OS/360 MODIFIED FOR REAL TIME MANNED SPACE FLIGHT SUPPORT

J. H. Mueller
Systems Design
1-5-65

The Real Time Computer Complex: The Real Time Computer Complex (RTCC) at Houston, Texas, is the NASA's ground support computing and data processing center for manned space flight. RTCC's primary responsibility is to provide real time information for mission control. But this role entails many additional supporting tasks: constant program system development; training of personnel for simulation exercises and flight support; providing and operating computer-driven diagnostics for RTCC and Mission Control Center equipment.

The RTCC Programming Systems: In general, each NASA mission, e.g., GT-6, GT-7, etc., requires three real time systems. The Mission Operational System (MOS) has approximately three quarters of a million words* of instructions and data, with a mission-to-mission carryover of about 30 per cent. The tracking network simulation system (DYNAMIC) has about a hundred thousand words. The Simulation Checkout and Training System (SCATS) has about two hundred thousand words. The two simulation systems have a carryover of about 65 per cent.

Several other real time systems are available for miscellaneous support functions.

All of the RTCC real time systems operate under a common Executive Control Program. However, each system requires a dedicated computer system.

Approximately one-half of the mathematics and related logic of the real time systems is represented by FORTRAN code. If more computer capacity were available, or if the compiler-induced penalty in object efficiency could be eliminated, more problem oriented code would be used.

*Missions become increasingly complex. The initial S/360 RTCC systems may be expected to double the sizes given here.

All of the program preparation and some of the program and subsystem testing is supported by a modified version of IBSYS/IBJOB. Individual programs (now called load modules) correspond to IBJOBs. These are collected by an extensively-modified IBLDR-Editor which produces the real time system tapes.

The RTCC 7094 Hardware Configuration: The current RTCC hardware configuration is illustrated functionally in Figure 1. The System Selector Unit (SSU) is a plug board switch box and provides the linkage between the real-time I/O devices and the 7094 systems. The SSU connections endow a computer system with a functional assignment. The SSU is transparent (non-programmable) to the computer system and, while some limited dynamic function exchange is possible, the SSU is essentially a static device. No condescension is intended. The SSU has served its intended function well.

Each of the five computing systems is an identical, independent 7094-II system. Each system has a modified relocate and protect feature that permits direct addressing throughout the 65K of primary memory. The 2361-Large Capacity Storage (LCS), provides a relatively large (524K words), fast (250K words/second) auxiliary storage accessible through a 7044-type data channel (7286). Output to the television display system is transmitted through the SSU by direct data on a 7286 channel. Other real time information is transmitted through the SSU over subchannels of the 7281 Data Communications Channel.

For mission support, the NASA reliability requirements are satisfied by fully duplexing systems: the mission computer has an immediate backup always available.

Some Considerations of the Current RTCC: The major problems of the current RTCC hardware are generally the limitations of the 7094 systems--no expandibility. The total system reflects these limitations. Within the multiprogramming environment, dynamic storage management and device allocation are, of course, provided, as is secondary storage allocation, but a true resource sharing a la multitasking/multijobbing was ignored in the preliminary RTCC design sessions of early 1963 for a number of reasons. Primarily, IBJOB could not support it. Ultimately, two software systems resulted with almost no interface at all. IBJOB was modified to produce a self-contained, self-loading real time system. Because early in the RTCC development the real time system was unavailable and because of the improved efficiency of an IBJOB environment for limited testing, an Executive simulator running under IBJOB was produced.

One of the more notable systems achievements was the development of the master libraries. IBLDR, primarily, and to some degree other components of IBJOB/IBSYS were modified to produce and handle both binary and symbolic master files. These permitted the programmer to construct real time tapes for execution by symbolic alters to cataloged source programs or by combining binary decks to produce any number of load modules. Today, some limitations are obvious--the lack of direct access storage and limited expandibility, but at that time the results gave a significant improvement over any previous methods for building and controlling large systems. (Incidentally, the need for a similar facility was apparently overlooked by the OS/360 designers though not by the OS/360 implementers who developed a nearly identical function as part of the ATP ? system.)

The RTCC S/360 Configuration: Figure 2 illustrates the RTCC configuration which will be operational in 1967. The rigid independence of each 7094 system has been softened by removing from the individual computers the control units and devices. Between the channels and the control units, the 2911 Remote Control Switches provide access to pooled devices. The 2911 switches significantly enhance reliability by permitting alternate paths to all real time critical devices; the 2911 switches eliminate plug board connections permitting automatic reconfiguration.

Each computer system is identical: Model 75J, with two-megabyte private LCS and one-megabyte LCS shared among all systems. Each system has a storage channel mainly for transmission between LCS and primary storage, a multiplexor channel with selector subchannels for tapes, data cells, unit record devices and real time transmissions, and a pair of selector subchannels for drums and disks. (Originally, no computer was to have private, non-pooled direct access devices. This may be impractical, at least for the near future.)

Each computer is equipped with a 32-bit CPU register for arbitrary interval timer interrupts with 10-microsecond precision. The timer requires no storage cycles to maintain time. The NASA time standard, GMT, can be read directly into core storage, also with 10-microsecond precision.

The computer systems may communicate via the direct control feature, by the shared LCS, and with reduced responsiveness via shared storage devices beyond the 2911 switches.

Clearly, RTCC has a computer facility that permits each computer to operate as an independent system by assigning the pooled devices. In the interim period between the first RTCC S/360 installation and sometime in 1967, Model 75s will coexist with 7094s as stand-alone computers, simple replacements for current computers. How the collective Model 75s will operate as multiprocessors in 1967 is currently in the planning stage. One fundamental consideration in these plans is the desire to retain sufficient compatibility with OS/360 as it evolves.

Commitment to OS/360 for RTCC: With the advent of preliminary OS/360 literature, RTCC began to evaluate the concepts and facilities of the future operating system. The initial survey concluded that OS/360, in comparison with the contemporary 7094 real time Executive, would increase overhead by an unknown factor. Most of the basic functions appeared to be there.

But, more important by far, whatever problems created by trying to live with OS/360 in real time, the alternatives were unthinkable. A separate, but equal, total operating system for RTCC functions could neither be justified nor funded. In addition, as real time support requirements (including the enormous number of simulations) keep growing, the inability to apply excess resources to program and system development would become ever more acute. The present dichotomy between the environments of the real time system generator (IBJOB) and the real time system should not be perpetuated through another technological generation.

The very specialized application at RTCC precludes complete coverage by any generalized operating system even one with the basic multiprogramming tools. The optimum compromise would preserve the maximum usefulness of the OS/360 control program at minimum cost. The end result of the RTCC effort would be an operating system which would provide all facilities of OS/360 to the normal, or default, user. However, for the real time user, standard OS/360 features would be augmented in some cases and restricted in others. One control program would support these two modes of operation concurrently, with the execution mode as an attribute of the job step.

Problems Peculiar to the RTCC Project: The most obvious area where local systems programming would be required is the real time input-output devices. No standard OS/360 access method is expected to drive the television system, the plotboard, or the other devices of the Mission Control Center. In fact, while some real time devices could be handled in normal data management fashion, most could not. In usage, most RTCC real time devices are handled less like standard data set volumes than the console typewriter in OS/360. Instead of one user who controls the data set, there are many users who send information to the devices.

*Some statistics from the 7094 mission support system may prove interesting: While launch phase processing has periods of extended 100% CPU utilization, the orbit phase uses approximately 25%. GT-7 combined less than 10 minutes of launch with two weeks of orbit.

A number of other functions are less obvious, perhaps but critical to the RTCC. These functions are provided by the current 7094 control program. Some of the functions are basically development tools with little meaning in real time support; they are, nonetheless, necessary:

- fail-soft facilities are necessary. These may require closing of input lines and imply a greater monitoring capability than that normally given to the control program. Practical tools must be provided for the problem program to define the direction selective work elimination should take.
- total failure (collapse) of the problem program system may be necessary for debugging, i.e., to find an elusive bug, but cannot be permitted in real time support. The control program must recognize which condition applies.
- a restart procedure must include all the elements needed for real time.
- a standard logging facility is necessary to record transmissions across the real time interface.
- the control program must recognize the switchover condition, in which the primary and backup computers interchange functions.
- specialized forms of time control are essential for efficient testing. These include adjusting time forward or backward and the ability to compress out idle time (a 90-minute orbit can with this feature be run in less than 20 minutes).
- simulated real time input and output device support must be provided to accomplish off line checkout of problem programs. Automatic handling of simulated data has been a mainstay in RTCC debugging: entire applications systems can be developed without recourse to any real time equipment other than a clock. Because this feature is part of the control program, all

applications systems can use the service. And more important, no change need be made to the applications programs when taken from the simulated data environment to real time support.

RTCC's Requirements and OS/360 Solutions: The nature of RTCC real time systems significantly differs from batch or job shop processing and from many other real time applications. Typically, real time data produces distinctly repetitious processing cycles in the program system. The duration of the cycle may vary from several hundred milliseconds to many seconds. Frequently, a number of processing cycles of varying duration exist concurrently. Superimposed upon, and continuously distorting the cyclic processing is the support for the button pusher, the NASA flight controllers (and if RTCC is anything, it is a man-machine communications system).

In this environment the geography of core storage is constantly changing. Also, in this environment, the price of finding and loading programs often measures the response of the system. The performance of core allocation, for example, must be measured not only in its utilization of core storage, but in the overhead extracted for the service. The effect of cyclic processing on system design will be shown later.

Because of the restrictions in interjob communications, each 7094 real time system would have to be translated into a single OS/360 jobstep. This preserves a logical integrity, but introduces some problems. Protection keys and rollin/rollout as jobstep-dependent functions would not suffice.

Within any OS/360 jobstep, the basic logic is reflected in a pyramidal hierarchy of subtasks. At the top of the pyramid is the primary subtask. This is the subtask initiated by the control program to start the job step. RTCC began building their real time systems on the 7094 with a type of pyramidal hierarchy of control. Experience soon led to the following conclusion: dependence--the main program with subroutine approach--is invaluable for relatively small functions. As size and complexity increase, the difficulties in communication and control increase geometrically with respect to the number of control levels. RTCC resolved this problem on the 7094 systems by expanding indefinitely the number of tasks which might share the highest control level (though not necessarily priority). The identical approach appeared feasible within (or almost within) the context of OS/360 design.

Modification 1. Independent Tasks: The logical conclusion was a new type of task--an independent task. A number of attributes were defined to distinguish an independent task from a conventional task. An independent has a name, which is the name of a task and not the name of a program initiating the task. An independent task may have subtasks, but is not a subtask of any other task. An independent task has a unique protect key, shared with any related subtasks. An independent task may return from its highest level to the control program without abandoning all acquired resources, notably subpool storage. This defines a new condition of a task, unique to independent tasks, the condition of being dormant. This is the normal state of an independent task which handles cycli~~d~~c processing when the cycle is temporarily idle.

The name of the independent task is in effect a serially reusable resource. Once a named task has initiated processing, that task may not again be initiated until it has relinquished control from its highest control level.

An independent task cannot be attached as are subtasks: an independent task is initiated as soon as priority permits after a work entry has been placed in the queue for the task. The dimensional characteristic of this queue provides a handle for introducing fail-soft techniques. For instance, an independent task processing cyclic input may permit no entries to be saved during a cycle. Backlogged work never chokes this task. Excess data is discarded. Certainly, not all data is discardable. But much is. And when a system is in danger of severe overload, much more is.

Finally, an independent task's priority is an attribute of the work queue entry. This priority need not be related to the priority of the entry's originator.

With priority, RTCC requires the facility to define and modify simply all of the priorities of a system. This is considered an analyst's problem and should be independent of the code using the priorities. A solution probably will result in symbolic priorities being used in the code and tables provided to translate the symbols into values. Dependent tasks, or subtasks, use standard OS/360 priorities. For any family of tasks,

an independent task must exist at the highest control level and its priority applies to subtasks in the normal OS/360 manner. The independent task is the building block for the RTCC real time systems. In fact, in the real time mode, a subtask literally can't get started without one.

The independent task--once defined--provides the necessary facility for data routing, a facility of the current 7094 control program.

Modification 2. Data Routing: On the current 7094 systems, real time input arrives in many cases multiplexed by message on a physical line. The messages which share the line need not be logically related and often are not. Each problem program to receive real time input defines selection criteria which can identify uniquely the messages required by that problem program. When a real time input message arrives, the control program examines the selection criteria to determine which, if any, programs are to receive the messages. Additional facilities available are:

- buffering--"accumulate five messages before placing an entry in the queue."
- time-relatedness--"place an entry in the queue every .5 seconds with as many messages as have arrived, even if none have."
- time out--"place an entry in the queue after five messages or .5 seconds, whichever occurs first."

For the RTCC modified OS/360 system, only independent tasks may receive routed data. Consequently, the control program is simply another originator inserting an entry in an independent task's work queue.

Control of real time input is maintained by the control program. An independent task's name is specified with the routing information. The task is involved only when work has been identified for it.

Modification 3. Data Tables: An additional requirement exists for a simple, efficient method of handling relatively small data sets. This was resolved by slight modifications to the partitioned data set facility. Libraries for real time would consist of programs and data tables as members. Like the programs, the data tables would be dimensionally static during the jobstep to permit in-place updating. Data tables could contain initial values or zeros prior to execution of the jobstep. Slight modifications to the basic LOAD and DELETE macro-instructions would permit handling data tables as logical extensions of subpool storage, shared between independent tasks. Write and read facilities would permit using data tables with simple (basically BDAM) accessing without DCBs, OPENs, DD statements and the like. Locking facilities would resolve by serialization any conflicts in concurrent usage.

The libraries--data tables and programs--would have received automatic servicing by virtue of an RTLIB statement, essentially lifted from the JOBLIB concept.

Miscellaneous Modifications to OS/360: After the "minor" modifications to OS/360 already specified, RTCC's other plans are somewhat anti-climatical. Independent tasks, data routing and data tables are the essentials. However, other problems must be solved.

Enqueue and dequeue facilities need to be provided where the resource is represented by a name, rather than a core location since independent tasks do not share core locations.

The core allocation algorithm will have to be modified for RTCC. The cyclic nature of the processing affects the choice of which program to overlay when room is needed.

Rollin/rollout must be solved another way. RTCC's experience with rollout has been that recovery from it is a sometimes thing with low probabilities. On the 7094 systems it was simpler to prevent rollout than to recover.

A facility for handling source and text libraries and using these in normal development work similar to the 7094 facility (and ATP) is being developed.

Two copies of the square root, cosine, etc., subroutines are too many; a LINK to these is tooinefficient. A compromise is being developed.

A method of handling applications systems parameters is needed to allow multiple programs to reference the correct current value of critical parameters, for example, capsule weight.

Directories, or subsets of directories, will be maintained in core to reduce the time for program fetch.

Undoubtedly, other problems are waiting to be discovered. We have only begun. But RTCC is confident that a highly responsive real time system and OS/360 can--more or less compatibly--live together and like it.

PRESENT RTCC CONFIGURATION

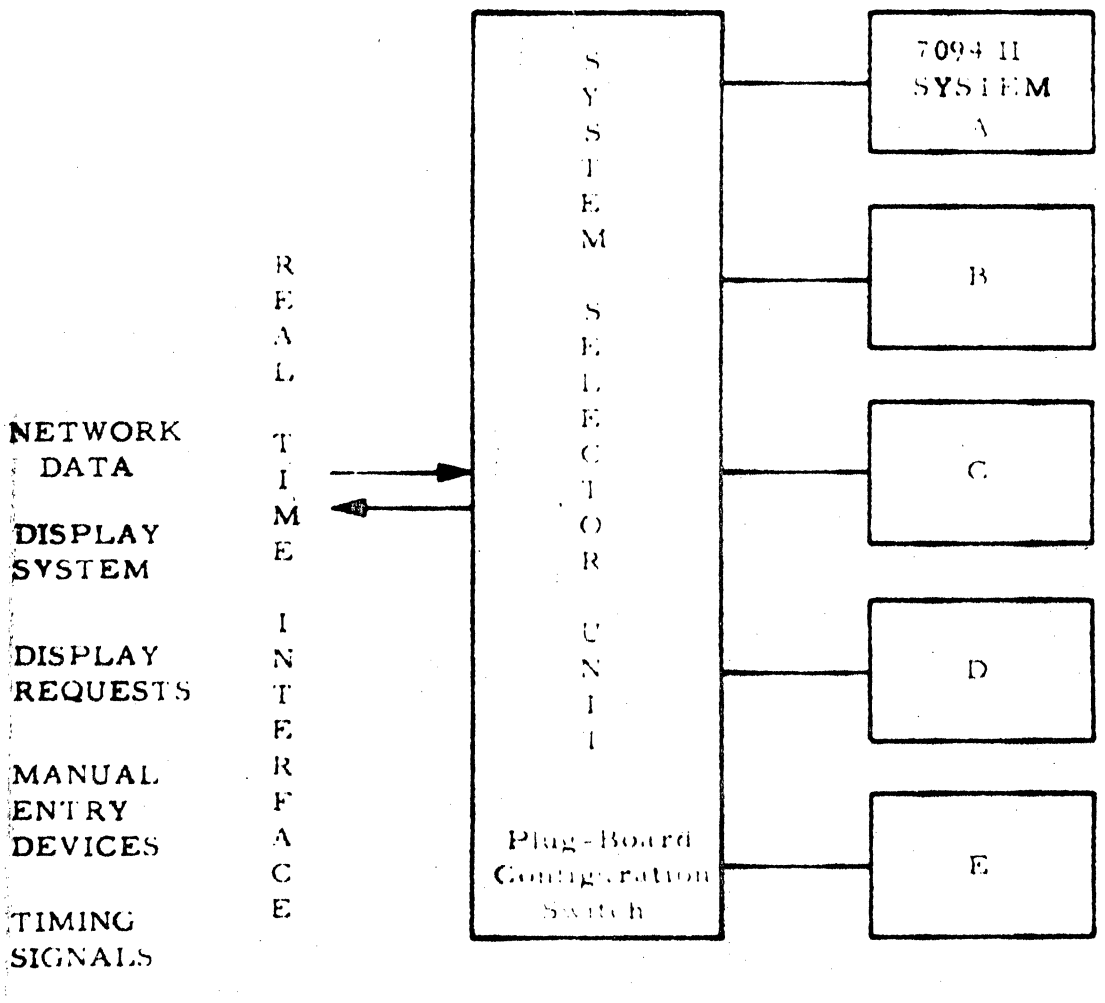

Figure 1a

RTCC 7094 II
SYSTEM

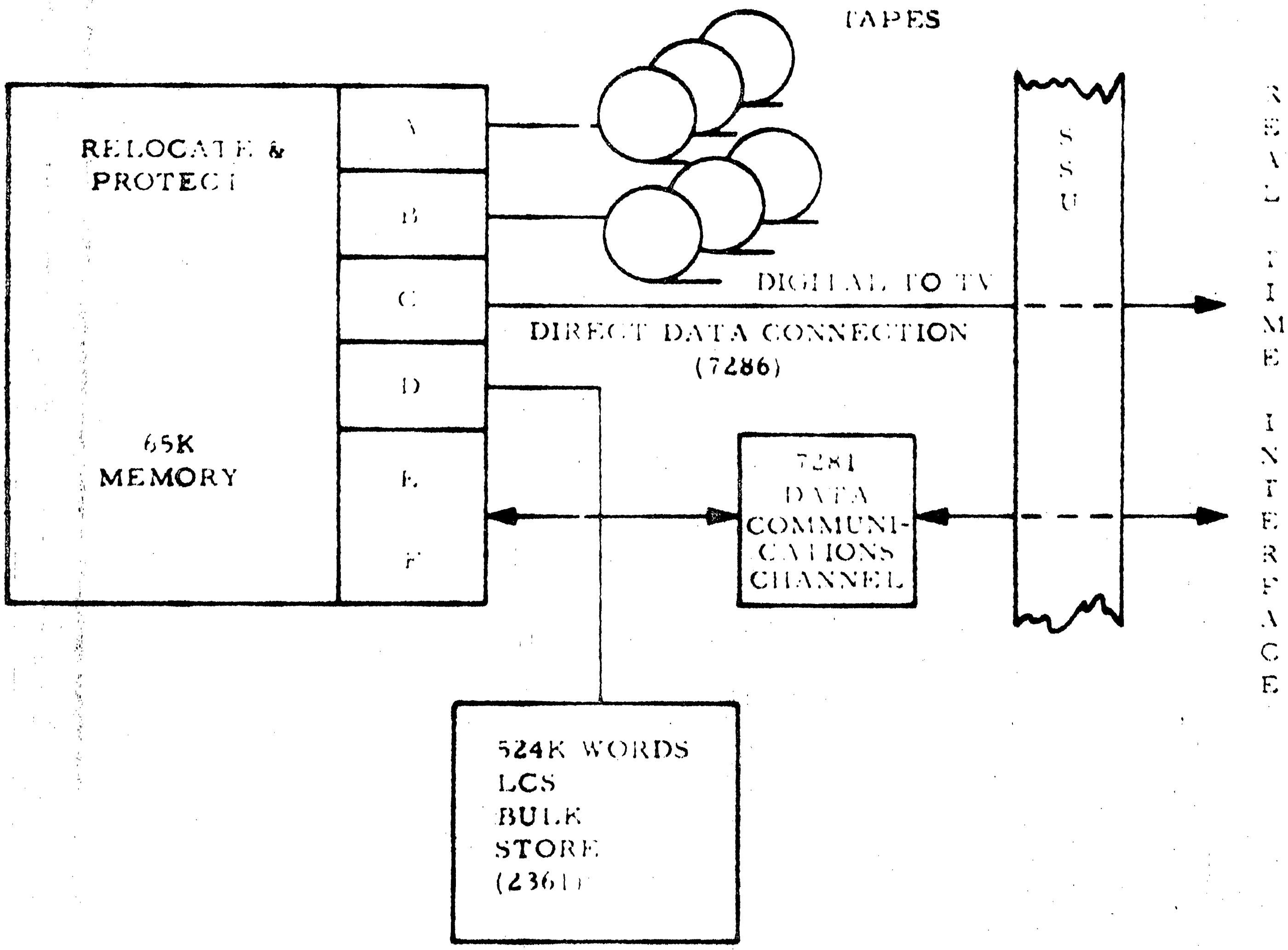
TAPES
RELOCATE & PROTECT
65K MEMORY
A
B
C
D
E
F
DIGITAL TO TV
DIRECT DATA CONNECTION (7286)
7281 DATA COMMUNICATIONS CHANNEL
SSU
REAL TIME INTERFACE
524K WORDS LCS BULK STORE (2361)

Figure 1b

FUTURE SYSTEM/360
CONFIGURATION

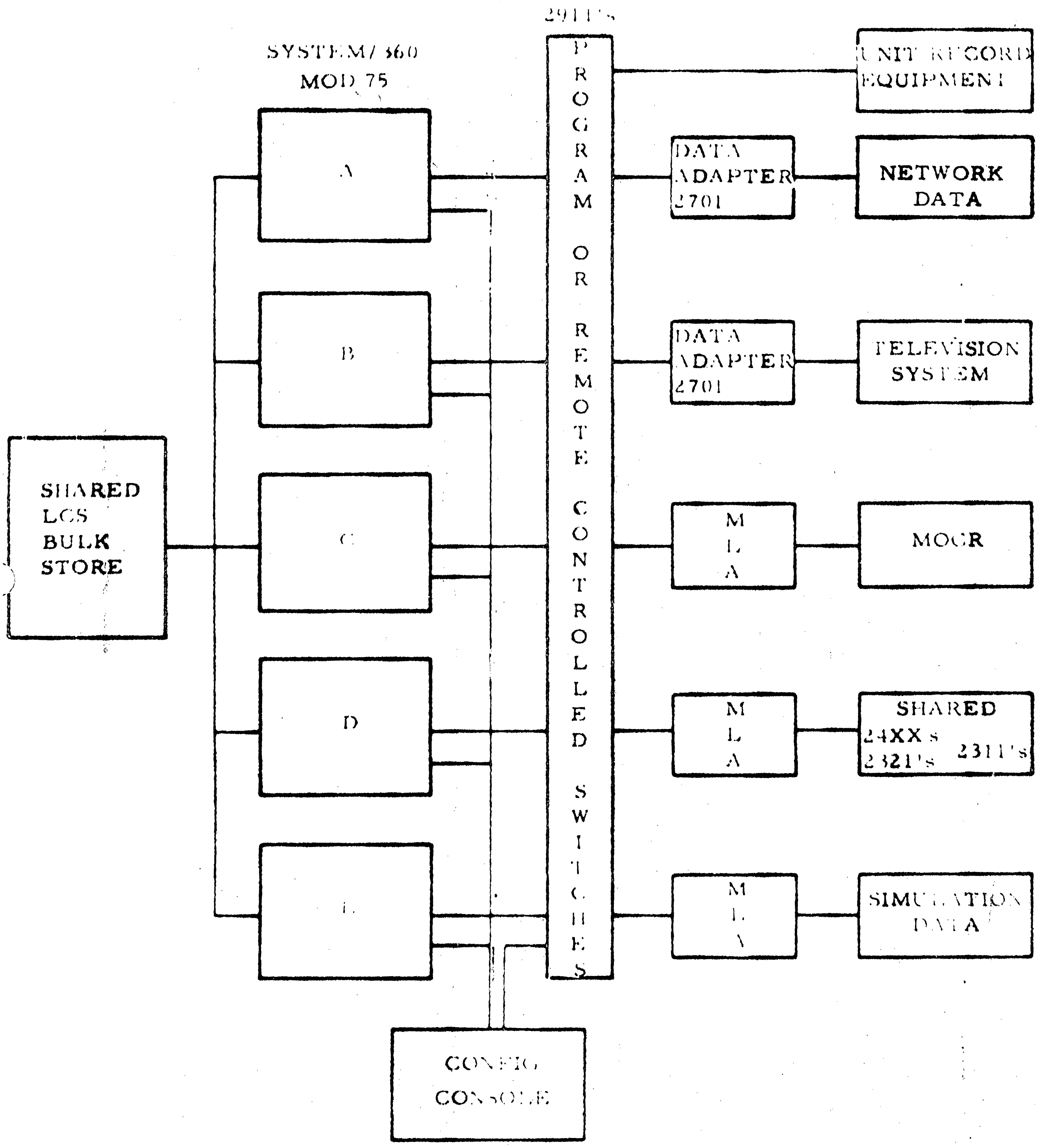

Figure 2a

RTCC SYSTEM 360
MOD 75 SYSTEM

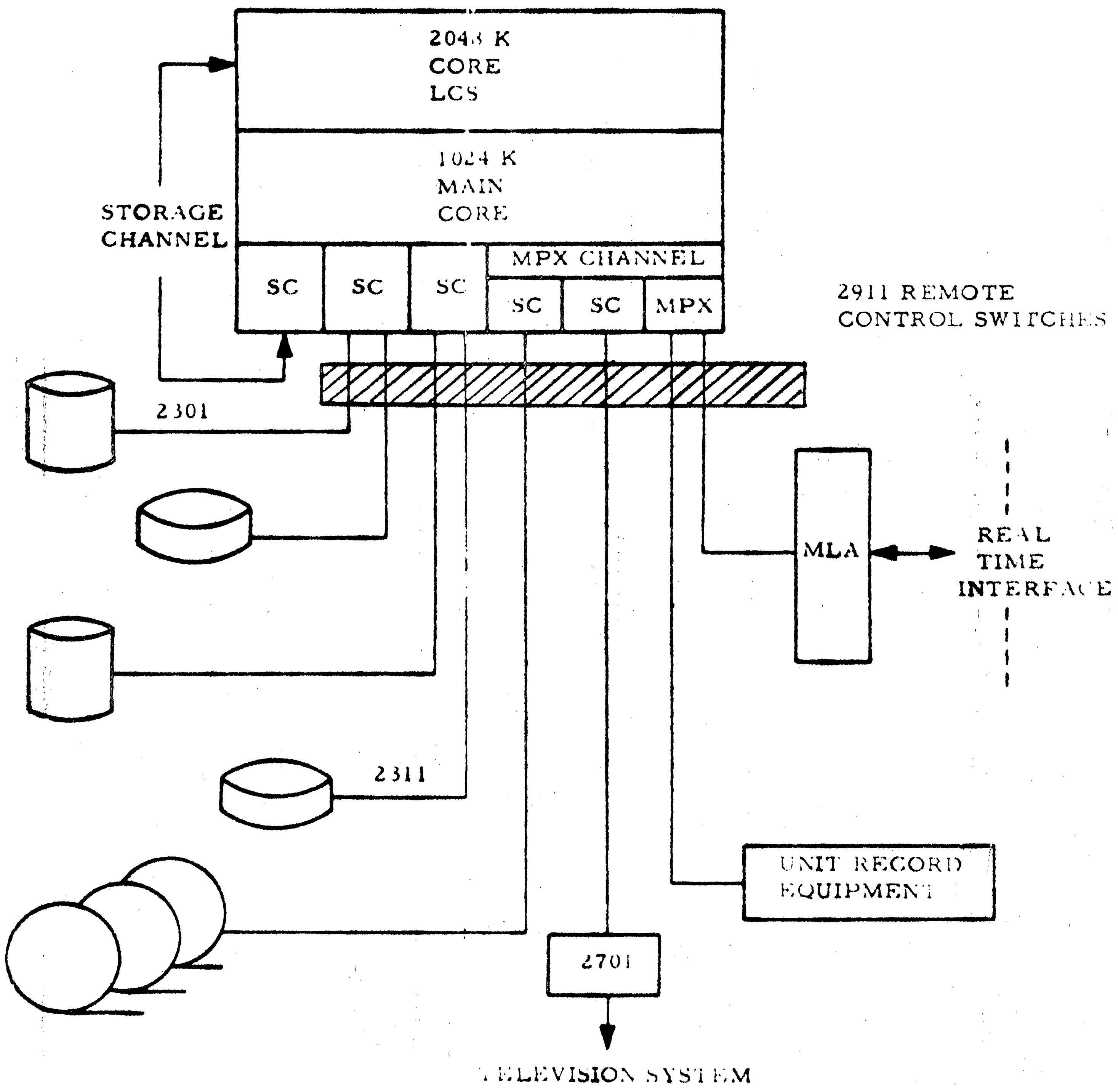

Figure 2b