A real time executive system for manned spaceflight

by J. L. JOHNSTONE
International Business Machines Corporation
Houston, Texas

INTRODUCTION

The Real Time Executive Control System discussed in this paper was the foundation for the applications programs developed in support of NASA's Gemini and early Apollo missions. Services provided by the Executive included dynamic storage management and allocation, two-level priority multiprogramming, real time data control and routing, real time error recovery, dynamic statistical monitoring, debugging facilities, and the program linkages and services that facilitated modular and independent applications system design. While a selection of these services may be available in other systems, the Executive design differs from other real time systems by these characteristics:

- Modularity—The Executive design permitted the addition of new services and facilities based on equipment changes or applications requirements with no impact on the previously provided services and facilities.
- Simplicity—Only a minimal instruction in Executive services was necessary before applications programmers could construct programs that operated in a complicated real time environment.
- Versatility—Executive could be used in the simplest simulated real time environment for the debugging of one applications program or the support of the most demanding real time missions.
- Generality—Executive was non-applications oriented; i.e., it operated equally well in a real time Gemini mission, an astronaut training session, or in a non-real time environment using simulated input from tapes.
- Invulnerable—The Executive was virtually unstoppable in real time; a feature vital for manned spaceflight.

The executive environment

The RTCC

A brief introduction to the Real Time Computer Complex (RTCC) is necessary before proceeding to any discussion of the Executive Control System. The RTCC is a functional part of the Mission Control Center at NASA's Manned Spacecraft Center in Houston, The RTCC's missions during spaceflights or training sessions are to:

- take spacecraft tracking and status data being received from NASA's global communication network and process it for display to flight controllers stationed in the Mission Control Room and the computer complex;
- compute and then forward antennae-aiming directions to tracking and communications networks all over the world so they can begin to track the manned spacecraft as it approaches;
- send calculated navigation and other information to the computer aboard the spacecraft; and
- simulate the data that network sites and space vehicles would generate during an actual mission so that personnel can be trained and equipment can be checked and readied.

To perform these missions, each of five IBM 7094-II's was assigned a different role, and the RTCC was engineered so that these roles could be exchanged at any moment. This unified set of computers allowed NASA to run either two practice missions at the same time, or a practice mission and an actual mission at the same time. Figure 1 gives a dramatic demonstration of the five systems at work in the latter configuration. In the mission configuration, network data flows into the RTCC from one of the communications processors at the Manned Spacecraft Center and is sent to the Mission Operational Computer and the Dynamic Standby Computer by a switching device called the System Selector Unit. In the simulation and training exercise, a nearby Gemini spacecraft trainer is in a closed loop system with one of the two identical Mission Operational Control Rooms (MOCR). The other MOCR is being used for the mission. One simulation computer contains a system which is generating simulated network data; the other computer is used as an operational computer. The

From the collection of the Computer History Museum (www.computerhistory.org)

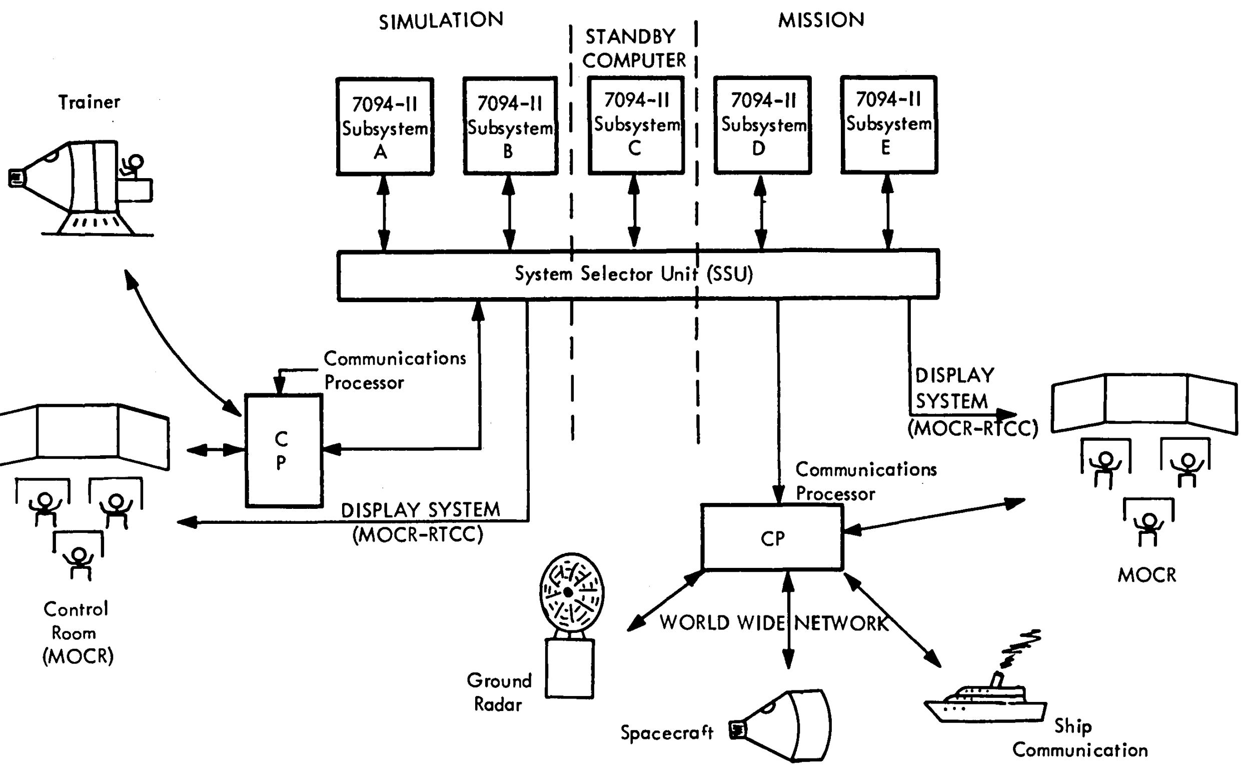

Figure 1 – RTCC data flow-simultaneous simulation and mission

fifth computer is a standby computer for both exercises; however, it is not idle but is processing a single-computer debugging exercise for checkout of a future mission or simulation system.

Although all three of the functions being performed in the RTCC are different, a single Executive is performing the control system functions for each.

The computer system

Each of the IBM 7094-II computer systems (Figure 2) in the RTCC has 65K primary memory, directly addressable through automatic relocation hardware. Each system has 524K words of Large Capacity Storage (IBM 2361) which is used as extremely high-speed buffer storage for programs and data. Programs are buffered between main memory and the Large Core Storage (sometimes termed "core file" or "COFIL") and placed in main memory wherever space is dynamically allocatable. A protect feature permits areas of storage to be protected from illegal storing operations. Tape drives are attached to standard data channels A and B. In addition, a card reader and printer are attached to channel A (not shown). The Direct Data Connection (IBM 7286) on channel C provides a rapid demand-response interface to the digital display (D/TV) television system. Access to large storage areas at a high data rate is provided by the use of the Large Capacity Storage on channel D. Real time acceptance and transmission of large amounts of data and control information are accomplished through the use of the IBM 7281-II Data Communications Channel (DCC) on channel F. At the RTCC, the DCC has 13 subchannels designated either input or output. Both the Direct Data Connection and Data communications Channel interface with the equipment and data networks serviced by the RTCC via the System Selector Unit (Figures 1 and 2).

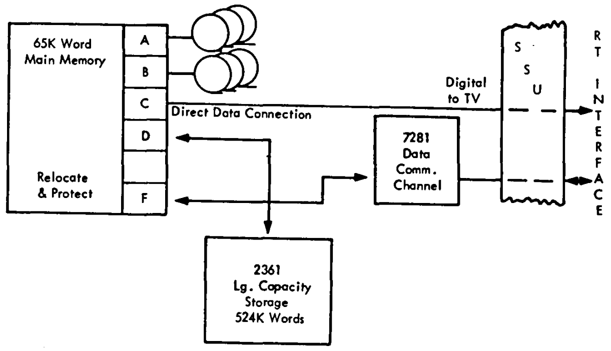

Figure 2 – RTCC 7094-II computer system

From the collection of the Computer History Museum (www.computerhistory.org)

As was shown in Figure 1, the SSU permits the individual computer subsystems to be configured either singly, or in combination, to perform any of the various mission, simulation, program testing, or equipment testing functions of the RTCC. It is designed to process the inputs and outputs of up to six RTCC computers. The System Selector Unit makes switching connections between computers and their inputs and outputs and routes data accordingly. The SSU's routing assignments are made by plugboards.

The preceding paragraphs have discussed the Executive environment—the RTCC and its IBM 7094-II computer in which the Executive performed the real time computer control functions from Gemini IV to XII and Apollo 201, 202, and 203.

The real time system design

What is executive?

We have placed Executive in an environment geared to real time operation. But, to enter into any discussion of real time, one must first define *his* version of real time; for there are probably as many definitions of real time as there are real time systems. To understand the Executive real time system design, one must realize that the response time for the NASA mission application must be an increment sufficiently small to guarantee positive control of a manned spaceflight. At the RTCC, the usual time frame (or increment) in which data is received and presented to NASA Flight Controllers is considerably *less* than a second. Appreciating the response time required for the real time Executive, we can now turn to a general description of Executive.

General description

Executive is a collective term for those routines which perform the support functions for the applications programs at the RTCC. Executive has two general responsibilities in this capacity: (1) to serve as an interface between applications programs and the RTCC input/output devices and communications lines, (2) to control the execution of and communications between the application programs. Executive is a non-applications oriented system; i.e., it supports equally well all the RTCC systems: the Gemini or Apollo Mission system, the Simulation Checkout and Training System (SCATS), the Dynamic Network Data Generation system (DNDG), the Ground Support Simulation Computer system (GSSC), or the Operational Readiness and Confidence Testing system (ORACT). The application programmers who design programs for these various Executive supported systems code routines in assembly language or FORTRAN to perform mathematical computations, interpret input data, or form output data. The programmers are relatively uninformed as to how Executive works internally in performing its responsibilities. All that is required of the programmer to use Executive is a basic knowledge of the communication mechanisms with Executive and what he is to expect in the way of input from Executive.

System modularity

The majority of the application system's situations call for processing logic programs which can be segmented into controlling logic and a series of controlled processing elements. The former programs are designated supervisors, and the latter are termed processors (see Figure 3). Supervisors are multi-element programs which control processors and treat

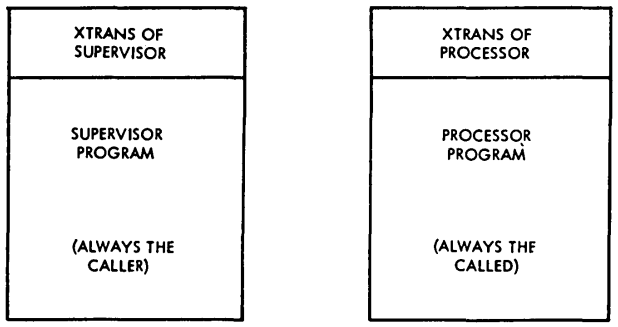

Figure 3—Supervisor and processor

them as closed subroutines. Processors differ from traditional subroutines in that no processor can call another processor. A processor can only execute and return control to a supervisor; usually, the supervisor that called it. Some processors are general in nature, as in the case of certain mathematical operations. Processors of this type can be shared by supervisors. Processors receive no input data from Executive. Supervisors receive and supply the processors with the data needed for the processor's execution.

Supervisors and processors are relocatable units; i.e., they are dynamically buffered from static storage on the Large Core Storage (LCS) to core by Executive when their logic is needed. The origin of a supervisor or processor in the 7094 core is at an address which is a multiple of 256. This address, termed base register, is set by Executive into the relocation register prior to execution of the processor or supervisor. Although every supervisor and processor is assembled with addresses relative to zero, the base address (contents of relocation register) and the offset (gen-

From the collection of the Computer History Museum (www.computerhistory.org)

erated by the program at assembly time) are summed in the hardware for specification of actual memory addresses. Address protection is also performed in the hardware; the upper and lower bounds of a supervisor or processor in core are set into the protect registers by Executive when the element is brought into core. If the processor or supervisor references a memory address outside these bounds, a protection interrupt occurs. Certain "protected" instructions also cause an interrupt.

Although the Executive is primarily a core resident monitor existing in the lower 13K of the 65K core memory on the 7094, it too has about a dozen functional programs in the form of supervisors and processors that it buffers in and out of core as needed. As a comparison, the Gemini Mission System contains about twenty supervisors for centralization of flight/vehicle control logic and for supervision of data processing and mathematical computation. Nearly 250 processors are callable by the supervisors.

Standard argument area

Every supervisor or processor contains a ten-word table, called XTRANS (see Figure 3 and Figure 4.1). When any program requires processing by another program, the "calling" program fills its own XTRANS with whatever data the "called" program needs to interpret the request.

For example, in a program used to compute square roots, an XTRANS convention would be established by the program. All other programs requiring square roots would follow the convention. This convention could be: when the square root program receives control, it will calculate the square root of the quantity contained in the first cell of XTRANS. This square root program also would specify that the third word of XTRANS will always be set to zero (Figure 4.2), unless some error occurs in the square root calculations (Figure 4.3). This simple example could be complicated slightly by changing the program to a generalized root extractor. In this case, the root extractor program might define the first word of XTRANS to contain the argument, the second word to contain the power, the third to return an error code or zero, and the fourth to contain the absolute answer.[1]

Once a supervisor or processor (program) defines its input and output XTRANS, that program's services are available to any programs requiring them. When one program (usually a supervisor) calls another program (usually a processor), Executive moves the contents of the caller's XTRANS to the XTRANS of the called program. When the called program completes and returns control to Executive, Executive moves the contents of the completed program's XTRANS back into the XTRANS of the caller, as shown in Figure 5.

Standard control interface

The Executive provides a standard interface which is used to pass control between application programs (supervisors or processors). By using this interface, the Executive solves such problems as: allocating a program to main memory prior to execution, executing programs according to their priority in the system, and multiprogramming the asynchronous flow of many paths of logic. (See Multiprogramming Aspects below.)

The responsibility of determining how the Executive should pass control from program to program rests with the programmer by use of the CALL statement. The CALL statement requests a service from the Executive, while the arguments dictate how the service should be performed.

The mechanism of a CALL statement is to enter a specialized Executive routine in the resident nucleus, provide the routine with arguments supplied in the CALL statement, and have Executive execute according to the definition of the service and the supplied arguments.

To reach the resident Executive routine to perform the service requested by the CALL statement, a subroutine, which was attached to the supervisor or processor (element) at assembly time, is first entered. (Each Executive service has its own subroutine.) This subroutine simply places a certain code in a Store-and-Trap (STR) instruction, and then executes the instruction. The executing of the STR causes an interrupt (trap) in the 7094. The Executive fields the trap, interprets the code, and transfers control to the specialized routine designated by the code.

Multiprogramming aspects

As noted in the Introduction, the Executive is a multiprogramming system; i.e., it permits many independent paths of logic to proceed asynchronously and is able to switch control of the CPU (Central Processing Unit) from one path to another, depending on the priority of a supervisor or processor and its availability for a particular path. The priority of the supervisors and processors is determined by the order of its entry in the Executive priority table. This entry not only establishes the element's priority but reflects the general status of the element at all times by giving the following indications:

- Is currently operating or idle.

From the collection of the Computer History Museum (www.computerhistory.org)

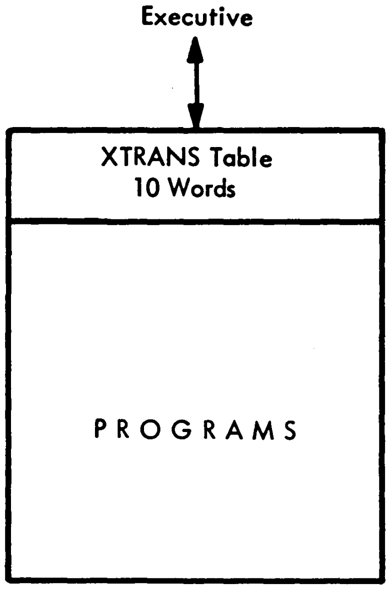

Figure 4.1 Standard Argument Area: XTRANS

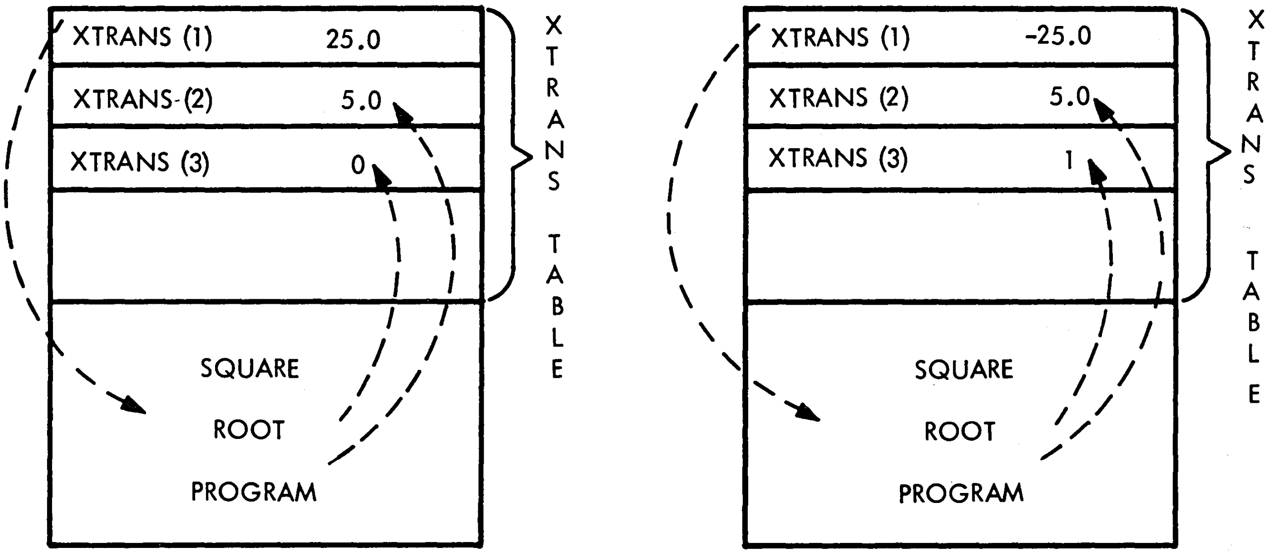

Figure 4.1 – Standard argument areas: XTRANS
Figure 4.2-4.3 – Square root program

- Has one or more XTRANS waiting in a queue to be sent to another supervisor or processor.
- Is being loaded into core.
- Is in core at location XXXXX, or is not in core.
- Is on LCS (program must be loaded from tape to LCS to core for execution; programs generally loaded into LCS from tape once per many core loads).
- Is a supervisor or processor.
- Is privileged (runs with Executive ignoring protect interrupts).
- Is suppressed from running.

From the collection of the Computer History Museum (www.computerhistory.org)

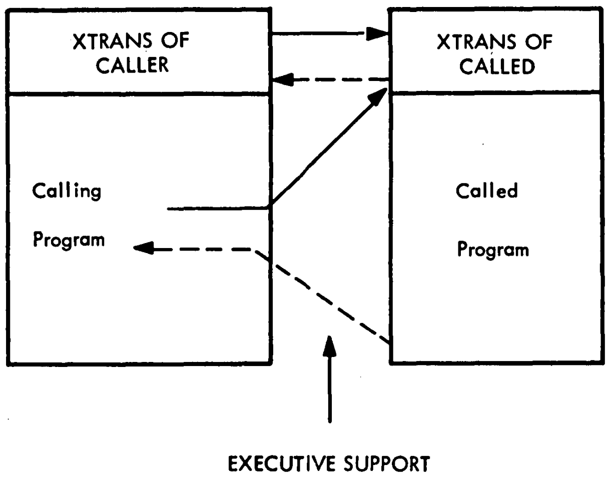

Figure 5 – Executive and XTRANS

The order of the entries in the priority table is established by the applications programmers through a macro at nucleus assembly time. During èxecution, Executive scans the priority table from the top each time a status change occurs. When Executive finds a program ready for execution, the scan stops, and that program is given control.

A supervisor further inhances multiprogramming in that it consists of one or more programming elements, called functions. Each function has its own XTRANS area and may operate independently; in addition, all functions share a single copy of a permanent data area kept for each supervisor in a special buffer called XTPERM. (See Figure 6.1.) XTPERM is permanent since the Executive preserves the contents of the table when the main core storage occupied by a supervisor must be made available for other uses. When the supervisor again receives control, the supervisor's XTPERM is exactly as the supervisor last left it. Processors have no XTPERM but many have temporary work space while executing. The size of XTPERM is established by the programmer to fit his needs for permanent data. There are probably no two supervisor XTPERM's the same size in the RTCC systems.

The second level of the two-level multiprogramming structure discussed in the Introduction of the paper is found in the functions of supervisors. Functions have an internal priority that determines which function is to receive control when two or more functions of a supervisor compete for control.

The function's design is based on the concept that a small package of functions (a supervisor) could effectively generate a number of parallel logic paths, and that multiprogramming will occur almost without

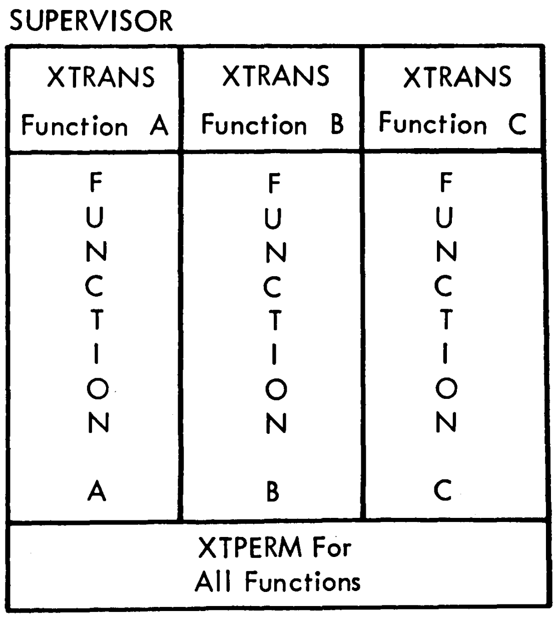

Figure 6.1 – Three-function supervisor

the programmer being aware of it. With a number of more-or-less independent logic paths operating asynchronously, the Executive can maximize the effective utilization of the CPU.

The supervisor function can call a processor several different ways. The classical method is to call a processor as a subroutine (see Figure 6.2). When the processor completes its task, it returns control to the function at the next instruction after the call. The function is out of operation, so to speak, until the processor completes.

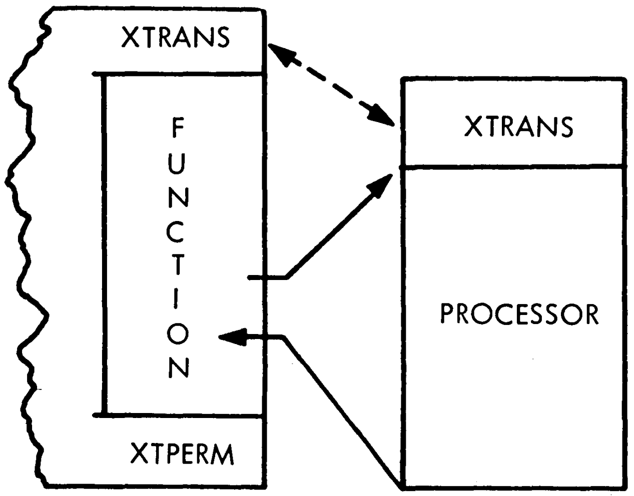

Figure 6.2 – Function of supervisor calling processor

From the collection of the Computer History Museum (www.computerhistory.org)

Anotner approach permits the supervisor function to "send" a call to the processor so that the function does not give up control. Consequently, for calls that are sent, a function may call a number of processor (see Figure 6.3). In this method, a function may initiate a number of parallel operations. Furthermore, sending calls provides another control option. The supervisor function, in sending a call, can permit the processor to determine whether a return is to be made or not.

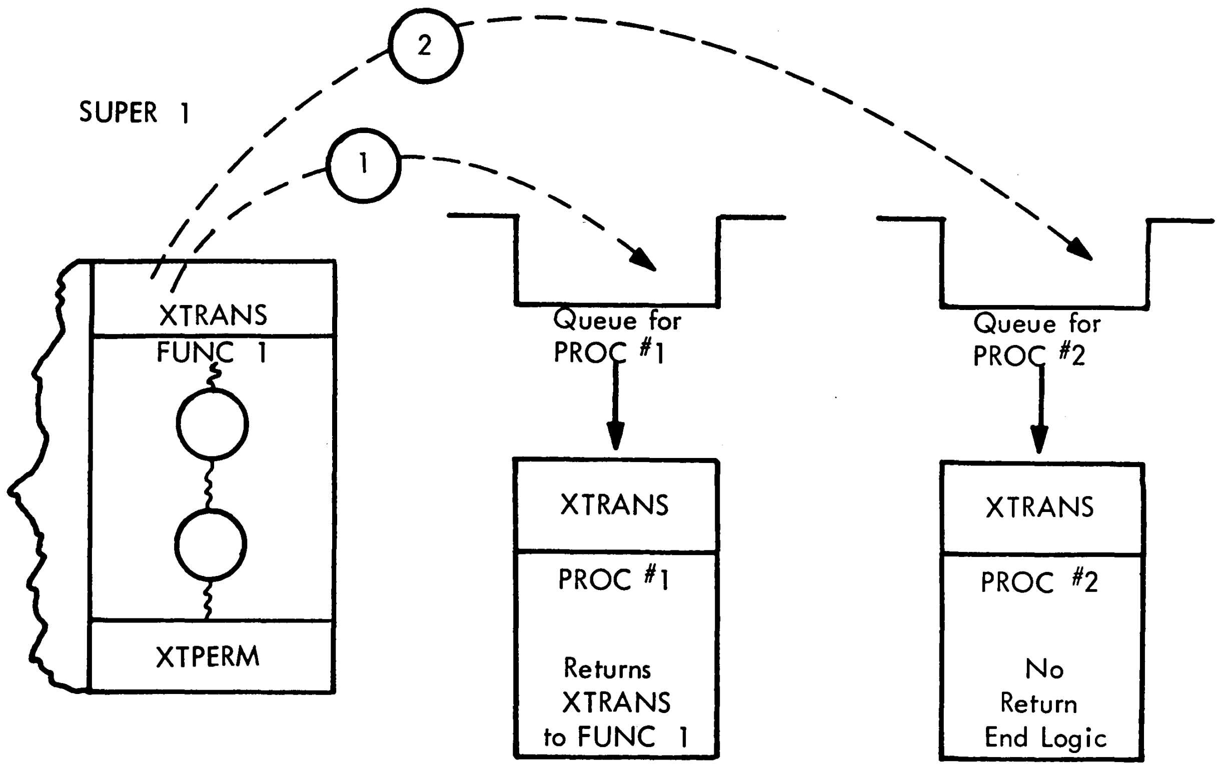

Figure 6.3 — Calls sent to processors

If no return is to be made, the processor represents an "orphan" task in multiprogramming that simply executes and completes the task entirely. This is a fairly typical operation where processors update D/TV displays and no return to a supervisor function is expected, that is, unless something unusual is uncovered in the processor's execution. If the processor returns, it must return to the start of the function specified by the arguments in a calling sequence of the original calling function. The function returned to may even be a function of a different supervisor. Therefore, the processor is effecting a transfer of control without knowing where this control is going.

Finally, functions of the same or different supervisors communicate by calls that transfer the XTRANS of the calling functions into the queue for the called function. If the calling function has the higher priority, control remains with that calling function.

Real time processing

Basically, we have placed the Executive in lower 65K core and stated that it performs allocation of supervisors and processors into main core storage from the LCS (more on allocation later in the paper) when a requirement for the supervisor and processor is known.

We have shown the supervisor (with its functions) and the processor giving request to the Executive for certain services. Now we turn to the major requirement for a supervisor or processor to be brought into operation. That requirement is the receipt of real time data.

Real time data receipt

In Figure 7 we find a processor in operation when a data channel trap (or interrupt) is received from the 7281 DCC (channel F). What has happened is that

From the collection of the Computer History Museum (www.computerhistory.org)

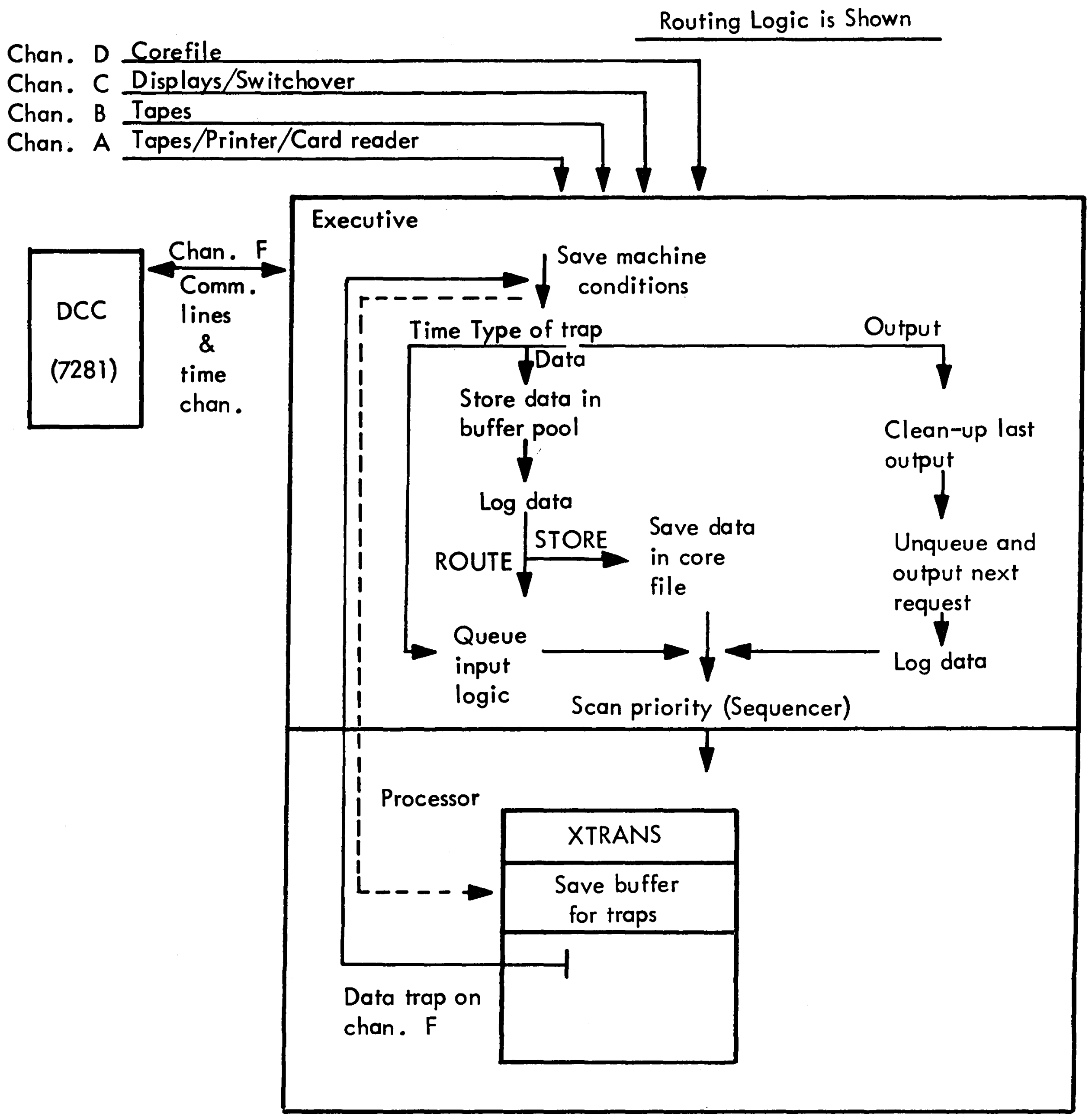

Figure 7—Trap control logic

data is now being placed in a buffer in lower core by the hardware. (Each of the DCC input subchannels has an addressed buffer in lower core.) The Executive receives control from the processor when the interrupt occurs and saves those registers and addresses that will permit return to the processor without change to the conditions existing before the interrupt occurred. Executive takes the data that has been placed in the input subchannel buffer and places it in a main core buffer pool for it to be logged onto an output tape. The other process that Executive performs on the data is termed routing and is covered in the next section. Now that Executive has received control via the data channel interrupt, it has an opportunity to scan the priority table to find the highest priority element (supervisor or processor) with a work queue waiting for it. The element found in the scan is then entered.

From the collection of the Computer History Museum (www.computerhistory.org)

There are, of course, other Data Channel Interrupts, as shown in Figure 7, for channels A, B, C, and D.

Routing

The RTCC computers must interface with other computer installations and man/machine devices. Additionally, the RTCC computers must keep in step with Greenwich Mean Time (GMT) so results will be maintained in real time and will be synchronized with computing efforts elsewhere. The Executive routing feature manages the input from the communications lines (DCC) and routes data and time (GMT) to the proper functions and processors.

The routing logic is part of Executive; however, Executive makes no original decisions as to the destination of input. Routing information (directives) for time and data is supplied to Executive by the application programs. This information is stored in routing tables associated with DCC input subchannels. The user must activate and deactivate the directives by chaining.

When data arrives in the system, the data identification (ID) is compared to all possible ID's of data that might arrive over that subchannel. When the data ID matches an ID stored in a chained routing table, the routing table information is used to queue the data to an input function or processor, to store the data until a future time, or to discard the data because they are not needed.

When time arrives in the system, the current time (GMT) is compared to all routing tables which are used to direct queues depending on time. Routing will generate a queue to all functions and processors for which the request for time has been met.

Time signal routing

One type of routing is time routing. For instance, suppose a supervisor has to produce display output every second. The supervisor would inform the Executive that the supervisor requires control every second. The Executive would file this request in a routing directive for future reference. (See Figure 8.)

Every second thereafter, the Executive would notice, while scanning its time routing directives, the name of this supervisor listed as requiring a call every second. The Executive would create an XTRANS for the supervisor and would, in effect, call its type; i.e., time, and the current time. The type code would distinguish this particular XTRANS from any other types of XTRANS the supervisor may receive. (It should be noted that supervisors may call upon this supervisor with other type codes in XTRANS.)

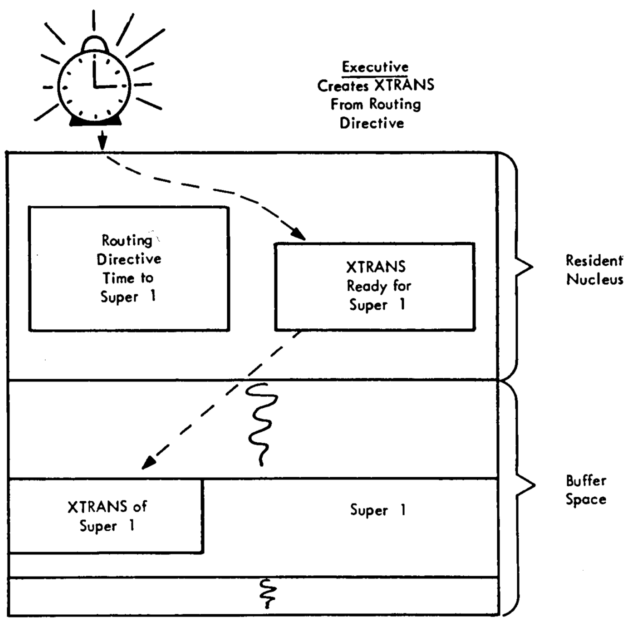
Executive Creates XTRANS From Routing Directive
Routing Directive Time to Super 1
XTRANS Ready for Super 1
Resident Nucleus
XTRANS of Super 1
Super 1
Buffer Space

Figure 8 – Routing of time

It is particularly important to note that the supervisor need tell Executive only once that repetitive calls are required each second. The Executive will generate these calls every second, indefinitely, until the routing directive is modified or cancelled.

After the XTRANS is created, the Executive would attempt to give this XTRANS to the supervisor so the supervisor can begin processing. Frequently, a supervisor or processor cannot immediately receive the latest XTRANS because:

a. The supervisor or processor is not in main core.
b. The supervisor or processor is busy doing something else.
c. More important work, i.e., some other supervisor or processor has to be done first.

These problems are avoided, or at least deferred, by inserting the XTRANS into a queue for the supervisor involved. Every XTRANS, given to or created by Executive, enters a queue for the program that is to process the request. The queue is ordered chronologically; the earliest request is always at the top of the queue. When the program involved is available in core and has the highest priority, the request leaves the queue immediately. Otherwise, requests wait in the queue for their turn. Each time a program completes processing of one request, the program is available for the next request in the queue (see Figure 9).

Real time data routing

The Real Time routing in Executive brings real time data to the application programs. The data routing

From the collection of the Computer History Museum (www.computerhistory.org)

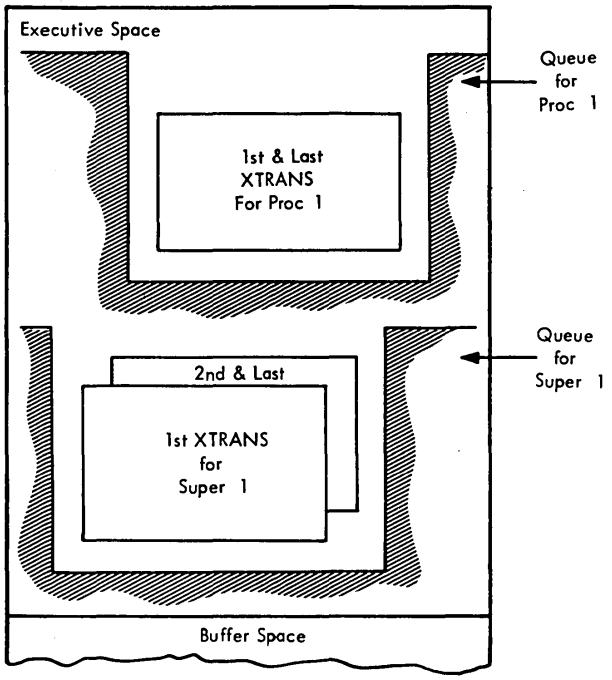

Figure 9—Queues of XTRANS

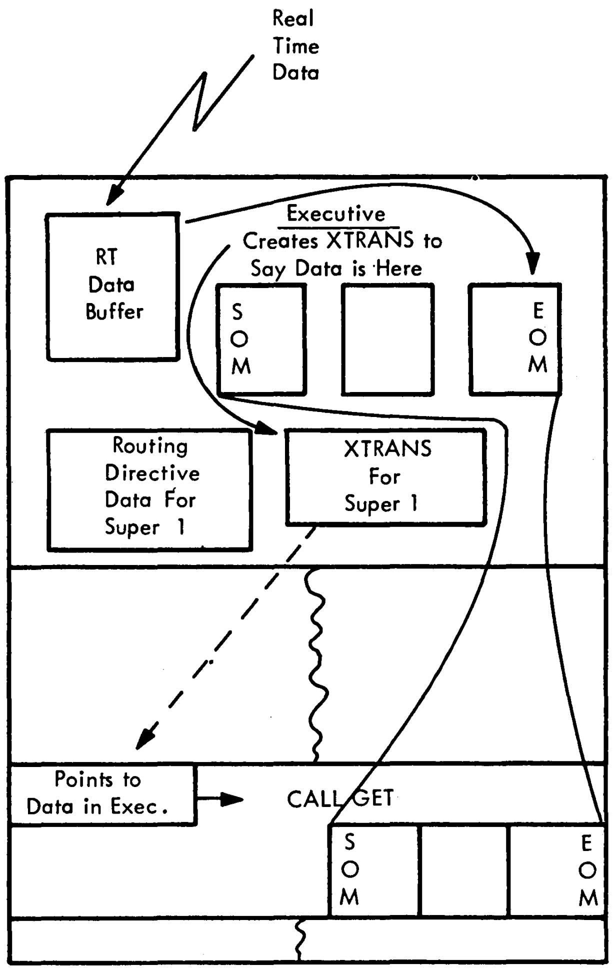

Figure 10—Direct mode data routing

process is similar to the routing of timing signals. The application program specifies a routing directive for data. This routing directive is composed of two parts: the first gives Executive the criteria to identify the particular data the program requires, and the second part of the routing directive tells Executive what to do with the data that satisfy these criteria.

There are basically two options for applications programmers in routed data; direct routing, and store-mode routine. (See Figures 10 and 11.)

Direct mode routing

RTCC has few variable length messages. Messages of a given type generally have a constant size. Some types of real time data messages are small enough to fit within an XTRANS. For these messages, the programmer can specify direct mode routing. When a data channel interrupt occurs, Executive simply creates an XTRANS table, places the data into the XTRANS, and places the XTRANS into the queue for the program named in the routing directive. When the program receives the XTRANS, the data are then ready for processing.

If the data are too large for the XTRANS, the Executive places the data into a buffer in lower core and places information in the XTRANS that describes the location of the data in the buffer. Using the information provided in the XTRANS, the program obtains the message by executing a CALL statement to request the Executive Real Time Input/Output Control System (RTIOCS) to move the message into the program's area.

The direct mode of routing (as demonstrated in Figure 10) is generally most effective for the small, non-repetitive real time data inputs.

Store mode routing

For cyclic real time data applications, processing usually consists of two phases: data collection, and data processing (see Figure 11). The data collection process can be processed entirely by the Executive. The programmer defines a routing directive that instructs the Executive to store the selected data into a data table on the LCS (data tables are termed Z-tables). The programmer also creates, separately, a routing directive that causes Executive to generate a periodic XTRANS as a function of time. When the

From the collection of the Computer History Museum (www.computerhistory.org)

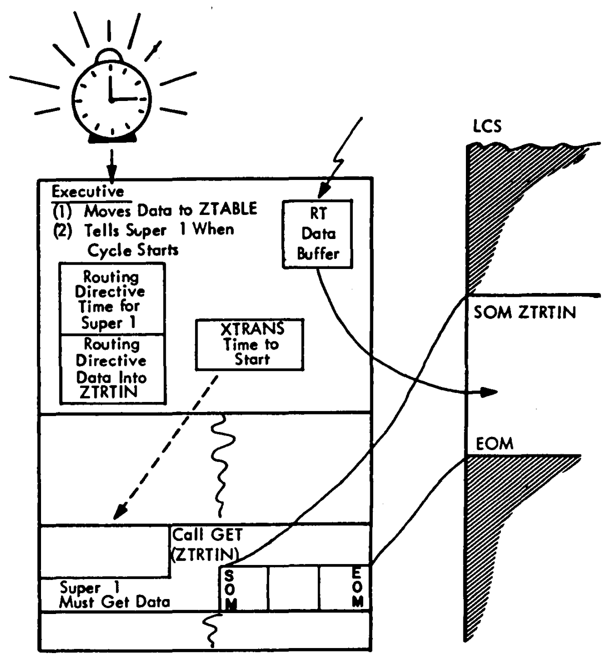

Figure 11—Store mode routing

program receives this XTRANS, it requests RTIOCS to read the collected real time data from the Z-table into a designated area of the program for processing.

The major elements of executive

We have placed Executive in the Real Time system and shown to a limited extent how it provides the support and control necessary to sequence the execution of the various supervisors, functions, and processors through their request and through the receipt of real time data. We have spoken of Executive in general terms and definitions; now, we can turn to a brief description of some of the major elements that give Executive its structure.

Executive linkage

Executive linkage with supervisor/processors is effected through library routines appended to supervisors/processors at assembly time as the result of CALL statements within the supervisors/processors and associated service routines within Executive. As has been shown previously, when an Executive capability is called by a supervisor/processor, the appropriate library routine in the supervisor/processor sets up and executes an STR instruction that contains a numerical code identifying the type of call. Execution of the STR instruction causes control to be passed to Executive under hardware control. Executive references the code in the STR instruction to determine the type of call and then passes control to the associated Executive service routine.

Executive sequencer

The Executive Sequencer consists of those routines within Executive which service the real time system by:

a. Interpreting Central Processing Unit (CPU) traps which occur from the execution of an STR instruction within a relocated element.
b. Servicing floating point and protect CPU traps.
c. Assigning control to the highest priority element which has work outstanding.
d. Saving and restoring machine conditions when a data channel trap or CPU trap occurs.
e. Interpreting requests for transferring control between the elements of the system.

Real time input/output control system

The purpose of the Real Time Input/Output Control System (RTIOCS) within the Executive is to provide a simple (from the programmer's standpoint), flexible communication link between the supervisors and processors and the various input/output media available in the RTCC (no "Raw I/O" is allowed outside the RTIOCS). The RTIOCS consist of a series of integrated routines which perform all necessary input and output functions to the following devices and storage media employed within the RTCC: Tapes (Channels A and B), 7286-II 512K core file (Channel D), 7281-II Data Communications Channel (DCC, Channel F) which consist of 13 subchannels connected to such devices as plotters on the output side and real time data receivers on the input side, the Digital/TV System (Channel C), 65K primary main core memory, printer and card reader.

The basic framework of the RTIOCS, from the user's standpoint, is a statement CALL I*GETT or I*PUTT (* = "R" if the call is made by a processor and "U" if made by a supervisor) and a series of three to five arguments showing the action to be taken. A typical call might be: CALL IRGETT (ZXAMPL, MYBUFR, NBRWDS, LOCINB, BLKNUM). This translates to: from data file ZXAMPL, starting at LOCINB (a symbol containing an integer) in BLKNUM (a symbol specifying a block number if this is batch data) transfer NBRWDS (a symbol which contains the number of words) into MYBUFR (a symbol for a buffer area, normally within the calling processor). The data file name termed Z-table name (in this example, ZXAMPL) is a symbolic name of a four-word table (File Control Block) in Executive which defines the data file (its location, its type, its

From the collection of the Computer History Museum (www.computerhistory.org)

size, and other information pertinent to the particular device that is the data file). Data files may be on tape, LCS, main core, etc.

Figure 13 gives an example of RTIOCS servicing a request for I/O from a user. The same type CALL service logic using the STR instruction that was discussed earlier in this paper is used for I/O request.

DCC servicer

The DCC Servicer processes all 7281-II DCC subchannel traps. These traps may be caused by both the input and output subchannels. The DCC Servicer moves the data from the cells in lower core in which the hardware placed the input data into the Executive Buffer Pool, sets up information for the Executive logging routines to log the data, and sends a request to Routing to route the data.

Dynamic main memory and auxiliary storage allocation

It was expected that the Gemini systems would change from mission to mission and would grow to exceed the capacity of the computer main memory. Since this change and growth could not be contained, the necessary flexibility was built into Executive to permit such change and growth. Part of the flexibility is in the design of the storage allocation routines of Executive. No permanent storage location is assigned to any problem program. Storage is allocated on demand and in the quantity necessary to accommodate the particular program.

Two distinct levels of storage allocation are used. The first, allocation of main memory, is essential to every run. It provides for the allocation of areas of memory to required relocatable supervisors and processors and uses the Executive RTIOCS capability to load the programs into memory from the LCS (see Figure 12).

The second, LCS allocation from magnetic tape, is necessary only when there are more relocatable programs than can be accommodated concurrently in the LCS. It provides for the allocation of areas in the LCS to relocatable programs (processors only) according to both actual and user-anticipated requirements. It also supervises the transmission of the programs from magnetic tape to the LCS.

When a request for a supervisor or processor is made, the Executive Sequencer determines if it is in main memory. If it is not, Sequencer makes an explicit request for the program by transferring to the Executive allocation routines. If the program is available in the LCS, the main memory allocation routine attempts to allocate memory for it. If the program is not in the LCS, the LCS allocation program is queued to bring the needed program from magnetic tape to the LCS, after which memory allocation will be reattempted.

Once main memory has been allocated for a program, the Real Time Input/Output Control System (RTIOCS) reads the program from LCS into main memory. Control is then returned to Sequencer. When no main memory can be allocated for a program due to relative priority, activity status, and length considerations, the request for allocation is retained so that it may be re-attempted later.

LCS allocation is initiated in response to actual requirements for programs to execute in main memory or in response to user calls that specify which programs will be placed and held in the LCS in anticipation of actual requirements.

Initializing the real time system

The final element in a discussion of the Executive is Initialization. Executive initialization provides user options for the Executive nucleus to execute in several modes and in various hardware configurations. Initialization occurs prior to entering Executive and prior to starting a real time or simulated real time operation. The initialization options permit distinctions between real time or simulated time, real input data or simulated data, and actual I/O devices or simulated replacements. The general scope of initialization includes:

- Establishing initial hardware conditions
- Establishing parameters and initial conditions for storage allocation
- Ensuring proper linkages between certain real time programs
- Assigning tape drives
- Loading initial data into data files on the LCS (Z-tables)
- Creating the Executive buffer pool
- Establishing debug request tables.

Prior to entering into the discussion of Initialization, it is essential that we give the two steps the user must take before his supervisor or processor is in an application system that is being initialized. The first step is to accomplish unit testing of his supervisor/processor and the second is to create the application system tape.

Job shop simulator

Since all the Executive services are provided via the CALL statement, a simulator of the real time environment was easily provided under the IBM 7094 IBJOB system. This capability permits testing of a

From the collection of the Computer History Museum (www.computerhistory.org)

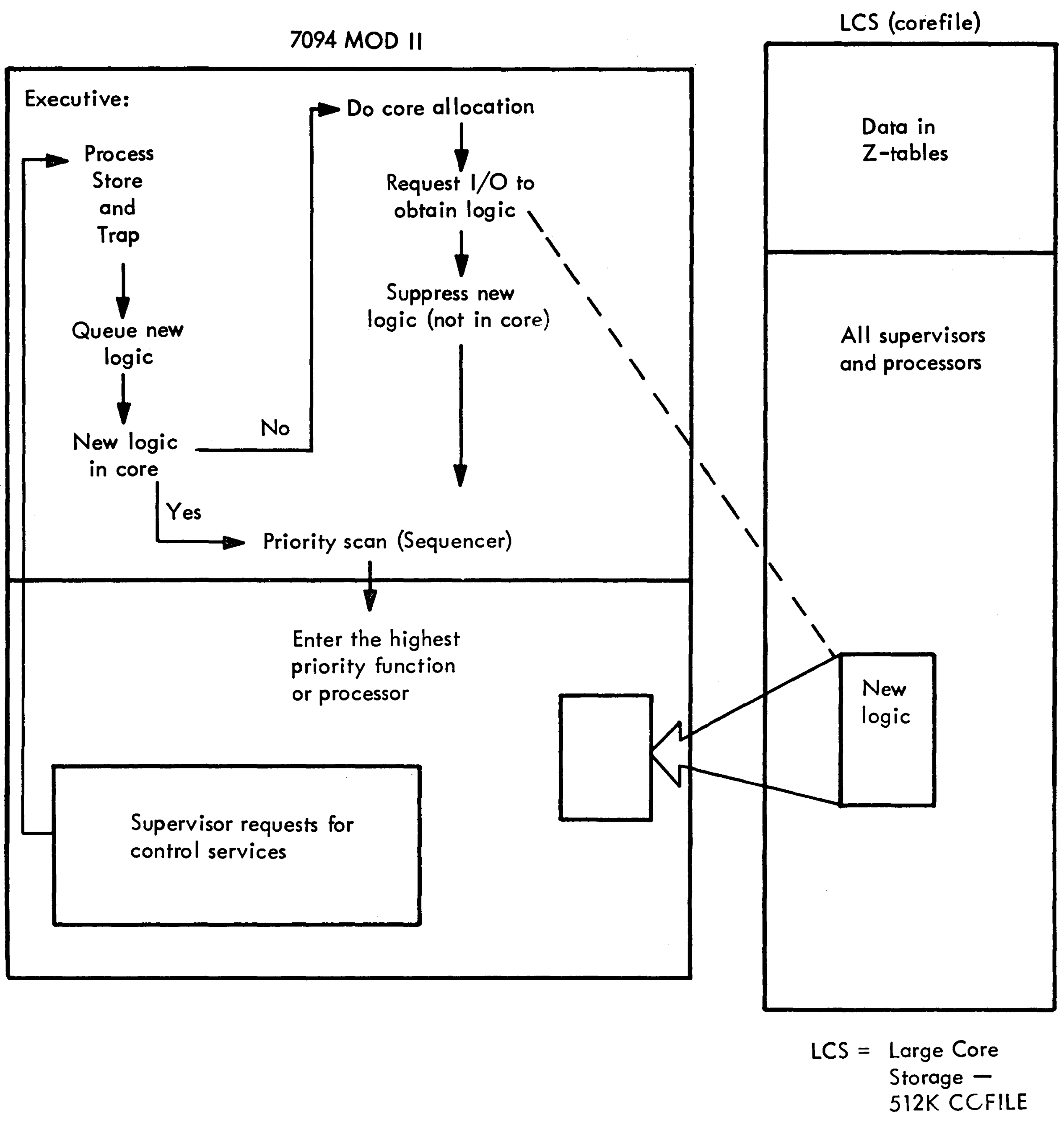
7094 MOD II
LCS (corefile)
Executive:
Process Store and Trap
Queue new logic
New logic in core
No
Yes
Do core allocation
Request I/O to obtain logic
Suppress new logic (not in core)
Priority scan (Sequencer)
Enter the highest priority function or processor
Supervisor requests for control services
Data in Z-tables
All supervisors and processors
New logic
LCS = Large Core Storage — 512K CCFILE

Figure 12—Program control flow

single processor or supervisor, or a supervisor and several processors, in a batched-job system. Some of the more exotic features of Executive's multiprogramming facility cannot be simulated adequately in a sequential (essentially IBJOB) environment. But for most unit, string, or subsystem testing, the Job Shop Simulator is very effective. The fact that processing is sequential often makes it possible to identify bugs before the environment changes completely (a constant problem in multiprogramming debugging). In addition, the capability of conducting significant debugging in a batch environment greatly economizes the computer time required to deliver checked out systems.

Creating the real time systems tape

When unit testing has been completed with the Job Shop Simulator, the programs can move unchanged

From the collection of the Computer History Museum (www.computerhistory.org)

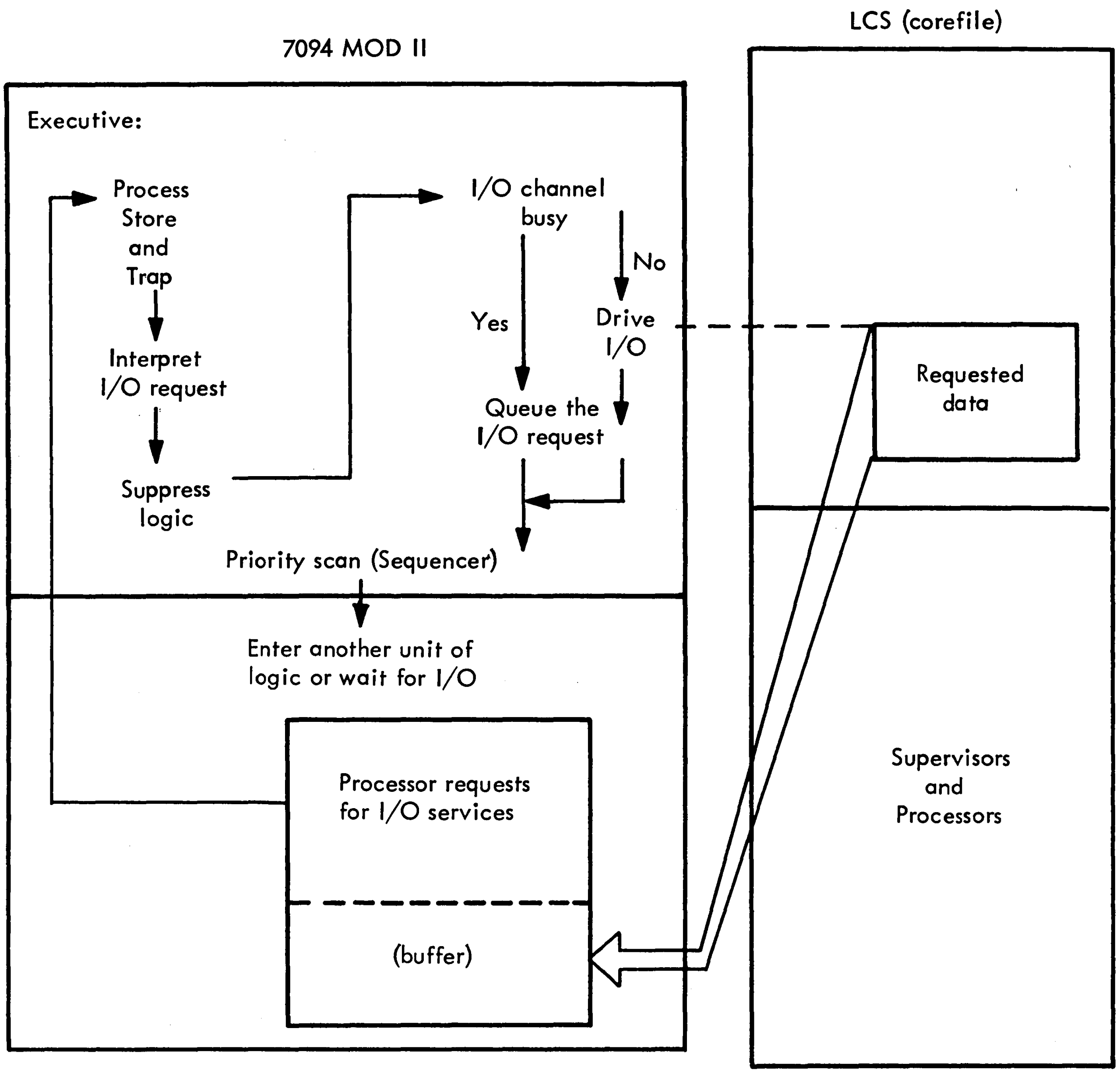

Figure 13 – Real time IOCS

into the real time system testing environment. The first step is to create a real time system tape.

The Executive has been described as a single-computer program serving many real time applications systems. The user must define his application system to the Executive by building tables inside the Executive. This is accomplished by macro statements that the user inserts into a special Executive card deck. When this deck is assembled, the user's section of Executive is created. These macros basically define:

a. The supervisor/processors of the application system and their relative priority (the priority table)
b. The file control blocks (for Z-tables)
c. The initial routing directives.

During the building of the system tape, the user's decks are combined with the Executive code to produce the Executive nucleus. In the process, all of the symbolic names (program names, Z-table names, etc.) are translated into indexes. The translation

From the collection of the Computer History Museum (www.computerhistory.org)

process simply trades pre-execution time to save translation in real time. The nucleus Executive is written on the real time system tape along with copies of the remainder of the non-resident Executive and all of the user's application system programs.

Real or simulated time

The first significant Initialization option is whether or not the user requires an internal/external clock synchronization. Unless external devices (including other computers and/or people) are involved, synchronized time is rarely used. Unsynchronized or simulated time simply uses an internal clock that never runs when the computer is idle. When idle time occurs, this clock is spaced forward to the next clock interrupt. For example, this feature permits an orbit of 90 minutes to be completed in about 10 minutes of elapsed time. No changes are required in any applications system program. In fact, there is no way an application program can tell that a simulated clock is being used.

When running in the simulated mode, the user can specify that any or all of the real time input devices (subchannels) are to be simulated with canned data from tape or, in some cases, the card reader. Any of the real time input devices can be simulated while the remainder accept actual data, or are unused, in any combination. Furthermore, the real time output devices can be used or the outputs can be diverted to the LCS by requesting on-line for Initialization to modify the file control block.

Debug request

The RTCC version of the IBJOB debugging system that permits core snapshot dumps, heavily used in job shop runs, is also available when running a real time system in the simulation mode. Since the debugging package operates with the simulated clock turned off, the applications programs cannot recognize that the debugging operations are taking place.

When a real time run is specified, initialization automatically removes any debug requests.

Error halt or error recovery mode

The normal inclination of the real time Executive is to continue processing, regardless of any errors that may occur. Some error recovery action is instituted in hope that the condition was only transitory. This is the sensible approach to real time support. But in debugging a system, especially a highly dynamic multiprogramming system, evidence should be saved as soon as the error is discovered. Consequently, another initialization option permits the system to run under either the error halt or the error recovery mode. Once this system is started in the error halt mode, the mode may be changed to error recovery and back, at the setting of a sense switch. Once the system is started in the error recovery mode, the mode may not be reset.

Real time statistics

Another initialization option permits the user to accumulate statistics during a real time run. The accumulation of statistics operates only in the synchronized real time mode and generally requires about five percent of the CPU in overhead. (The RTCC experience has been that the five percent of the CPU is not the difference between success and failure in a real time run.)

Once the initialization option has been set, the Statistics Gathering System (SGS) may be activated or deactivated dynamically from the Manual Entry Device (MED) or by the card reader used to simulate the MEDs. Statistics are accumulated in three categories:

a. Internal Executive Logic—frequency of use, average execution time, core allocation attempts and successes, etc.
b. Supervisor and Processor—number of uses, average execution time, number of uses per time loaded into core from the LCS, number of Executive CALL's, etc.
c. Total CPU Utilization—amount of time in execution, in waiting on I/O, and idle.

SGS, originally conceived and implemented to support the extensive GPSS (Gordon General Purpose System Simulator) modeling activities at RTCC, has proved useful to many of the applications programmers in analysis of their systems.

Operational features of executive

Real time run synopsis

When the run terminates, the user has the option of:

- A *synopsis* that consists of a formatted presentation of all the significant Executive tables: the state of the priority table, the state of all the processors and supervisors, the chains of XTRANS in the queues for all the processors and supervisors, etc.
- An octal dump with assembly language operational codes.
- A full symbolic dump.

If any debugging system snapshots were taken, these are formatted in the post-execution processing. Programmers generally take the synopsis and an octal dump.

From the collection of the Computer History Museum (www.computerhistory.org)

Real time internal control mode and generalized on-line display capability

The Real Time Internal Control Mode (RTICM) of Executive gives the user of Executive more control of the multitude of initialization and dynamic options, while giving the user the capability to exercise these options remotely. Most of the Executive options, prior to RTICM, required the user to set and reset the sense switches and keys of the 7094 console. RTICM permits all options to be punched into cards and allows selective dynamic use of these cards to be governed by the processing itself. The on-line display capability allows a user to select dynamically, via the Manual Entry Devices, information from main core on the LCS to be displayed on the television system. The information can be selected symbolically, saved, and recalled by name.

Reliability

During the Gemini and Apollo missions, the Executive is keyed to keeping the real time system up and running no matter what adverse conditions are encountered. In doing this, the Executive has provided an elaborate Error Recovery System to intercept program errors, hardware generated errors, and data generated errors. When one of these errors are encountered, Executive quickly examines the situation and produces an appropriate recovery method to enable the real time processing to continue. If the error encountered is of such a nature that recovery is either impractical or unfeasible, the Executive will recommend Switchover to a standby system. If necessary, the 65,000 words of core memory and the 524,000 words of COFIL memory can be transferred from this standby computer to a new operational computer by the Executive RESTART logic in less than five minutes. In the worst case, real time processing is never delayed more than three minutes. If an I/O device fails in any manner, Executive provides time-out logic to ensure that failure on one device will not interfere with the remainder of processing in the real time system. If tapes fail or become full, Executive provides tape switching logic.

CONCLUSION

Some of the basic ideas for the Executive were developed in the Real Time Mercury Monitor for NASA's Project Mercury and, in turn, some of the ideas conceived in the Executive design are being used in the development of the Real Time Operating System/360 for Project Apollo. It has been found in these endeavors that real time system development is an evolving creature, for the predominant requirement in its development is that its design must be able to evolve as the environment in which it will operate is understood.[2]

REFERENCES

1 J H MUELLER
The philosophy of the RTCC control programs
IBM Real Time Systems Seminar Proceedings 1966

2 R L HOFFMAN
Managing the design, development, and implementation of large scale generalized real time systems
IBM Real Time Systems Seminar Proceedings 1966

From the collection of the Computer History Museum (www.computerhistory.org)