image register is used to specify which half of the 40 bits in the memory access register is to be used. In the second cycle this data filters through the wired mask any one of the general purpose registers.

A full word memory reading is also available which can read two half words into two separate general purpose registers, but no unpacking is permissable.

3.3.5 *Memory Writing Instructions*

The half-word writing instruction consists of three main operations:

(*i*) Preread data of the memory location which is to be changed,

(*ii*) Operate logic to form the information to be written back into the location, and

(*iii*) Transmit new data to the store.

During the first machine cycle, address indexing is permitted and the resultant address is transmitted to the store over the memory address bus (Fig. 7). The proper mode bits are also transmitted to cause the store to perform a preread cycle. This results in reading a location, the address of which is retained by the store so the subsequent

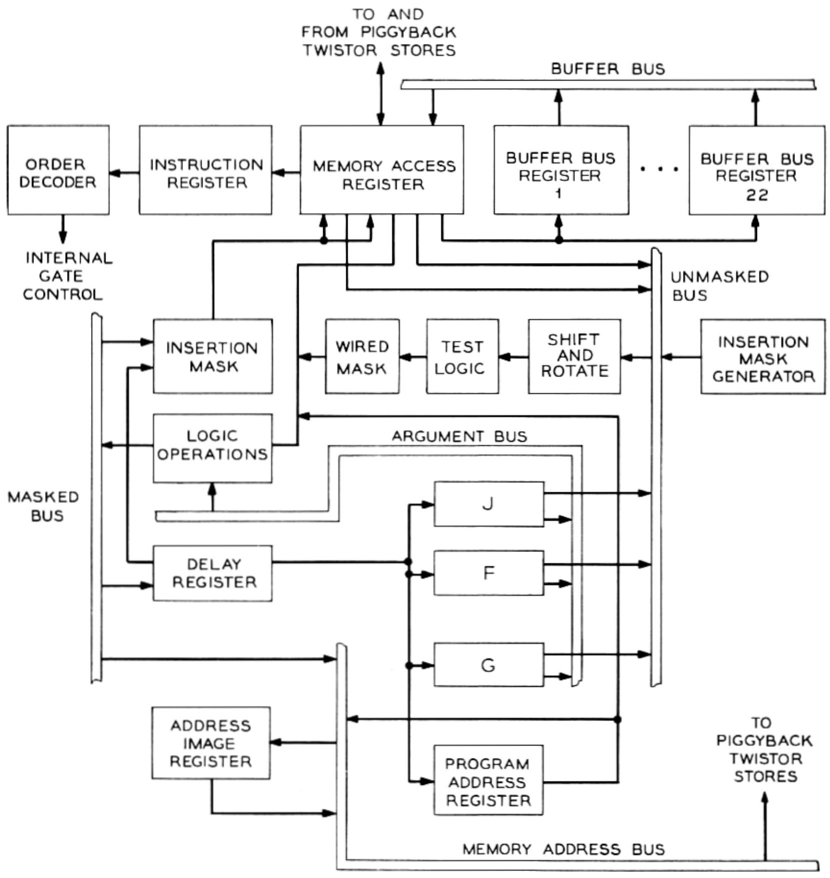


Fig. 7—Information flow for memory reading and writing instructions.

writing will occur at the same store location. The response from the pre-reading arrives in the memory access register near the end of the first cycle.

Writing often requires a packing process in which a byte is inserted in an existing word without disturbing the rest of it. This packing process (insertion masking) is accomplished with dedicated circuitry. The size of the byte is selected by program from the available wired masks.

In the second cycle the register containing the right adjusted byte or word to be written into memory is gated to the unmasked bus through the shift and rotate where it is left rotated to the proper bit position. It then flows unchanged through the test logic, wired mask, and logic operations to the masked bus and into the insertion mask. Here it is combined with insertion mask contained in the delay register and is gated into the memory access register in those bit locations specified from the delay register. This can occur in either half of the memory access register as selected. Hamming and parity bits are generated over this new 40-bit word and the internal store address which is present in the address image register. The data are then transmitted to the store.

The third cycle consists of the normal fetch and the transmission of a "write go" signal to the store after the data to be written has been verified.

When the indexed address contains the store name code 00000 the preread and subsequent write will occur at the buffer bus registers specified by the remainder of the address. Thus these registers may be controlled in the same manner as a store location.

3.3.6 *Peripheral Orders*

There are four main ac buses associated with information flow between the processor circuit and a unit of peripheral hardware. The SPC-peripheral unit address bus is dedicated to the units which are a part of every SPC installation; the application system peripheral equipment is located on a separate peripheral address bus. The central pulse distributors are located on a dedicated central pulse distributor enable bus, and one of them is activated on every peripheral order. The scanner answer bus is provided for responses from scanner-type units to the processor. This bus serves both the SPC and the application system peripheral units.

Figure 8 illustrates the flow of information for the normal two-cycle peripheral instruction. The address to be transmitted on the address buses is formed by the normal address indexing of the right half of the memory access register and one of the general purpose registers. This

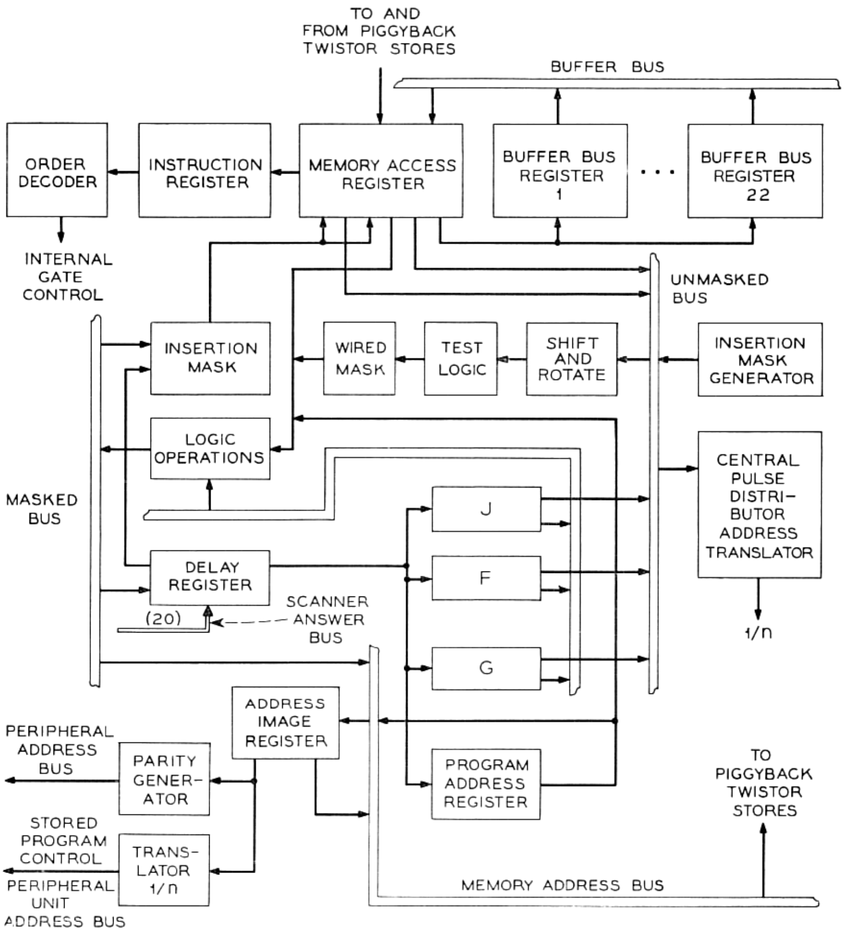


Fig. 8—Information flow for peripheral instruction.

address is gated from the masked bus to the address image register where it is stored until it is gated onto the peripheral address buses.

Concurrently with the address indexing operation, the central pulse distributor enable is gated from a selected general purpose register onto the unmasked bus and into the central pulse distributor address translator. The enable is then translated into a 1/8, 1/8, 1/16 code and transmitted to the central pulse distributor. This encoding is performed in the processor to reduce the number of logical units required in the various central pulse distributor circuits to decode the information to a particular 1/1024 selection. The internal peripheral address in the address image register is pulsed onto the two address buses in the proper time frame so the signal from the central pulse distributor and the address bus reach the chosen peripheral unit in coincidence. The data for the SPC-peripheral unit address bus is translated into a 1/8, 1/8, 1/4, 1/2, 1/2, 1/2, 1/2 code while the data on the peripheral address bus is transmitted without translation. An overall parity bit is generated over the data on the latter bus. During the second cycle of the instruction, checks are made for receipt of enable verify signals and other indications of correct transmission of all data.

3.3.7 *Features to Increase System Efficiency*

A write operation into the piggyback twistor store requires about 37 microseconds from the time the store receives the information to be written until it is ready to be accessed again. If the store passes internal tests made during the write operation, the store pulses a "write all seems well" signal to the processor about 17 microseconds after receiving the information. A write overlap sequencer enables the processor to begin processing the next instruction while the store is completing the write operation. This sequencer monitors the store for the signal and requests an E-level interrupt if it is missing. The sequencer is active until the store has sufficient time to complete its write operation. If an attempt is made to access a store which has not completed a write sequence, it will not return an "all seems well read" signal. The processor will reaccess the store each cycle until the write overlap sequence returns to normal or access is successful. All seems well read failures that occur with the write overlap sequences inactive result in normal store error sequencer action.

When an interrupt sequence is activated, control is transferred to an interrupt program. Once this interrupt program is in control, it requires the contents of the seven registers to be placed in memory so the state of the processor may be restored just prior to returning to the main interrupted program. To write this information in the piggyback twistor stores would require a relatively large amount of system time for the J level input-output interrupt since it occurs every five milliseconds. In order to save this time, the seven general purpose registers were designed to have two flip-flops for each bit of data. When a J level interrupt occurs, the contents of the registers are gated in parallel into the auxiliary registers. When the J level interrupt is completed and control is being returned to the interrupted, program, the contents of the auxiliary registers are gated back into the main register with no loss of system time.

3.3.8 *Matching Between Processors*

The principle means of detecting malfunctions is through the interprocessor matching system. The processors are normally locked in step executing the same instructions and matching selected data between the two units. The matching signals are transmitted between processors over dc connections to minimize signal delay. Three matches can be taken in each cycle.

Figure 9 indicates schematically the two independent matching units, matcher A and matcher B, which comprise the matching system in the processors. The match control is a buffer bus register which is controllable through a memory write instruction. This register is used to control the match sources which are to be used, the times at which the matches will be made, and the action to be taken on an abnormality. The A match unit in each processor is capable of matching any one of five groups of data against the same information in the other processor. The B matcher operates identically with the exception that it can sample any one of six different groups.

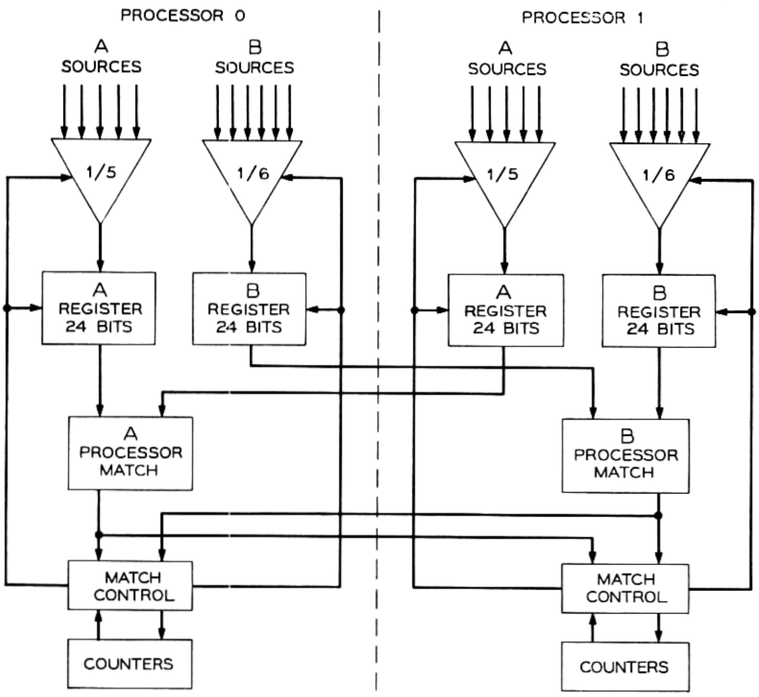


Fig. 9—Matching system.

The matching system may be operated in either the sample match mode or the directed match mode. In the sample mode, a single match is taken in one or both of the matcher units at a previously designated cycle and phase of an instruction under direction of match control circuitry which specifies when matching should occur. This sample match mode is terminated as soon as the match is taken with the match registers containing the sampled data.

In the directed mode, match control establishes a source and times for sampling for each matcher unit. Any of the sources may be specified and the sample may be taken in any or all of the three phases each cycle. These particular sources are then matched at the specified phases of each cycle until the mode is terminated by an abnormality or by program control.

The directed mode is active during normal operation with the unmasked bus and masked bus specified as the sources and a match occurring during each of the three phases. Virtually all the pertinent data associated with the execution of an instruction passes over these buses so a malfunction in a unit will be quickly detected.

Three automatic matching features are available while the matching system is operating in the directed mode:

(*i*) match on write instruction,
(*ii*) match on store error sequencer, and
(*iii*) match on store error correction.

When any of these automatic features are activated by setting a buffer bus register bit, the normally selected source will be automatically overridden for a particular cycle and phase and the memory access register substituted as the source.

When the results of the "matching" operation indicate a malfunction, the match registers and the counters are inhibited from further operation. All of these circuits are registers on the buffer bus and may be interrogated with a memory read instruction. The match registers are frozen with the data which caused the abnormality, a 4-bit cycle counter indicates the cycle of the instruction at which it occurred, and a phase counter records the phase. This information is used by the maintenance programs in detecting and isolating hardware faults.

3.4 *Control and Display Unit*

The control and display circuit is patterned after the No. 1 ESS circuit.[11] It provides a visual indication of the system status through indicator lamps and allows manual control of the system through keys and switches. Figure 10 depicts the arrangement of the apparatus used in its operation.

There are trouble lamps associated with the system as a whole and with the separate SPC units. When a system or unit malfunction occurs, an alarm is sounded and the corresponding trouble lamp is lighted. Lamps are used to display trouble active and power status. If the system becomes inoperable and cannot recover, manual control may be assumed. The craftsman may force any combination of stores and active processor for the first eight store frames. He may also isolate power from a particular store bus, peripheral bus, or central pulse distributor bus system.

3.5 *Program Tape Unit*

One of the major advantages of the piggyback twistor stores is their ability to change stored program and data electrically from the program tape unit by programmed means.

To load information into the piggyback twistor memories, the program tape unit, under system control, reads characters from its magnetic tape and places corresponding signals on groups of ferrods in the master scanner as shown in Fig. 11. The processor then reads and assembles the characters into a 40-bit word, combines this with the address of the store location which is to be changed, generates a hamming and parity code, and then writes the information with the hamming and parity bits into the proper store. Synchronization of the program tape unit output with the SPC processors is controlled by clocking pulses originating in the processor. The program tape unit also can be used to write information contained in memory on tape. For this type of operation the processor reads the piggyback twistor memory. The 40-bit word received from memory is broken up into five-eight-bit characters, a parity bit is generated over each character, and the characters are transmitted in sequence to the program tape unit over the peripheral bus system.

The format of information on the magnetic tape is shown in Fig. 12. Identity words are used to identify the blocks of data. The tape is prepared and written at 200 BPI density with non-return to zero writing mode. Following each block of characters is a longitudinal parity check character to verify the validity of the data forwarded to the processor. Each block of data is followed by a $\frac{3}{4}$ inch inter-record gap. Reading and writing is done at five inches per second tape speed to synchronize transfer of information with the five millisecond J level interrupts.

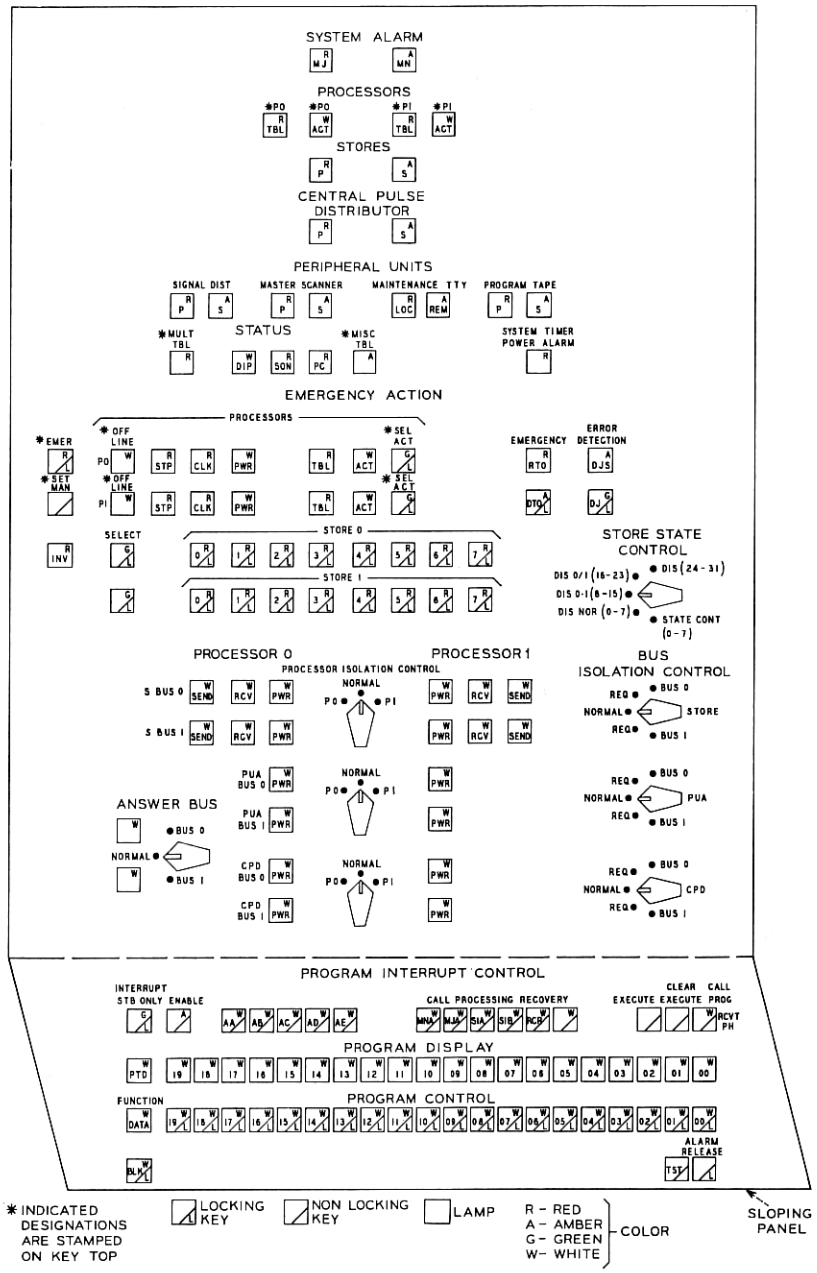


Fig. 10—Control and display.

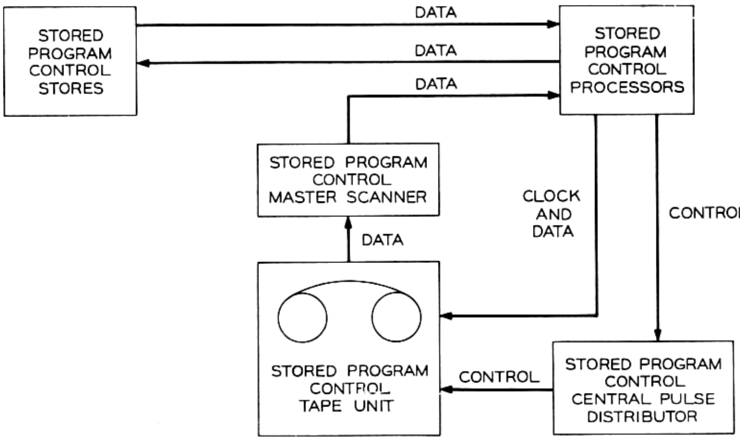

STORED PROGRAM CONTROL STORES
DATA
DATA
DATA
STORED PROGRAM CONTROL PROCESSORS
STORED PROGRAM CONTROL MASTER SCANNER
CLOCK AND DATA
CONTROL
DATA
STORED PROGRAM CONTROL TAPE UNIT
CONTROL
STORED PROGRAM CONTROL CENTRAL PULSE DISTRIBUTOR

Fig. 11—Block diagram of program tape unit.

3.5.1 *Read Mode*

Since the magnetic tape is written in the non-return to zero mode, the presence of a logical one is indicated by a flux reversal. The read head output consists of bellshaped bipolar pulses corresponding to appropriate flux reversals. These pulses are linearly amplified and then sent through an analog-to-digital converter (Fig. 13) which produces unipolar square pulses with leading edges at the corresponding peaks of the original bipolar pulses. After delays for bit deskewing and synchronization, the contents of the buffer are gated into a normal register slot; a general reset then occurs which resets the buffers, and the distributor is advanced to provide a new slot for the next character to be read from tape. The register has six slots each containing the nine bits of one character. Words are picked up by the processor at five-millisecond intervals.

3.5.2 *Write Mode*

The write mode is initiated by the processor. When the processor desires to transfer information from the store to the program tape unit, it signals via central pulse distributor points to condition the program tape unit circuit for writing and starts the tape transport running in the forward direction. The processor then transmits unit five or six characters sequentially to the program tape. These characters are held in the six normal register slots. The processor then signals the program tape unit to begin writing a record block. One character is written on tape

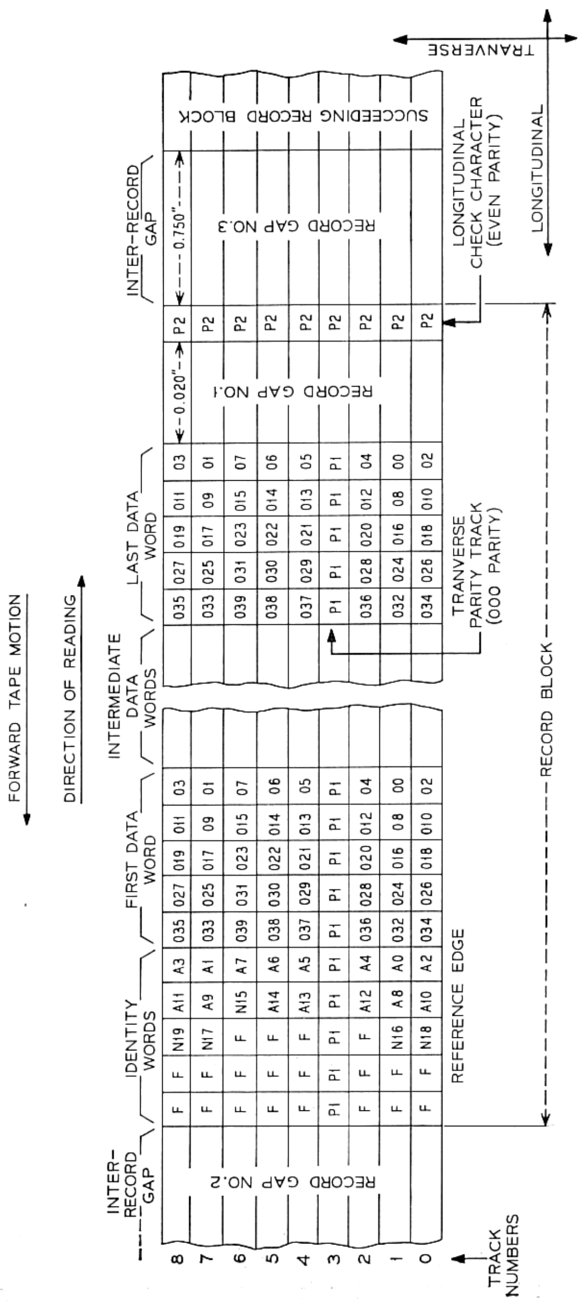


Fig. 12—Format of magnetic tape.

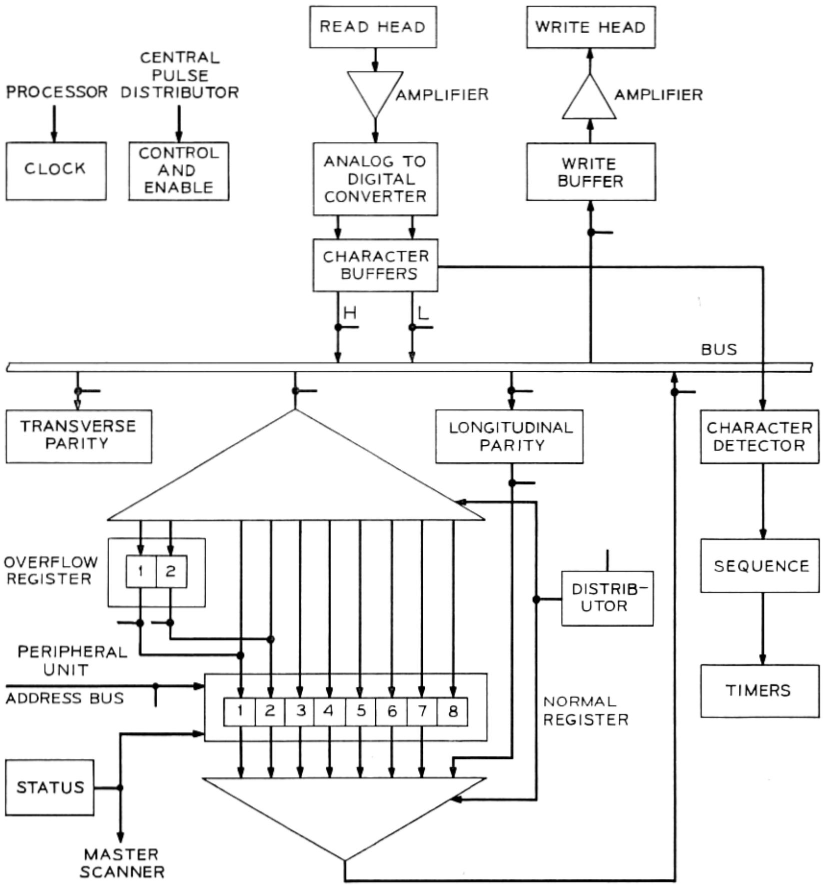


Fig. 13—Program tape unit.

each millisecond. Before each character is written, it is checked for transverse parity. Five normal register slots are emptied each five milliseconds, and the processor fills emptied slots with new characters. After a maximum of 151 words have been written, the processor will signal the program tape unit to end the record block. The processor times for the interrecord gaps.

3.6 *Teletypewriter Circuits*

Teletype circuitry as used with the SPC is almost identical to that used in No. 1 ESS and needs little explanation. The interface buffer between the teletypewriter and the processor is a low level logic circuit connecting to the peripheral bus. In operation, an enable signal is received from the central pulse distributor to enable the teletypewriter to accept the information which appears on the address bus. The character or control information for the teletypewriter is received in parallel from the peripheral bus and is shifted out serially toward the teletypewriter at a pace compatible with the mechanical speed of the machine. The teletype buffer receives a character when a message is being typed. The design of the interface buffer is arranged to simplify diagnostics of both the electronic and electromechanical parts of the teletypewriter.

3.7 *Communication Bus Systems*

To provide communication between the component parts of the system, an ac bus transmission system is used. These are parallel twisted pairs running from unit to unit in particular groupings called buses. They carry information words in the form of 0.45 microsecond pulses. Pulse generating and receiving circuitry is designed to produce recognizable signals at the receiving circuits of all the units connected to the bus. Bus impedances are about 50 ohms at the driving point. Since only one order may be sent in a 6.3 microsecond cycle, the duty cycle for pulsing is less than 10 percent thus allowing operation of the bus receivers without dc restoration.

The ac bus driving transformers are well shielded and the system is balanced with respect to ground. Contact protection networks are extensively used in relay circuitry to prevent generation of transient voltages. Paired leads are used wherever possible to minimize coupling of noise into the bus system.

Peripheral unit bus lengths can be as great as 450 feet. Store buses are restricted to 100 feet in order to minimize propagation delays for proper store/processor communication.

3.8 *Central Pulse Distributors*

The central pulse distributor is used to provide the SPC processor with directed access to many points within the system which require fast response to the instructions being transmitted. The unit is a translator decoder which takes an equipment address and provides a unipolar or bipolar pulse output on one out of a possible maximum of 1024 points. Address storage registers at the input of the central pulse distributor connect to parity error detection circuitry which check the input information for accuracy. The address registers simultaneously prepare a path through a matrix to an output point of the central pulse distributor. If the error check is satified, a pulse is applied to the apex of the matrix and appears at the selected output point. If internal checks for correct operation are satisfied an all seems well pulse is returned to the processor indicating the SPC has functioned properly.

3.9 *Signal Distributors*

The signal distributor is accessed from a peripheral bus and therefore uses low level logic circuitry as an interface with the system. Internally, however, nearly all logic is performed by relays. The selection matrix consists of relay contact trees which form a selected path under control of the input address information. The signal distributor is intended to operate magnetic latching relays, in response to instructions distributed by the processor. The apex of the relay tree matrix can connect to a -48 volt signal or a $+24$ volt signal to operate or release magnetic latching relays connected to the output points. Operation of the magnetic latching relay causes a pulse to be generated along the energizing path back through the matrix. When this signal is detected, the current to the relay is interrupted and the input registers are reset.

3.10 *Master Scanner*

Master scanners are used throughout the system for supervisory input from the various equipments to the SPC complex. A master scanner contains a matrix of ferrods connected in such a manner that they can be interrogated in groups of 16 to determine whether or not the ferrite material is magnetically saturated. Saturation of the material occurs when the point to be monitored causes current to flow through a winding surrounding the ferrite rod. A saturated ferrod produces a very low output in the output winding when interrogated, a non-saturated ferrod produces a high output. The output of the ferrods is transmitted to the SPC on the scanner answer bus for the use of the processor, in the handling of calls or execution of other work.

IV. ORGANIZATION FOR RELIABILITY AND MAINTAINABILITY

This section is concerned with those aspects of the SPC No. 1A hardware design and organization which are provided specifically to facilitate maintenance and achieve the over-all system reliability objective[12,13]. Thus, it describes the equipment redundancy scheme, the arrangements for switching, access to the various equipments, special circuit facilities for emergency recovery from troubles, maintenance instructions, and so on. Sufficient description of the workings of each function are given to indicate the basis and validity of the choices. However, the section does not discuss the complete strategies for recovery from trouble conditions. Rather it is intended to provide the background for such a discussion which will be given in the following section on the SPC program.

4.1 *Redundancy*

Duplication of subsystem equipment units in the SPC is provided for dependability rather than for an increase in the traffic handling capacity of the system. The memory subsystem consists of two store groups, each of which contains numerous information blocks (or stores). Each store of a particular store group has its corresponding image in the opposite store group. The stores contain two distinct categories of information: semipermanent data (program and parameter), and temporary data which may be intermixed in a store. The processors normally run in parallel, synchronously executing the same program, each from its own associated store group. Basic processor store communication is achieved with a partially dedicated bus system using a hamming code plus parity scheme for error detection. This system is arranged in such a manner that a store can only receive and answer over one bus, and each bus of the duplicated bus system has access to only half the stores. At the processor end of the bus, the static control of send and receive modes between each processor and the store buses is augmented by dynamic switching of the store answer bus by either or both processors using a store name match circuit. This program controlled circuit informs a processor when only one of the duplicated stores is in service and from what bus the answer can be expected.

4.2 *Store Communication*

4.2.1 *Control*

Generally, all communication buses are completely switchable at the SPC processor. Hence, various SPC processor communication bus modes can be established by means of unique configurations of the buses and processors. The output from a flip-flop in each processor dictates whether or not that processor is active. Communication channels to peripheral units and central pulse distributors are normally established only for the active processor. Since a change in the status of the system is established by means of the central pulse distributors, the active processor is in command of the system configuration.

The active processor selects the bus configuration for the appropriate processor-store communication mode and thus dictates the treatment of instruction and data communication to and from the stores. It can send on either or both buses while the standby can only send on a bus not being used by the active unit. Normally, each processor communicates with a separate copy of memory (fully independent access); but store communications control is provided to allow flexibility in choosing different bus configurations.

Some of the functions of the store communications control are:

(*i*) to specify the active processor store bus,
(*ii*) to allow each processor to independently communicate with a bus,
(*iii*) to allow the standby to receive from the active processor store bus and deny the standby write access,
(*iv*) to allow the active processor to send over both buses (implied denial to standby), and
(*v*) to allow both processors to receive from both store buses.

4.2.2 *Store Failure Bus Modes*

When the system has a single defective store, the system is reconfigured to establish a mode in which the active processor communicates with the good store group. The standby processor performs as dictated by the store communication control except when communicating with the defective store on its associated bus. When the standby processor addresses the defective store, it will momentarily switch its receiving bus so as to receive from the good copy. This switching mode is established by a program action that places the name of the defective store in a name code recognizer register in each processor. A program controllable bit is provided which specifies the receive bus that should be used by both processors when a match of the defective store name occurs.

If a second store failure occurs when one store in the system is out of service, the system can still function as long as the second failure is not in the mate of the off-line store. When there are two or more stores inoperable and all are on the same bus, the name code recognizer circuits will not be used. The processor-store bus mode which is used causes the active processor to transmit over both buses with both processors receiving from the active bus, i.e., the bus containing the full complement of stores.

When a single store is out-of-service on each bus (not mates), each processor uses its name code recognizer. Each processor transmits and receives over its respective bus except when an inoperable store is addressed. At this time, the processor which is addressing the defective store will receive from the other processor's bus.

When there are more than two stores defective with only a single store defective on one bus, the name code recognizer circuits of each processor are used to specify this store in both processors. The active processor is configured to transmit over both buses; and both processors are configured to receive from the bus containing the single defective store, except when that store is addressed. Both processors then receive over the other bus, i.e., the bus which has the good copy. This mode keeps the good stores on the standby bus updated although their data is not used.

When there are two or more stores defective on each bus (not mates), it becomes necessary to operate the store complex without redundancy. The action prescribed is to shut down one copy of the remaining redundant stores and to place the system in a mode in which the active processor transmits over both buses and both processors receive over both buses. In this mode of operation, the store name code recognizer circuits are not used. Any subsequent store outage during this mode will necessitate use of the emergency action facility to recover a working system if sufficient working stores remain. This mode of operation is used as a last resort to keep the system running.

4.2.3 *Selective Store Addressing*

The operational programmer who uses the basic instructions for writing data processing routines assumes troublefree operation and ignores subsystem duplication. The maintenance programmer, however, requires some other orders in addition to the operational instructions. These additional orders are used in fault recognition, diagnostic and routine exercise maintenance programs.

Maintenance instructions provide access which is either inconvenient or impossible to obtain with combinations of normal orders. Indirect paths to the desired points sometimes exist via the normal instructions but require reconfiguration of the interconnections between subsystems.

To test store buses, stores, or processor access to a particular store bus, selective addressing of either the 0 or 1 subsystem is accomplished within the processors by a special routing control. This control works in conjunction with a special set of instructions to allow selective addressing of stores, bus testing, selected access to internal points within a store or processor, etc. The order control signals are transmitted from that processor selected by the maintenance programmer in setting up the special routing control. Subsequently, both processors receive the reply.

Hamming, parity, and all-seems-well (ASW) checks are made by the processor(s) when executing maintenance instructions. But, since troubles may be expected to occur during these tests, no corrective action is taken. All responses to store errors are inhibited so that data containing errors may be analyzed.

Special bus testing maintenance instructions are provided in conjunction with special store hardware to test the input-output control leads between the processor(s) and the store complex.

4.3 *Processor Reactions to Store Errors*

In the fully duplicated SPC system, single errors are corrected using a Hamming error correction facility. The results of correction are then compared by the processors, and, if the match is successful, processing continues. If not, a reread is performed, because failure to match after correction indicates the original error was an odd, multibit error.* Reread failure now leads to a processing interrupt, resulting in store system reconfiguration. With any configuration, double errors, address errors, and ASW read errors immediately cause a reread; and reread failures lead to a store interrupt. Failures to write immediately lead to an interrupt.

With at least one processor or store out-of-service, the Hamming error correction facility is inhibited, because, on some or all reads, matching is not available to detect false error correction. Consequently, single errors are treated like double errors. If store errors which cause interrupts occur and the failing store cannot be removed from the system because its mate is faulty, the system's ability to function with single error correction must be checked. If it can be proved that the system is making only single errors (through use of test word reads), the system maintenance program will attempt to run the system with single error correction. Otherwise, manual intervention is required.

4.4 *Processor(s) Sending to the Store System*

A preread feature is provided on all write operations so a Hamming check can be made to insure that the internal address read from the store is the one intended for the write. The store retains the preread address for the write operation. Protection against erroneous writing into another store is provided by a parity check on the store name (higher-order bits of the store address). Because of the preread check, erroneous writing in the desired store is essentially limited to writing

* The Hamming code detects and corrects single errors and detects all double and some higher order even errors. It detects all higher order odd errors but treats these as though they were single errors.

erroneous data at the correct address. Invalid data put into the correct store at the correct address should be detected by the Hamming or match circuitry on a later read of that address. If the other copy exists, the processor(s) can continue without difficulty. In the fully duplex mode of operation, the Hamming check over the data and address field is backed-up by a processor match of the data returned independently to each of the two processors.

4.5 *Peripheral Communications*

SPC peripheral unit input-output orders use an enable code to identify the unit which is to receive the order. Certain bits of the code select a central pulse distributor and provide address data used by the central pulse distributor to select an output lead over which the central pulse distributor will send an enable signal to the particular peripheral unit. The data being sent to peripheral unit (which includes an internal selection code) is routed to a particular peripheral unit controller of a duplicate pair, depending upon which peripheral unit and central pulse distributor address buses are selected in the enable code. The enable signal from the central pulse distributor allows the information to be gated from the address bus to the proper peripheral unit controller.

Should any central pulse distributor, bus, or peripheral unit controller fail, there is sufficient redundancy to allow access to the peripheral unit via another route. This requires a change in the associated enable code(s). Most of the bits for the enable code in a peripheral unit are fixed. However, the two bits which control the choice of the peripheral unit address bus and the choice of central pulse distributor (and thereby indirectly the choice of a peripheral controller) vary as trouble conditions affect the status of these units. These two bits are often referred to as the "routing" bits. Since they reflect the status of the peripheral unit system, maintenance programs are required to keep the entire complex of enables updated with each change of the system.

4.6 *Maintenance Control of System Sanity*

When an error in the store containing the interrupt programs leads to an interrupt, there exists the possibility of system insanity. Two defensive mechanisms are used here. First, store communication at the interrupt level is limited to the store containing the interrupt program until the program has established that the system is operating properly. This protects system data. Second, an attempt is made by hardware logic to provide the system with the good copy of the interrupt program. This function is performed by conditional bus switching at the time of interrupt to the store bus *not* indicating the error. This function can be inhibited (under program control) if there is only one copy of the interrupt program available to the system and if that copy is in the store on the active bus.

4.7 *Emergency Action Functions*

The purpose of the emergency action circuit is to monitor system sanity through a variety of independent timing circuits, and to provide a sequencer for selecting a new hardware configuration if these circuits detect trouble.

A timer is provided for monitoring system sanity at all times. This timer, called the long timer, is active whenever its associated processor is active. This circuit times for 630 milliseconds; if, at the end of this interval, the timer has not been properly administered, it will cause an emergency action B-level interrupt. Two independent program controllable timers are activated by the emergency action circuit when a system malfunction is detected. These timers must be reinitialized periodically or the emergency action circuit will create an emergency action B-level interrupt. These timers can also be used to provide defensive protection in cases where a program action might result in an insane system configuration.

The emergency action sequencer controls the reconfiguration of the system during an emergency action resulting from the emergency action B-level interrupt by sequentially selecting combinations of processor and store buses. If all combinations of processor and store bus have failed to pass the tests administered by the emergency action program, an emergency action alarm will be activated. This indicates to the maintenance personnel that the system cannot recover.

An emergency action program is provided to complement the function of the emergency action sequencer. This program administers a series of hardware tests to determine the acceptability of the sequencer established configuration. These actions are discussed in Section 5.3.3.

V. DESCRIPTION OF THE SPC PROGRAM

5.1 *Maintenance Functions*

This system depends primarily on hardware-checking circuits for trouble detection during operation. When a trouble detection circuit locates a problem in the system, it notifies an interrupt circuit. The interrupt circuit immediately stops operational program processing and transfers control to a fault recognition program associated with the particular type of trouble indication. The functions of the fault recognition programs are to distinguish nonrepeating troubles (errors) from repeating troubles (faults) and, in the case of faults, to quickly determine an operational system configuration, establish it by switching out faulty units, and then return to operational program processing.

Fault recognition is the most important maintenance function in a real-time system. The minimization of time taken from the operational processing function and an extremely high probability of returning to processing with an operational (sane) system are the critical objectives which any automated maintenance technique must satisfy. Diagnostics with "fine" resolution are of secondary importance.

5.1.1 *Processor Errors*

The processor retrial program is designed to qualify a processor trouble indication as an error or a fault. The program also interrogates critical areas within the processors; and, if they are found to be faulty, it isolates the fault to the active or standby processor, records pertinent processor error information, and initiates actions required to return the system to normal operation at the conclusion of the program.

Since errors are expected to be more frequent than faults, and, as it requires considerable time to completely check both processors, the mismatch is first assumed to be an error. As the processors inhibit destination register gating on mismatches, it is possible to re-establish the state of the processors at the beginning of the interrupted order and re-execute most instructions. To determine if the mismatch was caused by a fault, the failing instruction is unwound and control is returned to the interrupted program. In the process of preparing the processors for re-executing the failing instruction, a hard fault may again make itself evident. In such a case, retrial of the order is unnecessary and actions will be initiated immediately to remove the faulty processor from service. If the instruction retrial causes another C-level interrupt, the mismatch is presumed to be a fault, and control is transferred to a complete check control routine in the processor fault recognition program.

The processor retrial program maintains error counters based on the results of processing mismatches. If any of the error counts become excessive during a given time period (implying an undetected processor fault), control is transferred to the complete check control routine. Otherwise, control is returned to the interrupted program for retrial of the failing instruction.

When a mismatch occurs, the C-level interrupt hardware sequence starts the 2-millisecond and 40-millisecond emergency action timers (See Section 4.7). If the active processor has a fault which impairs its ability to properly execute the retrial program, a time-out will occur and control will be switched through a B-level interrupt to the emergency action program.

5.1.2 *Store Errors*

Two basic facilities are provided in the maintenance programs for dealing with store errors. One is essentially a data collection facility for gathering pertinent information regarding the nature and source of store errors. It is expected that this will be useful to the maintenance craftsman for correcting problems that are causing infrequent but generally consistent store errors. The second facility is an automatic error analysis and reconfiguration strategy that is aimed at the problem of error bursts. In this case, the error rates are so high that immediate and automatic recovery actions are required to avoid severe system penalties.

The G-level interrupt is used to invoke the program that performs store error bookkeeping. This interrupt occurs after any single error is successfully corrected or after any multiple error is eliminated on a reread. The G-level interrupt mode is normally active. However, it is deactivated when certain maintenance programs are running to avoid undesirable interaction. Also, a program governor action is present which limits the number of G-level interrupts to approximately two every two seconds. Allowing more interrupts would probably only generate redundant data and cause an excessive overhead (real-time) penalty.

The G-level program records counts of the various types of errors (repeating single, nonrepeating single, multiple) for each store, internal store addresses for the first 15 errors of each type, and the failing bit for the first five occurrences of repeating single bit errors. The tables, if nonempty, are printed out on the teletypewriter every hour and then cleared.

If transient single errors are prevalent, the craftsman can request a special matching mode that matches the received data before error correction occurs. The resulting mismatch on an error will invoke the C-level interrupt program which generates a printout that identifies the store, the address, and the failing bit.

The error analysis program which is concerned with error bursts or sustained high error rates performs a periodic scan of hardware single and multiple error counters. If either of these counters overflows in a specified interval, the error analysis mode is initiated. The analysis program collects error data for an additional period of time and attempts to determine from this data whether the errors are caused by store, store bus, or processor trouble. If the analysis is successful, it removes the noisy unit from the system configuration and identifies the unit by a printout. It also causes an immediate dump of the G-level store error tables which should provide useful information for localizing store troubles. A diagnostic is requested for a suspected noisy store, but the store is not automatically returned to service if the diagnostic passes. Because of the intermittent nature of errors, it is likely that the diagnostic will pass. Thus, the system provides printouts of all pertinent data on the problem and leaves the decision to place the unit in service to the judgment of the craftsman.

5.1.3 *Peripheral Unit Errors*

Malfunction of an order sent to a peripheral unit is detected through immediate checks made on the communications with the units and by a gross check of whether the unit successfully completed the desired operation. The immediate checks include: a central pulse distributor execute return signal indicating that the proper central pulse distributor was enabled; a central pulse distributor all-seems-well signal indicating normal functioning of the central pulse distributor; an enable-verify signal indicating that the peripheral unit received an enable signal; a scanner all-seems-well signal indicating successful readout of a scanner row (in the case of a scanner); and, in the case of the application peripheral address translator, an all-seems-well indicating successful address translation for data transmitted to the application peripheral unit bus. The signal distributor provides a gross check of normal operation by indicating through scan points that it has returned to an idle state after executing the order.

Failure to receive the immediate check signals result in an F-level interrupt. The fault recognition program at this level handles all problems associated with high-speed peripheral orders; i.e., those not directed to units involving relay operations or other such long-work-cycle operations. Discrimination of errors from faults is derived by repeating the high-speed order in the F-level and observing whether the failure indication is repeated.

For the slow operating units such as the signal distributor or similar applications system equipment, retrials are performed by a J-level program. At a later time, the order that caused the F-level is retried in J-level to determine whether the failure was due to an error or a fault. Troubles detected by checks within the signal distributor or similar frames are indicated to the programming system through Master Scanner ferrods. These ferrods are checked in J-level every 25 milliseconds. This check occurs just before an order is sent to the unit by the peripheral order buffer execution program. The failing order is then retried in J-level. This retry will determine whether an error or a fault occurred.

The procedures described above apply to all SPC peripheral units except the teletypewriter and the program tape unit. Maintenance on these units is initiated by the maintenance craftsman. The strategy applied to application peripherals will depend on the characteristics of the device.

With any of the units discussed above, a successful retrial will result in immediate termination of activity. The program involved in error detection may increment a counter or print a message indicating the nature of the error, but no further actions will be performed. The suspect unit is returned to normal service, and program execution returns to its normal schedule.

5.1.4 *Processor Faults*

In the SPC No. 1A system, the processor fault recognition and processor diagnostic programs have been integrated. This method assumes that, with sufficient match access, fault recognition can be a subset of diagnostics. That is, those procedures which test the logical capability of a processor also provide results for diagnostic resolution. The programs will be discussed as separate entities, but one should keep in mind that the test sequences are one and the same. Only the control programs are separate entities.

For repeating troubles, the processor fault recognition program determines which processor is faulty, switches this unit out of service, records significant error information, and places a request for a diagnosis of the faulty unit. Because it runs during the C interrupt level, the fault recognition program checks only those circuits which could have caused a mismatch and those peripheral system communication circuits which may not have been in use but will be employed in the new active processor. A complete analysis of the faulty processor is made by the processor diagnostic program at a later time. It is important to understand that the objective of fault recognition is to find a working configuration. This means that the faulty processor might be active and diagnosing the good processor. This is possible when faults are in areas such as the following: processor-external bus

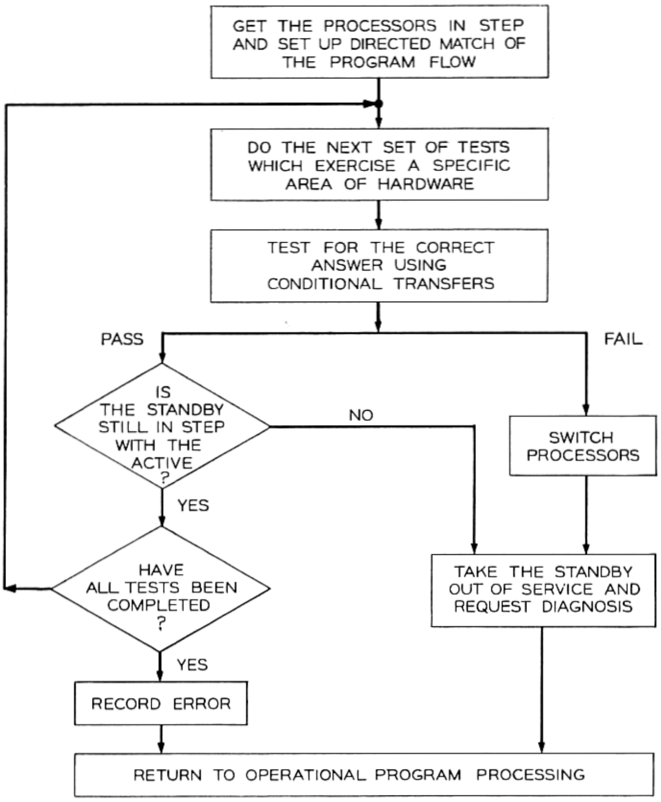

GET THE PROCESSORS IN STEP AND SET UP DIRECTED MATCH OF THE PROGRAM FLOW
DO THE NEXT SET OF TESTS WHICH EXERCISE A SPECIFIC AREA OF HARDWARE
TEST FOR THE CORRECT ANSWER USING CONDITIONAL TRANSFERS
PASS
FAIL
IS THE STANDBY STILL IN STEP WITH THE ACTIVE ?
NO
SWITCH PROCESSORS
YES
HAVE ALL TESTS BEEN COMPLETED ?
TAKE THE STANDBY OUT OF SERVICE AND REQUEST DIAGNOSIS
YES
RECORD ERROR
RETURN TO OPERATIONAL PROGRAM PROCESSING

Fig. 14—Processor fault recognition test mode.

communication circuits not presently in use by the existing configuration, the match circuits, and processor-processor interfaces. The diagnostic program has been designed to detect and correct for this anomally.

The basic program flow technique used within test phases in the fault recognition program is shown in Fig. 14. The tests are designed as if each processor is testing itself. Tests consist of data manipulation operations which check for the proper circuit response followed by conditional transfer orders. If the active processor fails, it will switch processors (which it alone can do by program). The faulty unit, now standby, will be removed from service and the diagnostic program requested. If the active passes, checks are made to see if the standby is following the operations of the good active unit (by examining the match circuits). If the standby is found to be out of step, it is removed from service. This testing process is continued until all tests have passed or a faulty unit is found.

The fault recognition program is entered by the retrial program when that program decides that a hard fault exists in the system (i.e., two consecutive C-level interrupts occur at the same program location) or when the retrial programs' error counters have exceeded their specified thresholds. The processor fault recognition program is also entered via a B-level interrupt from the emergency action program after a normal processor switch to test the new configuration. In this case, the processor fault recognition program performs internal tests (sanity), establishes and tests new store configuration, tests peripheral unit communications and configures it, if necessary, and returns the system to normal operation.

If the program was initiated by a normal processor switch or by a processor mismatch, the standby processor is made to operate in step with directed matching of the masked bus and unmasked bus. The standby processor is stopped whenever it disagrees with the active processor. If the program was initiated by an emergency action, only the active processor is tested and only the basic sanity tests are performed.

The sanity tests include arithmetic and logic functions, transfer decision, writing and insertion mask, store error detection and correction, and instruction decoding. The completion of the above tests indicates that the active processor has passed the tests. The store name clamp, which limits processor access to the base store, is removed before further tests which check auxiliary index registers, store addressing to the store with a complementary address name of the base store, and store address interaction.

When the fault recognition program is ready to release the system from interrupt control, measures are taken to determine if the return to normal system operation can be to the interrupted point (the point at which the mismatch occurred) or must be made to an arbitrary reference point. If the return is to the interrupted point, the retrial program will attempt to unwind the interrupted instruction. If the retrial program cannot unwind this instruction, it transfers to the fault recognition program, which forces a return to an arbitrary starting point (reference return).

5.1.5 *Store Faults*

Normally, a store reread or write failure creates an E-level interrupt and calls the store fault recognition program. The basic steps the program follows are:

(*i*) Find the faulty store, bus, or processor. Isolate the failing unit from the rest of the store-processor complex, and request an appropriate diagnostic program.
(*ii*) Update status records.
(*iii*) Reconfigure the remaining system in order to create a system configuration with maximum store duplication for call processing.
(*iv*) Return the system to normal processing.

Program transfers to out-of-range addresses or memory reads or writes into out-of-range addresses will also create an E-level interrupt which initiates the store fault recognition program control program. The basic steps the program follows in this case are:

(*i*) Recognize that the error data saved by the processors on all store errors indicates an out-of-range condition.
(*ii*) Recognize that the communications address register contained an unequipped store address at interrupt time.
(*iii*) Request a selected set of high level audits in an attempt to find and correct the error in temporary memory.
(*iv*) Request a return to a reference point in the executive control program.
(*v*) Request the maintenance restart program to turn on special timing because of possible data mutilation in the system. If this event occurs repeatedly in the next 16-second interval, this will automatically trigger a call processing recovery phase (or phases). The latter is discussed in Section 5.3.3.
(*vi*) Establish the store and processor configuration existing prior to the interrupt.

Store fault recognition program programs are, in general, organized so that an overall control program, uniquely associated with some particular broad functional task, links together a number of routines and tests. These routines and tests range in scope from simple test routines to comprehensive testing and analysis routines. Also, many contingencies can arise which do not cause E-level or G-level interrupts and, hence, do not directly engage the store fault recognition program programs. Nevertheless, they do require the testing of the store-processor complex. For the aforementioned reasons, it was desirable to organize the store fault recognition program to create a pool of routines and tests useful to the maintenance and system programs which must handle such contingencies.

The primary purpose of the store fault recognition program is to establish in a minimum amount of time a workable set of stores that provide access to at least one copy of all data. Determination of the faulty store is of secondary importance. In order to accomplish this objective, the store fault recognition program keeps a record (or maps) of the present set of in-service, and off-line but usable stores. After determining the failing unit and updating the maps, the store fault recognition program uses a generalized routine which uses the maps to determine and establish a maximum store configuration. Once this configuration has been established, this program executes a quick access check to each store unit. If this passes, it is assumed that the faulty unit has been successfully isolated. Then, on a deferred basis, the faulty unit is diagnosed.

If a configuration cannot be established because history indicates mate* stores are in trouble, or if the access test fails, this program starts a bootstrapping operation in which a bus is chosen and all stores on that bus are tested using test pattern techniques. The set of stores not passing on this bus are then tested on the other bus. If a complete configuration is obtained, the system is restarted as outlined above, and, on a deferred basis, the remaining faulty and/or untested stores are tested. Those that pass the tests are returned to system operation; those that fail are diagnosed. If the basic bootstrapping operation fails, an attempt is made to establish a working set of stores using the single error correction features; that is, previously failing stores are tested to see if they have only a single-bit oriented fault. If successful, a working system is recovered. Otherwise, manual intervention is required.

A program controllable (on/off) feature is available in the interrupt sequencer to insure that, on an E-level interrupt, the active bus after the interrupt sequence was not the bus causing the E-level interrupt (i.e., conditional bus switching on interrupt). This feature is activated by the store fault recognition program, if the store containing the E-level program is in service on both buses. It is useful in minimizing recovery time. When it cannot be used, the emergency action timers provide a back-up mechansim.

5.1.6 *Peripheral Unit Faults*

Peripheral unit faults are detected by the same checks used to detect peripheral unit errors. Some of these checks are central pulse distributor all-seems-well, enable verify, and peripheral unit all-seems-well. An error is distinguished from a fault by the success of the initial retry. All work performed on faults detected by an all-seems-well failure or

* Both copies of the same unit.

an execute reply failure is performed in F-level. The failing order is retried using a different central pulse distributor from the original order. If these retries fail, a retry is made utilizing the off-line processor. Control of the central pulse distributor and peripheral unit buses is given to the standby machine and a retry is executed. When any retry succeeds, the faulty unit is removed from service and control is returned to call processing. The procedure described for central pulse distributors is also followed for the application peripheral unit address translator. However, in the case of this unit, control is transferred to an application subprogram to complete the final cleanup work.

The SPC system contains only one unit, which returns a scanner all-seems-well; viz., the master scanner. Fault recognition for the master scanner progresses in the following manner. The order is retried using the mate master scanner controller. If the retry is successful, the original scanner controller is marked in trouble and all enables for the unit are updated. If the orders to the mate fail, an attempt is made to determine if the scanner row presently being addressed is faulty; this is done by reading other rows in the same master scanner. If other rows can be successfully read, the all-seems-well check is deleted from all enables for this scanner, and control is returned to normal processing. If neither of the above strategies has been successful, the order is executed through the standby processor in an off-line mode. The active processor will be removed from service if this procedure is successful.

When an enable verify failure occurs and the unit involved is of the fast* type, all possible routes will be tried in an effort to achieve a working configuration. Here, again, enables will be updated, the faulty equipment will be removed from service, and control will be returned to the normal processing programs.

Fault recognition for the slow units is also a logical extension of error recovery. When the initial retry procedure has failed, a series of retries using other routes is attempted in order to achieve an operational configuration. The retrial procedure is implemented in J-level on a time-shared basis to provide proper order timing. All routes are attempted to insure recovery. The first successful retry causes termination of the procedure and a request of the appropriate diagnostics. Although the details of the retry strategy may vary depending on the unit involved, the following is representative of the procedure followed. If central pulse distributor and bus status indicate a good route exists to the mate controller, the failing controller is removed from service and the order is

* A fast unit can receive and process orders at electronic speeds, whereas a slow unit requires about 25 milliseconds to process an order.

retried using the mate. When this fails to achieve success, all possible routes are tried in an arbitrary fashion. There are four possible routes to each of the slow units. The routes are determined by using either central pulse distributor with either peripheral unit bus. If none of these retries succeed, the order is executed from the standby processor in an off-line mode. If this does not succeed, no further action is taken.

The SPC peripheral fault recognition programs contain a great number of control tables to allow easy adaptation to application demands. An application system may easily provide special routines and programs for handling non-SPC equipment. In this way, the SPC peripheral system has the flexibility necessary to adapt to a wide variety of application needs.

5.1.7 *Processor Diagnostics and Exercises*

The primary objective of the processor diagnostic program is to isolate a fault in the standby processor to a small number of replaceable circuit packages. This objective is met by performing a series of rigorous tests on the standby processor, recording the pass-fail results of each of these tests, processing the resultant pass-fail test data by a dictionary number generating program into a fixed-format trouble number, and transmitting the trouble number to the maintenance craftsman via the maintenance teletypewriter. The maintenance craftsman refers to a trouble locating manual to associate the trouble number with the corresponding circuit package(s). Replacement of the circuit package(s) associated with the trouble number is expected in most instances to remove the fault from the standby processor.

5.1.7.1 *Matching Features.* The primary means of trouble detection for the processor is the matchers. They do, however, have a secondary function of providing a means for diagnosis of the processors; i.e., they provide access to many points within the processors.

Basically, the match system must be able to perform both a directed match and a sampled match. In a directed match, the matchers are directed to look at match sources* at specified time segments on a continuous basis.† In the sampled match, the matchers are directed to look at specified sources at a specified time segment, a given number

* A match source is comprised of a set of internal points within a processor; e.g., an internal bus, decoder points, sequencer status, etc.

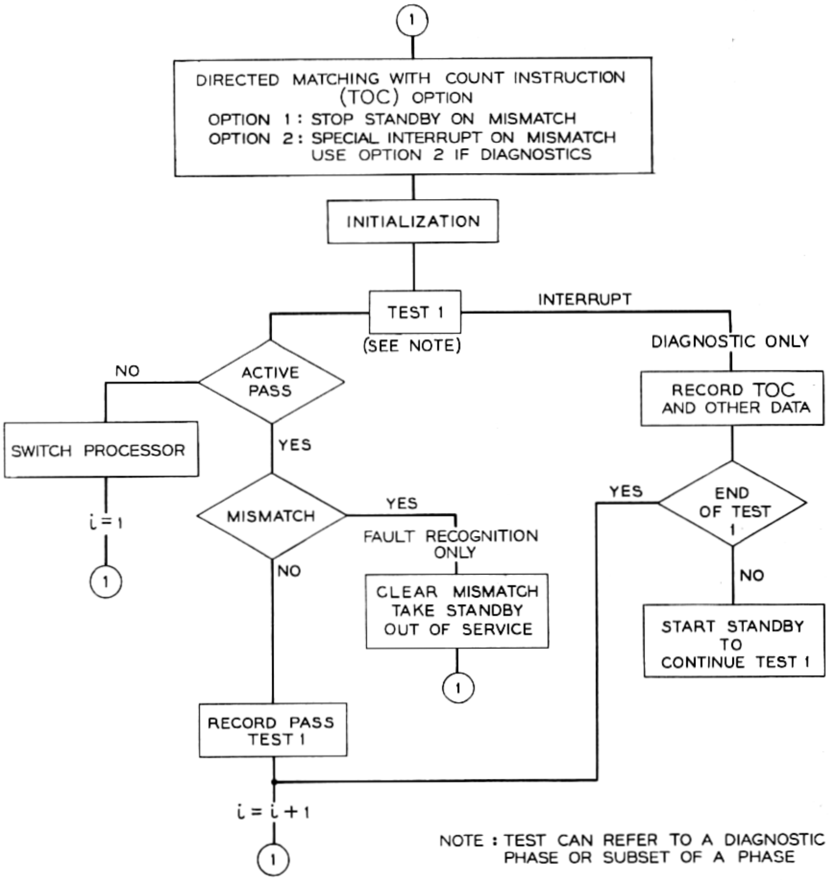
† A variation because of machine complexity is for the matcher(s) to be directed to look at a set of match sources on a sequential basis. The source to be matched may also become a function of the instruction(s) being processed.

of machine cycles from the point of initialization. This is frequently called selective, discrete, or snapshot matching. This is a powerful tool and it is used in certain "diagnostic only" tests in the integrated program method to be discussed in the following sections. However, the integrated tests use the directed match mode with the flexibility provided by varying the match sources and/or the segments of the machine cycle when matching occurs. In this mode, the specified sources are matched continuously on every machine cycle at one or more of the three time segments.

In the processor diagnostic program, it is assumed with a high level of confidence that the standby is faulty and that the active is fault-free, or that the active is faulty in circuitry not effecting the normal system operation in the system's present configuration. Thus, the active can be used to test the standby. The assumptions of single failure and no redundant logic are applied when designing tests. The execution of a test follows a strict pattern:

(*i*) initialize circuits external to the circuit being tested so their interaction with the tested circuit will be consistent,
(*ii*) apply inputs to the circuit under test,
(*iii*) deactivate or reinitialize the tested circuit when necessary,
(*iv*) compare outputs with expected outputs, or the active processor results, and
(*v*) record results indicating whether the test passed or failed. If it failed, record results about how it failed.

5.1.7.2 *Integrated Approach.* In the integrated approach, a single functional block (or phase) of tests performs both diagnostic and fault recognition functions. The majority of the tests are "integrated" tests. The basic difference between using the program for fault recognition or diagnostics is the collection of data. When this program is run as a fault recognition program, data from the standby processor is not collected. When the program is run as a diagnostic, a failure in test *i* will provide data which "points" to the *i*th executed instruction and indicates the bit pattern of mismatch. An interrupt technique invokes data recording for diagnosis (Step 5 in the preceding outline of the program structure) so that this function can be omitted when the tests are performed for fault recognition. A general flowchart for the integrated program for a test is shown on Figure 15. Note that the test now includes all instructions including initialization. Thus, this is a continuous sampling technique.



1
DIRECTED MATCHING WITH COUNT INSTRUCTION (TOC) OPTION
OPTION 1: STOP STANDBY ON MISMATCH
OPTION 2: SPECIAL INTERRUPT ON MISMATCH
USE OPTION 2 IF DIAGNOSTICS
INITIALIZATION
TEST 1
(SEE NOTE)
INTERRUPT
DIAGNOSTIC ONLY
RECORD TOC AND OTHER DATA
ACTIVE PASS
NO
SWITCH PROCESSOR
i = 1
1
YES
MISMATCH
YES
FAULT RECOGNITION ONLY
CLEAR MISMATCH TAKE STANDBY OUT OF SERVICE
1
NO
RECORD PASS TEST 1
i = i + 1
1
END OF TEST 1
YES
NO
START STANDBY TO CONTINUE TEST 1
NOTE: TEST CAN REFER TO A DIAGNOSTIC PHASE OR SUBSET OF A PHASE

Fig. 15—Integrated technique.

5.1.7.3 *Processor Diagnostic Control Program.* The processor diagnostic program may be entered to run a complete diagnosis as a result of a fault recognition program detecting a processor fault, for a demand exercise to perform one or more phases (or subphases), or for an automatic exercise of one or more phases (or subphases). The processor diagnostic control program must be able to determine the nature of the request and base control decisions on the particular function being performed.

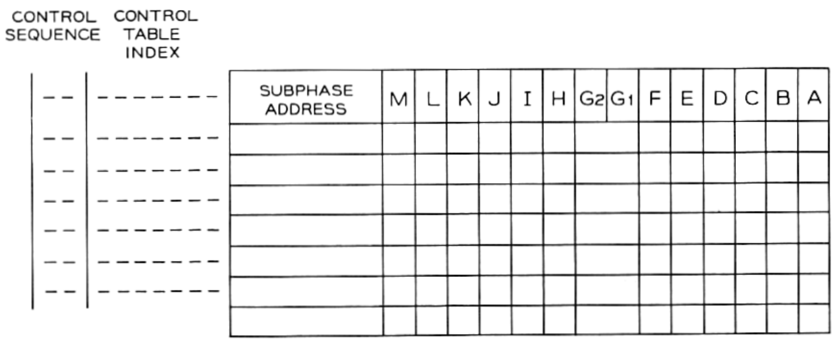
The processor diagnostic control program performs general initialization, phase and subphase initialization, linkage between phases and subphases, and termination functions.

A diagnostic program is broken into logical testing entities called subphases or phases. A phase is the series of tests performed between entries to the dictionary number generation program which consolidates this information. This dictionary number generation program preprocesses a maximum of 40 raw data (pass-fail or bit pattern of mismatch) words at a time. A subphase is used to describe a functional block of tests and comprises a series of tests between entries to the diagnostic control program. Several subphases comprise a larger functional entity which is used to form a phase and which satisfies the maximum raw data words constraint.

Processor diagnostic program general initialization includes the marking of control bits to determine which diagnostic phases should be run, the disjoining of processors, and similar details. In addition to some common initialization required by all phases and subphases, the control program uses a control table to determine special initialization necessary for the subphase to be run. The control table concept is illustrated in Fig. 16.

The control program uses a pair of ordered-bit words for linkage between phases (phases 1 through 20 are represented in a 20-bit word and phases 21 through 36 in the other). A "one" is marked in the bit position representing each phase to be run. The processor diagnostic program linkage of phases and subphases is performed by the control program which scans the two control words for the next "rightmost one". The bit position of the "one" indicates the phase number to be run. A head table is indexed by the phase number and contains the address of the first word of a control table related to the phase to be run. Subphases within a phase are sequentially placed in the control table depicted in Fig. 16, with the last subphase of a phase containing an "end-of-phase bit." This bit signals the completion of a phase and indicates a need to request the dictionary number generating program to process the data obtained. This process continues until all requested phases have been processed. Then, the dictionary number generating program is entered to produce a trouble number.

5.1.7.4 *Tests and Test Phases.* For testing purposes, the processor circuits are organized into functional blocks that are tested as independently as possible. A hierarchy of tests was established based on the critical nature of the particular circuits and processor functions that are being checked. The hierarchy of tests is the basis of the selective phase skipping and early termination procedures which will be described in a succeeding section.



CONTROL PROGRAM ACCESS TO CONTROL TABLE

1. THE CONTROL PROGRAM MAINTAINS A POINTER (CONTROL TABLE INDEX) INDICATING THE CONTROL TABLE LOCATION BEING ACCESSED.

2. THE CONTROL SEQUENCE IS ALWAYS EXECUTED IN ASCENDING ORDER OF CONTROL TABLE ADDRESSES.

METHODS EMPLOYED BY PDIAG USING CONTROL TABLE CONTENTS

1. USE BYTE A–D AS ORDERED BITS BUFFER:
   (a) LOCATE RIGHTMOST "1".
   (b) EXECUTE HARDWARE OPERATION.
   (c) LOCATE NEXT RIGHTMOST "1".
   (d) EXECUTE HARDWARE OPERATION.
   (e) ETC.

2. USE BYTE E AS AN INDEX TO A TRANSFER VECTOR TABLE TO PERFORM FIXED SUBROUTINES: AN EXAMPLE IS MATCHING INITIALIZATION.

3. USE BIT L TO DESIGNATE THE FINAL SUBPHASE WITHIN A PHASE.

4. USE THE SUBPHASE ADDRESS AS THE POINT OF ENTRY INTO THE SUBPHASE PROGRAM TO BE EXECUTED.

Fig. 16—Control table arrangement.
Control program access to control table:
(*i*) The control program maintains a pointer (control table index) indicating the control table location being accessed.
(*ii*) The control sequence is always executed in ascending order of control table addresses.
Methods employed by processor diagnostic program using control table contents:
(*i*) Use byte A–D as ordered bits buffer:
(*a*) Locate rightmost "1."
(*b*) Execute hardware operation.
(*c*) Locate next rightmost "1."
(*d*) Execute hardware operation.
(*e*) And so on.
(*ii*) Use byte E as an index to a transfer vector table to perform fixed subroutines: an example is matching initialization.
(*iii*) Use bit L to designate the final subphase within a phase.
(*iv*) Use the subphase address as the point of entry into the subphase program to be executed.

An individual test might consist of any one of, or a combination of, the following actions: (*i*) examining various state indicators for specific conditions, (*ii*) completely exercising circuits to verify the ability to assume all possible states, or (*iii*) applying inputs to circuits to effect expected actions. Groups of tests form subphases. A subphase may test one or several circuit areas.

5.1.7.5 *Recording Results of Processor Diagnosis.* There are essentially two methods of evaluating test results. In one method, the active and standby processor test results are compared. In the other method, the test results are compared with expected values. It should be noted that various logical functions are used to compare and record results within the framework of the two aforementioned methods of test evaluation.

An interrupt mechansim is used by the processor to record diagnostic test failures in a general purpose scratch area (see Fig. 15). The interrupt does not occur until the completion of the instruction causing the mismatch, so that the active processor will contain the complete data necessary to continue at the next executable instruction. This interrupt causes the diagnostic recording program to be entered. After recording the data the diagnostic program is re-entered at the next executable statement. This process continues until the end of the subphase.

Periodically, a check of accumulated test results is performed in order to isolate certain active processor faults. An active processor test failure causes the diagnostic to be aborted. A diagnostic message with data pertinent to the active processor failure is printed on the teletypewriter. This message is also useful in isolating faulty circuits being used as test tools by the processor diagnostic program.

Recording of data for all phases uses a standardized record area. Figure 17 shows the recording for a typical phase. Normally, a match source point is associated with each test area. The first four words are used for storing the mismatch bit pattern. Word five is used to record the point of execution (cycle and phase) *within* the instruction sequence at which the failure was detected for the first three failures. Each independent test is recorded "pass" or "fail" in words six through forty as needed for a given phase.

5.1.7.6 *Selective Test Skipping.* The running of phases is subject to a selective test skipping mechanism which is controlled by a table. Each phase has a control table entry which is applied in one of two ways: (*i*) the entry specifies a set of phases to be skipped if this phase fails; or (*ii*) the entry specifies a set of phases to be skipped if there have

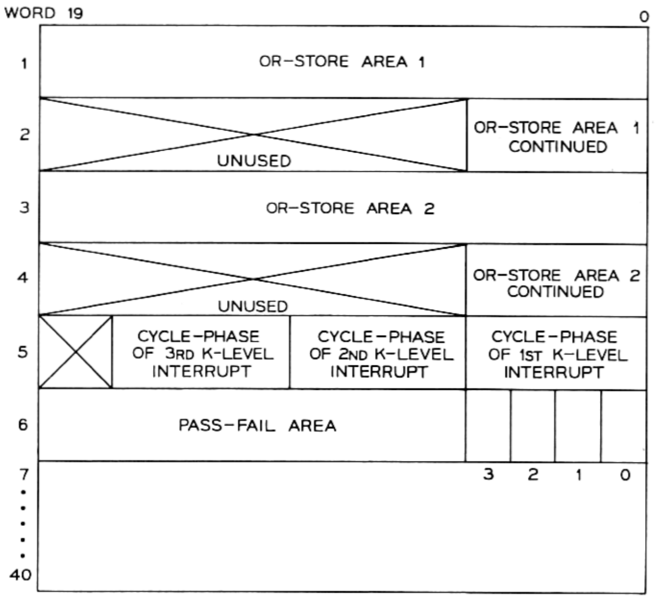

WORD 19
0
1
OR–STORE AREA 1
2
UNUSED
OR–STORE AREA 1 CONTINUED
3
OR–STORE AREA 2
4
UNUSED
OR–STORE AREA 2 CONTINUED
5
CYCLE–PHASE OF 3RD K–LEVEL INTERRUPT
CYCLE–PHASE OF 2ND K–LEVEL INTERRUPT
CYCLE–PHASE OF 1ST K–LEVEL INTERRUPT
6
PASS–FAIL AREA
7
3 2 1 0
40

Fig. 17—Typical phase recording.

been any failures recorded by the diagnostic up to and including the present phase. The skipping rules were derived logically based on the axiom that diagnosis should continue only if verified test circuits are available to test the next circuit. For example, if phase seven, matcher access tests, fails, then phase eight, matcher function testing, is skipped. The skipping table entries were later modified considerably as a result of sample fault analysis.*

An additional abort rule compares the accumulated number of distinct tests which have failed over a number of related phases with a threshold. If an unacceptable number of failing tests have been recorded, the diagnostic is terminated. This rule is used to minimize inconsistent results by ending the diagnosis if many basic tests fail, thereby preventing the running of more sophisticated tests which could lead to confusing results. A second purpose of this abort rule is to avoid giving the standby processor control of a store bus and/or peripheral bus if a basic fault has already been uncovered. This prevents adverse interaction with normal data processing.

* Thousands of faults were physically inserted into the processor(s) to evaluate the diagnostic program.

Finally, individual tests are skipped or the diagnostic is terminated within a phase when certain critical tests within a phase fail.

5.1.7.7 *Some Results.* The techniques discussed previously produced a considerable reduction in the number of faults which resulted in inconsistencies when diagnosed repeatedly. Previous comparable diagnostics have resulted in approximately one of twelve faults producing differing results during repeated fault simulation. In contrast, only one out of sixty faults fell in this category during fault simulation with the SPC processor diagnostic program. The improvement in consistency can be attributed to differences in hardware design (which uses a "bit-oriented" hardware layout), recording techniques, abort rules, and program structures.

5.1.8 *Store Diagnostic Program and Exercise Program*

The store diagnostic program attempts to locate a fault within a store which has previously been found faulty. The tests are performed using both processors when they are in service. Most of the testing is performed using special maintenance orders provided for this purpose. Using these orders, only the store on the bus specified by the order responds. The route (i.e., bus 0 or bus 1) for these maintenance orders can be specified by a control flip-flop within the processors. Since stores are dedicated to a particular bus, selectivity is achieved. In addition, a pair of special maintenance orders were designed with associated special store circuitry to test the bus leads associated with the store complexes. These two orders provide complementary mode control bits to the store, directing it to transfer its input directly to its output. Thus, a minimal amount of store control is required to test the store interface with the bus system.

The store tests are divided into related groups called "phases" which test various portions of the store circuits. A special store monitor bus is used for observing internal dc points; a store "snapshot" register is available to sample internal functions in a manner similar to a processor match circuit when it is being used in a sampling match mode. Control of this "snapshot" register is provided by special maintenance instructions called control reads and control writes.

If the all-tests-pass diagnostic of the store occurred after a faulty circuit pack was replaced, certain sections of memory may have become outdated (i.e., they do not contain recent changes in variable data) before the power was restored to the frame. Therefore, before a store is returned to the working system after an all-tests-pass diagnostic result, steps must be taken to ensure that the store memory contents are current. The contents of the diagnosed store should be the same, or be made the same, as the contents of the mate store on the other store bus.

If the all-tests-pass diagnostic occurred as a result of a scheduled, automatic exercise, no rewriting should be necessary. The store diagnostic program keeps stores updated while diagnostic tests are being run.

The checking and updating of a store which has passed diagnosis is performed in two millisecond segments, with call-processing interrupts inhibited. Each segment, whether checking or writing, deals with eight 40-bit words. The checking includes two parts: comparing 40 bits of each word with the corresponding word in the mate store, and a check of a processor buffer bus flip-flop bit. The flip-flop can be set by a parity failure, a double bit error, an address error, or a read or write all seems well failure.

If one or more 40-bit mismatches occur during a checking segment, the entire eight-word block will be rewritten from the mate in the following segment. The eight-word block is then rechecked in the next two millisecond segment to see that the writing was performed properly. As stated previously, the buffer bus flip-flop is checked only if 40-bit mismatches have not occurred in the current checking segment. If the flip-flop indicates an error, the eight-word block will be rewritten and rechecked in the following two segments.

Stores are automatically diagnosed and updated once a day to insure that maintenance circuits are excercised and that residual single errors in memory are not allowed to accumulate.

5.1.9 *Peripheral Unit System Diagnostic and Exercise Programs*

The SPC System provides diagnostic programs and exercise programs for all its major peripheral units. The exercise routines keep records on manually induced equipment states and initiate routine diagnostic testing of their respective frames. The diagnostic programs provide testing in response to fault recognition requests and manual teletypewriter requests. The five SPC peripheral units with diagnostics are: the master signal distributor, the master scanner, the central pulse distributors, teletypewriters, and the program tape unit.

The teletypewriter and program tape unit diagnostics do not conform to the scheme described above. The teletypewriter diagnostic can be requested manually and is also run as a scheduled exercise. This is because there is not a fault recognition program for the teletypewriter.

The program tape unit diagnostic is only run in response to a manual request; no fault recognition or routine exercise is available. Since both of these frames are under manual supervision, this scheme does not present any problems.

Although the basic functions of the diagnostics and exercises are those mentioned above, certain special features do exist. These factors provide the flexibility necessary to allow the system to be used for many applications. The scanner and signal distributor diagnostics are specially designed to allow use of many of their program segments in diagnostics provided for similar application peripherals. An application system with frames similar to the scanner or signal distributor can eliminate a great deal of additional program by utilizing these features. The diagnostics use tables and transfers to special application sub-programs to achieve this flexibility.

The central pulse distributor diagnostic also provides transfers to application sub-programs to provide tests of central pulse distributor features that are dependent on frame assignments. This means that individual applications are not restricted to any special scheme in assigning central pulse distributor points for their equipment.

5.1.10 *Off-Line Testing*

The off-line control programs provide testing tools to assist the maintenance operator. They are generally used to locate the cause of hardware problems which cannot be readily detected by the diagnostic programs (due to inconsistent results, intermittent faults, etc.).

Off-line testing is limited to processors, stores, and store buses. No special provision is made for off-line testing of peripheral equipment since test facilities are available at the master control center and independent use of the peripheral buses would pose major problems.

5.1.10.1 *Off-Line Functional Tests.* Off-line functional tests fall into two basic categories: those which are repetitively executed, and those which are performed in a single operation. Repetitive tests include:

(*i*) Execute a specified instruction or group of instructions at a fixed repetitive rate.

(*ii*) Write a specified 40-bit word repetitively into one or more consecutive unprotected store locations.

(*iii*) Write a specified 20-bit word repetitively into one or more consecutive unprotected store locations.

(*iv*) Read 40-bit word(s) repetitively from one or more store locations.

(*v*) Read 20-bit word(s) repetitively from one or more store locations.

One-shot tests include:

(*i*) One-shot initialization of a processor buffer bus register.
(*ii*) One-shot initialization of an unprotected store location.

5.1.10.2 *Non-Off-Line Functional Tests.* Non-off-line functional tests provide a means of printing data on the maintenance teletypewriter concerning the state of the active system upon execution of any specified program instruction. This data is useful in analyzing hardware troubles or verifying program changes.

To obtain the data printout, the maintenance operator sets the switches on a plug-in auxiliary matcher unit (which will generate a G-level interrupt on a match condition) to the octal address of a desired program instruction. He then activates this monitor function by typing in an appropriate input message.

Activation of the plug-in unit will not affect normal system operation or matching. The first time the program instruction specified on the plug-in unit is executed, a G-level interrupt takes place. The G-level program then enters a dump routine which produces a teletypewriter printout of all active processor buffer bus registers. In addition, any 12 memory locations specified in the input teletypewriter message will be dumped.

The plug-in unit is deactivated and no further dumps will be made unless requested by the maintenance operator. System operation will continue with no effective change of state.

5.1.10.3 *Method of Performing Off-Line Repetitive Exercise.* Repetitive exercises give the maintenance operator the ability to execute any specific program instruction(s) available in the off-line store(s) or specified via the teletypewriter while observing the results on an oscilloscope or making other operational checks. The instruction(s) are executed at a repetitive rate of approximately once every 50 milliseconds. An oscilloscope may be synchronized on the start pulse of the standby processor or any signal directly related to the start pulse.

Briefly, the sequence of performing repetitive exercises is accomplished in the following manner. The input teletypewriter message either specifies the address of the first instruction to be exercised or the program internally generates this start address. The same conditions apply for the last instruction to be executed (end address). If only one instruction is to be executed repetitively, the start address and end address are identical

The end address plus four (pseudo end address) is inserted into the match register physically located in the active processor. The matchers are set up in the directed mode in a manner which causes the standby processor to stop after completion of the instruction specified as the original end address.

When entered by the J-level interrupt control program, the off-line program starts a time-out counter in the active processor and inserts the stored starting address into the standby processor program address register. The match circuitry is activated to stop the standby processor on a match of the pseudo end address. The standby processor is then started. While the standby processor is executing the instruction(s), the active processor continuously monitors the "standby-processor-stopped" flip-flop.

As soon as the standby processor stops, the active processor switches buses so that it can write into both sets of stores in order that the standby store system can be kept up-to-date during call processing. It then returns to the main program for normal call processing. At the next scheduled entry from the J-level control program (nominally 50-millisecond intervals), the entire procedure is repeated.

In order to defend against the standby processor's inability to reach the pseudo end address and stop, the active processor counts the number of cycles elapsed since the program was entered by the J-level control program. Each time the standby processor does not stop within a fixed period of time (approximately one millisecond), a time-out counter is incremented. The standby is forced to stop, and the normal off-line routine described above is repeated at the next J-level entry. The time-out counter is interrogated at each J-level interrupt entry into the off-line control program. If the number of time-outs exceeds a programmed limit of 100, the off-line-control program informs the maintenance operator of the time-outs, clears all off-line indicators, and restores the system to normal. A request is made to diagnose and update the off-line store frame, and the standby processor is left out of service.

5.1.10.4 *Limiting Write Access to a Single Off-Line Store Frame.* The off-line processor is restricted to sending and receiving on a specified off-line bus. There is also a further restriction on write operations by the off-line processor. Write instructions are completely inhibited by the processor unless specifically requested by the maintenance operator. When requesting an off-line system configuration, the operator must request write access to a specified off-line store frame.

The store name code is placed into the off-line processor down-store name match register. The off-line write access flip-flop is then set. Normal down-store name matching in the active processor is not affected. However, if the off-line processor attempts to write into a store, the store name code of the store being accessed is automatically compared with the store name code in the down-store name register. If the store name codes are not identical, an E-level interrupt is generated and the off-line testing is terminated. This defensive action is necessary to protect against the mutilation of processing information.

The above restrictions have no effect on normal processing, since the active processor sends on both buses during all time intervals between actual off-line program activity.

5.2 *Input-Output Control Programs*

The SPC system provides a number of basic input-output control functions for the use of application systems. There are four major input-output control functions provided by the SPC system. These are: program tape unit read and write control; teletypewriter message translation and control; control and display panel input control; and POB execution for buffered control of various units. The design of these programs embodies the flexibility needed to provide service for a wide variety of application devices. This flexibility is achieved through the use of structured tables and provision for application sub-programs in appropriate areas.

The teletypewriter control program provides control and structuring of all system output messages. It provides the necessary features for a wide variety of output message formats. It also provides a variety of input message translations and numerous mechanisms for passing control and data to programs responsible for reacting to the various input messages. The program is designed with provisions for variable size application message translation tables in addition to the SPC translation tables. This feature allows an SPC application system to rely on teletypewriter control program for all teletypewriter control and message construction.

The program tape unit control program provides three major functions, viz., bootstrap recovery, system loading, and system copying for SPC and application systems. A bootstrap program is provided to reload the system programs when operation has been curtailed by severe mutilation of protected memory. There are two copies of this program on each store bus to insure availability for system recovery. This enables the bootstrap to function after almost any system failure. The bootstrap program can be initiated from the control and display panel when the system is not operational by inserting a special card into the processor to cause transfer of control to the bootstrap on a manually induced A-level interrupt.

When the bootstrap program receives control, it determines a workable combination of master scanner, central pulse distributor and peripheral address bus for communication between the processor and the tape unit. It then begins the system reload by unlocking all frames on both store buses and writing the data obtained from tape into duplicated memory. The store unload logic is written in such a way as to minimize the total size of the program. Also, the program must be self-sufficient so that it does not depend on any parameter data to obtain store or program tape unit enables. In this way, it can reload any SPC system regardless of the degree of mutilation in other programs. To some degree this requires a blind approach to the memory writing operation. For example, all store frames are unlocked before each 40-bit word is written. However, since the program must wait for the slow-speed tape unit (5 inches/second), this inefficiency is acceptable and permits an office independent design.

The program tape unit control program provides two major functions during normal system operation: system reloading for installation of generic program changes; and system copying to provide permanent tape records of protected memory. The program to provide these facilities operates in J-level and base level. The J-level entry provides accurate timing for communication with the program tape unit. The base level program assembles data for the J-level program. In this way, a minimum of J-level occupancy is required.

The craftsman can use the program tape unit program in a variety of ways. Copies can be made of any desired sections of memory. Also, there are automatic features to provide dumping or loading of all office data areas and all program areas. These areas are defined by application tables and may be easily redefined for any application.

The A-level interrupt control program provides filtering of emergency requests made by the craftsman using the control and display. When a system reconfiguration request is initiated from the control and display, the control program insures that proper software status is established for the new configuration. The program also responds to manual requests for call processing recovery phases and passes control to the system initialization and recovery programs. Additionally, there are several minor functions provided. The primary one is reinitialization of the teletype software to allow the craftsman to gain control of the teletypewriter when it is not functioning properly. In addition, for the use of application systems, several spare keys are also provided. Modification of a table used by the A-level control program enables these keys.

The last input-output control program residing in the SPC generic system is the peripheral-order-buffer execution program. For the SPC, this program provides queuing of data destined for the signal distributor. In application systems, it can provide similar services for any peripheral unit with a low data rate. In addition, the program also provides specialized services for fast units. In this way, specified order sequences can be executed by the program and rigidly defined timing between events will be automatically provided by the input-output program. For the basic SPC system, orders for the scanner and the central pulse distributor may be initiated by the peripheral-order execution program in any combination with signal distributor orders and delay orders.

The SPC implementation of the peripheral-order execution program provides a great deal of flexibility for adapting to application needs. All order sequences in the buffer consist of a control code followed by one or more data items. Each control code defines a unique subroutine designed to provide a particular service. These subroutines are accessed through a transfer vector table thus allowing any application system to tailor the program to its needs by merely providing a special subroutine for each new device to be serviced.

5.3 *Maintenance Administration*

5.3.1 *Maintenance Control*

The base-level executive control program enters the maintenance control program through the lowest priority base-level class (Class E). At this time, maintenance control searches for requested work according to a priority hierarchy. Deferred fault recognition work is higher than diagnostics, which in turn are higher than exercises. In honoring new requests, maintenance control insures that the highest priority request is performed first. Within the broad priority categories, the search for work requests is made by hardware unit type. These unit types are arranged in the status tables such that maintenance for the more critical units is performed first.

When maintenance control finds a job request, the requested client program is started and the job is carried out on a segmented basis. Segmenting of the maintenance program is necessary to prevent interference with the processing of normal system tasks. Each client segment is approximately ten milliseconds long after which maintenance control returns control to the base level executive control. When maintenance control is reentered in Class E, it initiates the client's next segment.

Maintenance control administers, coordinates, and schedules maintenance client programs through the maintenance control register, a common temporary scratch pad and control block. The scratch pad area is used by the client to record test data. Since the scratch pad is used by almost every client, maintenance control performs housekeeping functions to avoid conflicts. Maintenance control also provides periodic status reports on the teletypewriter regarding all units in the system.

The maintenance alarm scan is a subprogram of maintenance control that is entered in Class C of the base level every two seconds. It provides periodic scheduling for maintenance control clients such as audits and automatic exercises. It also provides general purpose timing for maintenance control as well as the maintenance control clients and performs administrative functions such as controlling lamps on the equipment frames.

5.3.2 *Maintenance Interrupt Return*

The Maintenance Restart and Restore Program provides a common return mechanism for maintenance interrupts. It runs any necessary clean-up jobs and determines where the system should be restarted.

The restart program provides several entry points for programs returning from maintenance interrupts. Each entry provides special functions appropriate to the returns it receives and then enters a common clean-up thread. The clean-up thread is a general purpose task dispenser. It executes a number of subprograms that check the integrity of data structures after an interrupt. Programs that cannot be reentered after maintenance interrupts provide abort routines to be run by the clean-up task dispenser. For example, the peripheral order buffer execution program provides a subprogram to insure that it will not be reentered if a maintenance interrupt occurs while it has control. Similarly, all programs administered by the maintenance control program are aborted when a maintenance interrupt occurs. Finally, one of the subprograms tests the hardware and software interrupt counts to determine if they are excessive.

The final action performed is program restart. The point of return is established, the registers are restored if appropriate, and control is returned to a lower level. Selection of the point and level to which the return is made involves several factors. Certain maintenance interrupts or combinations of maintenance interrupts must never return to the interrupted point. Instead, a reference point return is made. Reference points are provided in the executive control program, and they will restart processing in a natural fashion. When a reference point return is not required, execution will be reinitiated at the point and level where the current interrupt took place. In some situations this may also be a maintenance interrupt level. One other factor is considered in determining the restart point. If a software recovery phase has been requested or was in progress at the time of the interrupt, control will be passed to the beginning of the software recovery control program in the emergency action program. A new software recovery phase will then be initiated. After the return point has been established, the registers are restored if a direct return is to be made, and control is passed to the appropriate program.

The restart program provided with the SPC is easily modified for use by any application program. The subprograms mentioned above can be augmented to provide data structure validation and interrupt sensitivity testing for all programs unique to a given application.

5.3.3 *Emergency Action*

The emergency action program provides monitoring of system normalcy and control of recovery from abnormal operation. When the system is incapable of normal operation due to a major hardware failure, the B-level portion of the emergency action program is entered. This program operates in conjunction with the hardware emergency action sequencer to provide an operational hardware configuration. The hardware provides a series of initial processor and store bus configurations, and the program tests these configurations. The processor is tested by the processor sanity test program under control of the emergency action program. All processor functions except peripheral communications are tested. A failure in one of these tests activates the hardware sequencer which will then select a new configuration. When a working processor has been found, the central pulse distributors are tested. A series of tests are run to verify the basic addressing and functional mechanisms of the central pulse distributor. For each central pulse distributor pair, the best central pulse distributor is selected as the active unit. The final action in hardware emergency action recovery is selection of an operational store system. This is accomplished through the store bootstrap program. This program makes a series of read and write tests to select an acceptable store system. If a complete system is not available, the hardware emergency action sequencer is started to select another configuration.

The second major function of the emergency action program is to insure the sanity of the software system. This is done by a number of tests which originate in a high-priority J-level program which is entered every 100 milliseconds. One of the functions of this program is to administer a hardware long timer. Failure to perform these actions will cause an emergency action (B-level) interrupt. In addition, the program makes several tests for software insanity of a less obvious nature, and it initiates system overload actions. Five basic tests are performed. The systems' interject response is monitored to detect loops in base level programs. The time between Class E base level visits is monitored to determine if the system is in an overload state. J-level activity is monitored to insure that no J-level program is in a loop. The base level job visitation schedule is monitored to guard against various forms of base-level processing insanity.

Hardware and software caused maintenance interrupts are also monitored. Software-caused maintenance interrupts are considered to be a very strong indication of mutilated data or residual program errors. Hardware interrupts are not considered to be a conclusive indication that the system has generated and used invalid data. Therefore, a small number of software-caused interrupts are considered to indicate a serious loss of normal processing, whereas the count for hardware interrupts must be relatively high to reach the same conclusion. For all of the above tests, certain failure thresholds are set. If these thresholds are exceeded, a call processing recovery phase will be initiated.

The emergency action recovery program provides control of call processing recovery phases. The control program initializes hardware and makes requests for all appropriate software routines to audit and correct the variable areas of memory. It also monitors the sanity of the programs operating during the phase. If a phase fails to recover the system, the control program will restart the action and run the audit routines appropriate to the next higher phase. The recovery control program is table-driven and has the flexibility necessary for use in any application system. With appropriate audit routines, a complete call processing recovery phase system can easily be developed for any application.

VI. SUMMARY

The SPC No. 1A has been designed as an economical and reliable central processor which can be easily applied to varied real-time control and logical processing functions. This article has attempted to give an overview of the SPC hardware and software design while placing emphasis on the more novel features of the system. Thus, particular attention has been given to certain aspects of the order structure, the development of standard interfaces for the application systems, duplication and switching arrangements, and special program structures. It has been assumed that the reader has some familiarity with the stored program switching systems of the Bell Telephone System such as the ESS No. 1 system.[1,14] Thus, areas of similarity which have already been well documented in connection with these systems were described only briefly herein for completeness.

The common system design of the SPC proved to be of great value in the adaptation of the second application system, the electronic translator (ETS). A number of installations of both the TSPS and ETS systems are now in service and the SPC has proven to be dependable and easy to maintain.

REFERENCES

1. Keister, W., Ketchledge, R. W., and Vaughan, H. E., "No. 1 ESS: System Organization and Objectives," B.S.T.J., *43*, No. 5, Part 1 (September 1964), pp. 1831–1845.
2. Jaeger, R. J. Jr., and Joel, A. E. Jr., "TSPS No. 1: System Organization and Objectives," B.S.T.J., this issue, pp. 2417–2443.
3. Baker, W. A., Kinder, G. W., Myers, F. H., and Culp, C. A., "TSPS No. 1: Stored Program Control No. 1A Store," B.S.T.J., this issue, pp. 2509–2560.
4. Freimanis, L., Guercio, A. M., and May, H. F., "No. 1 ESS Scanner, Signal Distributor and Central Pulse Distributor," B.S.T.J., *43*, No. 5, Part 2 (September 1964), pp. 2205–2283.
5. Comella, W. K., Day, C. M., Jr., and Hackett, J. A., "TSPS No. 1: Peripheral Circuits," B.S.T.J., this issue, pp. 2561–2623.
6. Baldinger, W. R., and Clark, G. T., "TSPS No. 1: Physical Design," B.S.T.J., this issue, pp. 2685–2709.
7. Harr, J. A., Hoover, Mrs. E. S., and Smith, R. B., "Organization of the No. 1 ESS Stored Program," B.S.T.J., *43*, No. 5, Part 1 (September 1964), pp. 1923–1960.
8. Cagle, W. B., Menne, R. S., Skinner, R. S., Staehler, R. E., and Underwood, M. D., "No. 1 ESS Logic Circuits and Their Application to the Design of the Central Control," B.S.T.J., *43*, No. 5, Part 1 (September 1964), pp. 2055–2097.
9. Harr, J. A., Taylor, F. F., and Ulrich, W., "Organization of No. 1 ESS Central Processor," B.S.T.J., *43*, No. 5, Part 1 (September 1964), pp. 1845–1922.
10. "No. 1 ESS," B.S.T.J., *43*, No. 5 (September 1964), pp. 1831–2609.
11. Dougherty, H. J., Raag, H., Ridinger, P. G., and Stockert, A. A., "No. 1 ESS Master Control Center," B.S.T.J., *43*, No. 5, Part 2 (September 1964), pp. 2283–2319.
12. Downing, R. W., Nowak, J. S., and Tuomenoksa, L. S., "No. 1 ESS Maintenance Plan," B.S.T.J., *43*, No. 5, Part 1 (September 1964), pp. 1961–2020.
13. Beuscher, H. J., Fessler, G. E., Huffman, D. W., Kennedy, P. J., and Nussbaum, E., "Administration and Maintenance Plan, "B.S.T.J., *48*, No. 8 (October 1969), pp. 2765–2816.
14. Quinn, T. M., Toy, W. N., and Yates, J. E., "Control Unit System," B.S.T.J., *48*, No. 8 (October 1969), pp. 2619–2668.