X-565-72-134

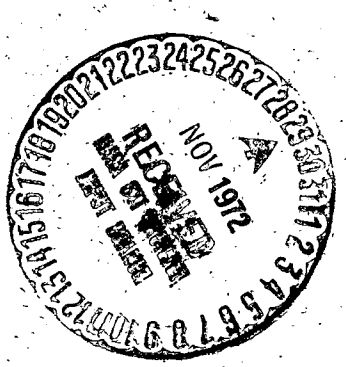
PREPRINT

NASA TM X- 66090

DESIGN OF A HIGH-SPEED REAL-TIME SYMBIONT

EUGENE I. GRUNBY

MAY 1972



GSFC

GODDARD SPACE FLIGHT CENTER

GREENBELT, MARYLAND

18 p.

X-565-72-134

DESIGN OF A HIGH-SPEED REAL-TIME SYMBIONT

Eugene I. Grunby

May 1972

GODDARD SPACE FLIGHT CENTER
Greenbelt, Maryland

ACKNOWLEDGEMENTS

The author wishes to express his sincere appreciation to Helen T. Bonk, Systems Programmer, who persevered through a syndrome of early scheduling and equipment problems; to William Robinson, designer of the 1108-DHE interface, who transformed existing equipment into a dependable network; and to Charles Bird, Univac Customer Engineer, who obviated a re-design of the interface for the new two-path strategy.

CONTENTS

DESIGN OF A HIGH-SPEED REAL-TIME SYMBIONT

HISTORICAL BACKGROUND OF TELEMETRY PROCESSING

Historically the Information Processing Division at Goddard has converted telemetry data from analog to digital form by two methods. The first method uses hardware designed for a specific telemetry format. Analog data is converted to digitized information on magnetic tape. The second method uses a CDC 3200 digital computer with a Beckman programmable analog front-end. Both conversion and "on-the-fly" data manipulation are accomplished by the latter method and result in partially processed information on magnetic tape. All tapes are then transferred to the UNIVAC 1108 multiprocessor system for reformatting and refinement of information, correlation of telemetered events with time and position information, and scientific analysis.

These two methods--while efficient--are not always attractive solutions to analog telemetry processing. They both use dedicated systems which do little more than the fundamental task of conversion. In addition, they produce magnetic tape--a rather unreliable and slow storage medium for rapid processing. With the addition of the Data Handling Equipment analog front-end to the UNIVAC 1108 direct and timely processing became possible.

THE DATA HANDLING EQUIPMENT (DHE)

The DHE functions as a time-division demultiplexer to convert serial PCM data inputs from receivers, tape recorders, or PCM simulators into demultiplexed digital and analog outputs. The PCM-DHE is capable of responding to a wide range of bit transmission rates and telemetry formats to produce outputs suitable for direct computer entry, data and function displays, and magnetic tape recordings.

The PCM data input to the DHE is composed of a serial time-division multiplexed data stream with timing references that mark specific points in the telemetry sampling or commutation cycle. Programs loaded into the DHE memory define the format to be received, timing relationship for decommutation, operating frequency, timing reference patterns and various parameters for checking. Using the program parameters the DHE converts serial PCM input data into parallel digital and analog output.

The DHE accepts satellite data in real-time or from analog tape. A Goddard-designed interface packs the data and sends it to the 1108. The interface has

three 36 bit registers which hold the data until the 1108 is ready to receive input. If the 1108 cannot accept the data and the registers are filled an overflow status is sent to the 1108. Data is lost during this condition.

GENERAL OBJECTIVES

One of the first objectives in designing a program for use on the UNIVAC 1108 was to avoid impacting an already heavy schedule of production work. Since interference with the batch job environment had to be minimized, the program could not use much core space or CPU time when operating at a real-time priority. At the same time it was desirable to receive telemetry data at rates up to 128 kilobits/second. This would permit accepting transmissions from virtually all Goddard satellites.

The second objective was to avoid making any modifications to the 1108 hardware, the real-time front-end, or the EXEC8 operating system. This rather obvious goal was prompted by limited manpower, the urgency of deadlines and the need to minimize impacting the 1108.

Since the real-time front-end is physically remote from the 1108 but adjacent to a teletype terminal facility, control over processing functions could be performed by the terminal operator. The ability to direct and monitor programmed DHE functions from the terminal and to observe DHE performance on its status panels and oscilloscope was considered very desirable and became the final general objective. This ability could also be used to finely tune the program to accomplish its telemetry task with a minimum of effort and computer facilities.

PROGRAM DESIGN

To satisfy the objectives of high speed data reception and minimal 1108 impact the program was structured as a spooling or symbiont task. The program transfers data from the DHE to an output device (fixed head drum, moveable head drum or magnetic tape drive). Essentially no analysis is performed. Interrupts associated with the completion of input or output transfers result in the activation of the next output or input operation. While all transfers are in process the program is in an inactive state and remains there until an I/O interrupt occurs. This permits the multiprogramming operating system to execute other tasks for the duration of the data transfer. To insure good response from the operating system the program is executed at a high real-time priority level; since batch programs are executed at a lower level, interference with honoring interrupts is reduced.

To avoid modification of the operating system it was necessary to design the program to execute as a worker task. The effect of this objective was to require that all I/O activity be performed upon requests to the operating system. The Arbitrary Device Handler module (ADH) of EXEC8, a direct access method, was used for this purpose.

The program is capable of executing either in a batch or demand terminal mode. As a batch program no dynamic controls or monitoring queries can be made during execution. Nevertheless, status information is available on the DHE hardware. Coordination with the 1108 operator is required for the initiation of execution to insure that the DHE is ready and in fact has not prematurely begun transmission. Termination of transmission is simpler since the DHE operator can send an end-of-transmission interrupt to the program.

As a terminal task the program has considerably more flexibility. In the demand terminal mode the program can initialize the DHE hardware, start and terminate I/O activity, and report on the quality of the telemetry input and the timeliness of the program activities. Some reports are produced periodically while others may be requested by a spontaneous command. In addition to reading out information the program is designed to accept dynamic modifications to various parameters.

The program uses the conversational ability of the terminal mode to best advantage during execution of hardware diagnostic tests and during program debugging. In the first instance, the DHE operator is adjacent to the equipment and can schedule and monitor tests in cooperation with a maintenance engineer. Of course, the engineer and operator may be the same person. In the second instance the programmer can trace and often correct malfunctions as they occur. While not an original objective of program design, conversational debugging was a natural outgrowth of program development.

PROGRAM ACTIVITIES

The UNIVAC 1108 operating system permits a program to have more than one activity. There are six activities in the DHE program. Since all activities are multiprogrammed the program is designed to multiprogram with itself. In this manner activities in a wait state--such as waiting for I/O--need not influence the progress of other activities in the program. Furthermore, since the 1108 multiprocessor at Goddard uses two Central Processors in its configuration, two activities may be executing simultaneously.

The Read Initiation Activity is the nearest approximation to a conventional main program. This activity begins the execution by establishing the Teletype Control Activity, if required, initializing the DHE and issuing read requests. It ultimately causes the scheduling of other activities by the resultant input interrupts. The Read Initiation Activity runs asynchronously since it is constantly sending read requests and is not explicity coordinated with other activities. This technique helps insure that a read request is always available when data is present.

Read initiation depends on the availability of core buffers, which receive input data, and ADH controls packets, which inform EXEC8 of the desired I/O operations. A control packet for I/O is the communications table between the program and the operating system. It contains such information as the file name, count of data words, location of a core buffer, the I/O operation and present status of the operation. By program design a core buffer contains several input areas defined for blocking purposes. Consequently, a new buffer is needed only when the last input area of the current buffer has been allocated. When there are no available core buffers, the Read Initiation Activity enters a wait state. Once an input area in a buffer is available, the activity acquires an ADH control packet and issues a read request for that area. If a packet is not available the activity enters a wait state until this condition changes. The Read Initiation Activity continues in the buffer and packet acquisition and read initiation loops until the DHE operator signals to terminate the program or until an I/O error is detected.

Other activities are responsible for cancellation of the wait states in the Read Initiation Activity. A wait for buffers is removed when the Output Completion Activity frees a buffer. The inter-relationship here reflects the symbiont process of the program; that is, input data is received in a core buffer, the same core buffer is used to output the data to a storage device, and the core buffer is available for re-use only when the output transfer is complete. A wait for ADH control packets is removed when the Read Completion Activity indicates that the input operation is complete and the packet no longer required.

The Read Completion Activity is responsible for accumulating summary performance measurements and for re-defining input data buffers as output buffers. Upon receipt of each input interrupt EXEC8 schedules the execution of this activity. The activity examines the external interrupt status word to determine if data was transferred normally or if the operator requested program termination. Since the ADH control packet governing input is no longer needed it is released for re-use. Also at this time it is determined if all the areas in the core buffer have been filled. Upon finding a partially filled buffer, the activity informs EXEC8 via an exit request that no further action is desired. When full core buffers are detected, the Output Initiating Activity

is informed before an exit request is made. Abnormal conditions exist when data was received within a six second read request interval, an input operation was terminated prematurely by an interrupt, or data pile-up occurred due to slow program response. This information is summarized in a table of totals containing number of time-outs, a count of interrupts and occurrences of data pile-up. Table items are available for inspection by the Teletype Control Activity.

The Output Initiation Activity transfers blocked telemetry data from core buffers to an output device. Once registered for execution by the Read Completion Activity, the Output Initiation Activity remains active until all full waiting core buffers have been scheduled for output. This design permits the output request rate to temporarily fall below the buffer arrival rate. Temporary delays in output cause an accumulation of buffers until the output device can recover. When all waiting buffers have been scheduled for output with EXEC8, the Output Initiation Activity terminates.

Scheduling buffers for output consists of two phases. First, a free control packet from a pool of output control packets must be found. If none are available the Output Initiation Activity places itself in a wait state until the oldest outstanding packet becomes available. The available packet is then tested to see if a preceding output operation terminated in error; detection of output errors results in activation of the Termination Activity and an exit from the Output Initiation Activity. The second phase initializes the packet for output of the specific buffer and computes the destination address if the output device is a drum.

The Output Completion Activity receives control upon occurrence of an output completion interrupt. The main function of this activity is to free output control packets and core buffers for re-use. If an output error is detected the Termination Activity is scheduled.

The Termination Activity has the responsibility of bringing the program to an orderly end. An operator may signal termination of the program by depressing a button on the DHE hardware or by making an appropriate keyin at the demand terminal. Also any input or output error will cause termination. An orderly end is achieved when the Termination Activity sets a flag to inform the Read Initiation Activity to cease execution. Then, since some input operations may be pending still, the Termination Activity waits until all input control packets are inactive. Finally, the Termination Activity schedules all remaining buffers and a buffer of software end-of-file flags with the Output Initiation Activity, waits until output transfers have been completed and exits to EXEC8.

The Teletype Control Activity is established at the beginning of program execution. When the program is submitted from a batch input device (e.g. card reader) no activity is established. When submitted from a demand terminal, the program registers this activity as an independent task. The Teletype Control Activity is normally inactive and awaiting a line of input from the DHE operator. In response to input the activity performs a desired action, prints an acknowledgement on the terminal, and returns to an input wait state. The various commands available to the operator are summarized in a later section.

SOLUTIONS TO A REAL-TIME RESPONSE PROBLEM

Since the original program design was guided by a general understanding of operating system performance, no specific system measurements were made in advance of implementation. The dynamic monitoring capability of the program itself was used to aid evaluation. With exercise of the program at various input bit rates it was discovered that above 20KB/Sec. significant loss of input data began to occur. Further study revealed that the time between receipt of consecutive input interrupts was less than the minimum operating system response time. That is, the operating system often could not issue a read request before the DHE sent the next interrupt. Since the 20KB rate was much less than the design objective of 128KB, some revisions were required.

Because reasonable care was taken during initial design, no unusual inefficiencies were found in the program. Also the program did not force the operating system into unnecessary areas having high overhead. Therefore the only apparent remaining approach was to consider possible hardware or operating system reconfigurations which would have a minor impact on manpower and equipment. This investigation led to one expedient and effective solution: to reconfigure the hardware and operating system to access two logically independent devices which in reality were one physical device, the DHE.

Logical replication of the DHE from a hardware viewpoint was dependent upon the UNIVAC 1108 multiprocessor components available at Goddard. The important items were two central processor units, an independent I/O path from each central processor to the DHE, and a shared Peripheral Interface (SPI) controlling which path had access to the DHE (See Figure 1: "Hardware Components Used in Real-Time Telemetry Processing"). Previously, the SPI was used as an electronic switch to insure that at any instant only one central processor had control of the DHE; operating system philosophy prevented parallel accessing of the DHE over redundant I/O paths. The new approach allows simultaneous read requests to be issued to the DHE with conflicts resolved by the SPI.

At the SPI, each read request identifies which I/O path will be active in the subsequent data transfer. Simultaneous requests are resolved on a priority basis. When one path is already active, a read request received by the SPI from the other path is stored in a holding register until the DHE announces the end of the current transfer by sending an external interrupt to the SPI. Occurrence of an external interrupt allows the SPI to relay the interrupt down the active path, to select the pending read request and its associated path, to send the request to the DHE and to accept data transfer on the new path. From a software viewpoint logical replication of the DHE was accomplished simply by specifying that the I/O paths on each central processor were independent paths to different devices, rather than redundant paths to the same device. The only changes needed here were to system generation parameters. EXEC8 could then issue simultaneous read requests across parallel paths to the DHE and insure that at least one request was always in advance of an interrupt from the DHE. The SPI would preserve the order of requests. The program was then modified slightly to permit the Read Initiation Activity to cyclicly send requests to these apparently different devices.

With these revisions rates over 200KB/Sec. could be realized. However, a new but correctable difficulty was introduced by these changes and was observed at the higher data rates. Since it now appeared that input interrupts were being received from two independent devices, the operating system would no longer consistently inform the program of the order of the interrupts.

Correction of this secondary problem was effected by providing a mechanism in the program to defer input interrupt processing until the expected interrupt was received. This method is equivalent to re-establishing the order of interrupts.

Three conditions are required to defer interrupt processing: First, to re-code the Read Completion Activity as a re-entrant routine; second, to provide each executing instance of the routine with the address of the ADH packet causing the interrupt; and third, to establish and maintain an ordered list of all pending ADH packets. (The second condition is provided as a standard feature of the operating system). Since the list of packets is in the same order as the desired sequence of interrupts a means of comparison is available. Each time an interrupt is received an instance of the re-entrant routine is activated. The routine compares the address of the ADH packet responsible for the interrupt with the oldest pending packet address in the list. If they do not agree the routine enters a wait state until an update to the list results in agreement. When agreement occurs, normal interrupt processing resumes. Also, the oldest packet address in the list is deleted so that the next oldest may take its place. In this manner only one instance of the routine is active at any time and it is processing the desired interrupt.

QUEUES

In order to obtain re-entrancy, ease of coding and efficient execution queuing techniques were incorporated into the program design. These techniques take the form of explicitly coded subroutines and implicit ways of using the hardware and operating system to exercise their queues.

The hardware queue resides in the Shared Peripheral Interface that is connected to the DHE. The SPI queues read requests that cannot be honored until the DHE completes a present read operation. The SPI resolves all access conflicts, switching of I/O paths, priority sensing, transference of all hardware signals and queue maintenance without the aid of software.

When the program issues output requests before previous output operations are complete the operating system queues these requests. This implicit queue is used by the program because EXEC8 overhead in processing output requests can be accomplished in a period when time is available. Unfortunately there is no implied queue for input processing with the ADH.

There are three pairs of explicit queues in the program: The available ADH packet and active (pending) ADH packet queues, the available core buffer and output core buffer queues, and the available output packet and active output packet queues. All queues are controlled by a common set of re-entrant subroutines and therefore all exhibit the same link and control characteristics (See Figure 2: "Queue Format").

The control subroutines establish queues in a first-in-first-out order. Bi-directional linking is used to permit detachment of oldest queue entries while new queue entries are being added. It is also possible to inspect the oldest entry of a queue without modifying the linkages. Another control feature is the ability to dynamically limit the maximum number of available entries in the three availability queues. Since the Teletype Control Activity can modify these limits at any time this feature is especially important during fine-tuning of the program. The teletype can also inspect the number of entries in each of the six queues for assessment of program performance.

The available ADH packet or active ADH packet queues is used by the Read Initiation Activity, Read Completion Activity and Termination Activity. The Read Initiation Activity acquires available packets, (See Figure 3: "Queue Item Motion"). The active queue is used by the Read Completion Activity to determine the desired order of input interrupts. Once a particular interrupt is processed this activity restores the packet to the availability queue. The Termination Routine simply waits until the active queue is empty before closing out the program.

A core buffer queue is used by the Read Initiation Activity, Read Completion Activity, Output Initiation Activity, Output Completion Activity, Termination Activity. The Read Initiation Activity acquires an available core buffer before an input operation is requested. After input operations have filled a core buffer, the Read Completion Activity places the buffer into the output core buffer queue and schedules the Output Initiation Activity. The output buffer queue is essentially a list of items awaiting transfer to the output device. The length of this queue is a measure of the output backlog. The Output Initiation Activity iteratively performs its writing task until the active queue is emptied. Each time output is requested, the associated item is removed from the queue but is not attached to the availability queue until output transfer is complete. The output buffer queue is unique because normally as new entries are being inserted old entries are being removed. When each output transfer is satisfied the Output Completion Activity determines which core buffer was used in the operation and attaches it to the availability queue. The Termination Activity acquires an available buffer, fills it with software end-of-file flags, places this and any unwritten buffers in the output buffer queue, schedules the Output Initiation Activity and waits for the end of output before retiring.

The available output packet queue is used by the Output Initiation Activity when an output request is to be issued to EXEC8. Since EXEC8 will permit only a fixed number of packets to be in process simultaneously, the maximum length of this queue is set to the same number to effect space economy. The Output Initiation Activity places the packet in the active queue. The Output Completion Activity restores the packet to the availability queue. The primary value of these queues is to allow dynamic observation and fine-tuning in the area of output.

PROGRAMMED WAIT STATES

The philosophy of symbiont processing is to transfer data from one I/O device to another without significant use of central processor time. An important design consideration is the method of waiting for interrupts and other key events. In this design three waiting techniques are used: waiting for a completion status in an I/O packet, waiting for a specified time interval, and waiting for input from a teletype (when in demand mode). In each case a wait state permits the operating system to seek other tasks for execution.

Since I/O is central to the operation of a symbiont many program events are logically associated with the occurrence of specific I/O interrupts. The following list summarizes programmed wait requests for a completion status in an I/O packet.

- No available input buffers, wait for completion of the oldest output request.

  This event is associated with the Output Completion Activity function of releasing an output buffer. Input and output buffers are from the same buffer pool.

- No available input packets, wait for the completion of the oldest active input packet.

- No available output packet, wait for the completion of the oldest active output packet.

A programmed wait request for a time interval occurs when the current Input Completion Activity is out of sequence and must wait for its turn. A one millisecond wait request is repeated until the sequence is restored.

The Teletype Control Activity is always conditioned by design to receive an input line from the teletype. This activity is placed in a wait state by the operating system until a completed line can be transmitted. This feature reflects the time sharing characteristics of the operating system.

TELETYPE COMMANDS

When the program is in the demand mode an operator issues spontaneous commands to the DHE equipment and directs program execution. A command consists of an action request and any number of operands. An operand may be a signed or unsigned number or a pre-defined mnemonic. To facilitate the entry of commands the program provides for free-field format. The operator is informed by an output message about the completion of each commanded action. The various classes of commands are summarized in the following paragraphs.

The first class of commands governs initialization. Normally initialization of hardware states and program parameters is predetermined by the program. However, the operator may command simple changes. He may send signals to the equipment to clear it and to condition it for telemetry input. This ability is useful in providing virtually complete control of the equipment during trial runs and between sequences of data reception. The operator may also specify an initial execution priority to the operating system to help balance system loading.

Another class of commands governs execution control. These commands start the processing of the telemetry stream and terminate it, delay program execution and permit it to resume (as may be desired during a long drop-out of data), and allow termination of the Teletype Control Activity without affecting other activities.

Monitoring of hardware and program performance is an important feature available from the teletype. By requesting the editing of various values by specifying their mnemonics the operator may learn the present count of data blocks lost by slow operating system response, the number of time intervals during which no data was received, and the current total of all accepted data blocks. Also he may inspect the length of software queues for available and active I/O packets and data buffers.

Modification of program performance is available primarily as a tool for fine tuning. The operator may change the number of maximum available items in the software queues. Since his action may occur at any time and since he may monitor the effects of these changes, reasonable values of queue lengths can be determined and subsequently incorporated. In this way program size and speed can be balanced before actual production begins.

To insure useful production the operator may command execution of diagnostic routines. These tests promote confidence that the DHE-1108 interface is functioning correctly and that the data, status information and time values being received from test sources is accurate. If problems are discovered, the maintenance engineer has a convenient tool available for assessing the area of difficulty and to verify correction of the malfunction.

A final class of commands is available to assist in future developmental programming efforts and was used in the implementation of the present program. This class is program debugging aids. The designer may inspect any memory location whether or not the contents are static or dynamically changing. He may also select a range of memory locations and take a snapshot of them. When desired the designer may change the value of any memory location. Hence, parameters, control points, and instructions may be modified to accelerate the convergence to a functioning program.

CONCLUSION

From a viewpoint of design and implementation it should be stated - perhaps obviously - that an attempt to extend the capabilities of a system beyond normal operating limits is a delicate undertaking. Flexibility of both program

design and personal predilections may be necessary to overcome difficulties. Alternatives must always be appreciated as appropriate steps to the "final solution."

From a pragmatic viewpoint success of the DHE symbiont design has been established. Training a small staff of operators has been simple since only a few hours of "hands-on" experience is required. Production has progressed smoothly for many months without a need to modify the program. The program has been readily adapted for a new but related application. Finally the symbiont has accommodated a wide range of data rates without perceptibly degrading the performance of the operating system or affecting overall throughput.

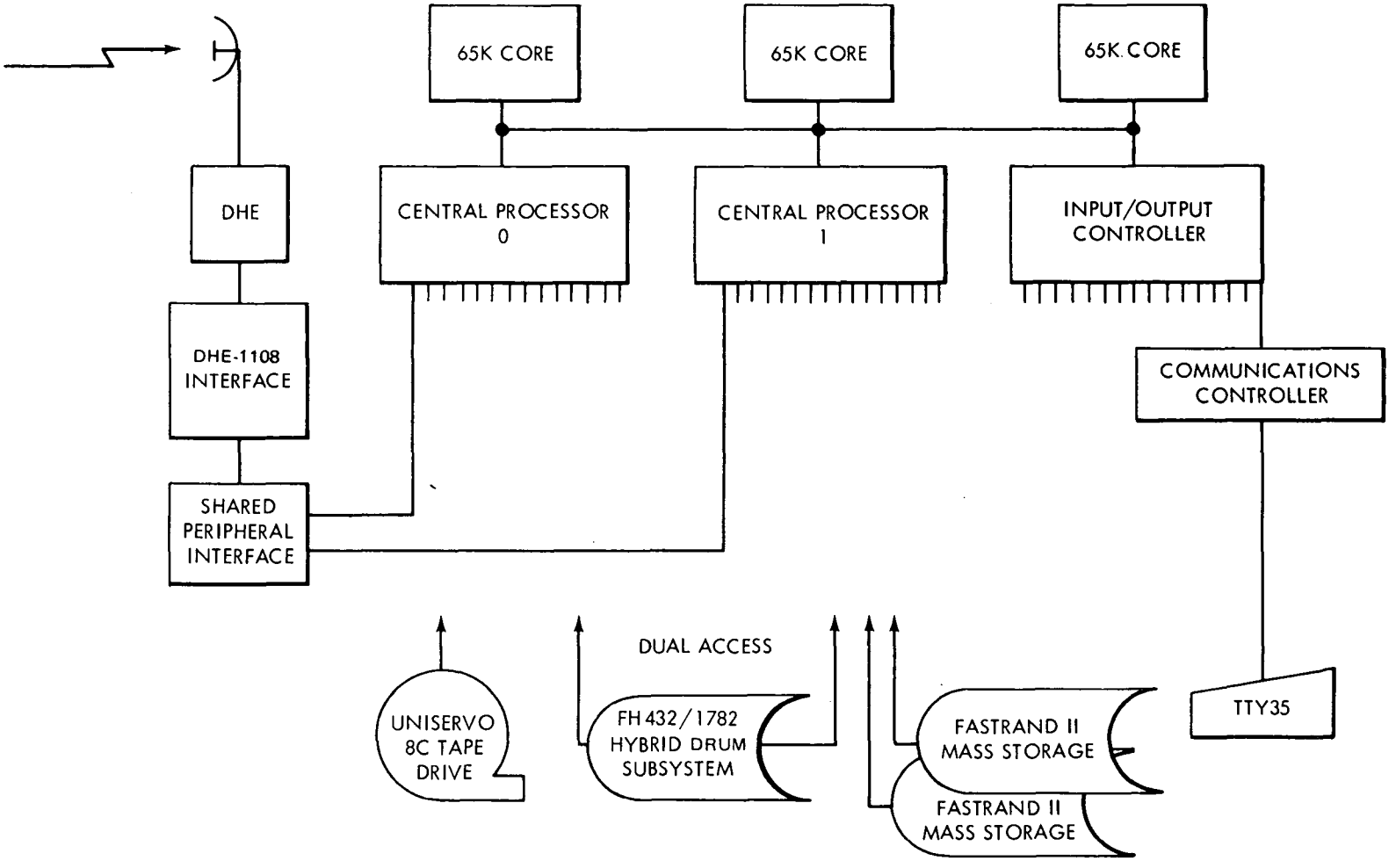


Figure 1. Hardware Components Used in Real-Time Telemetry Processing

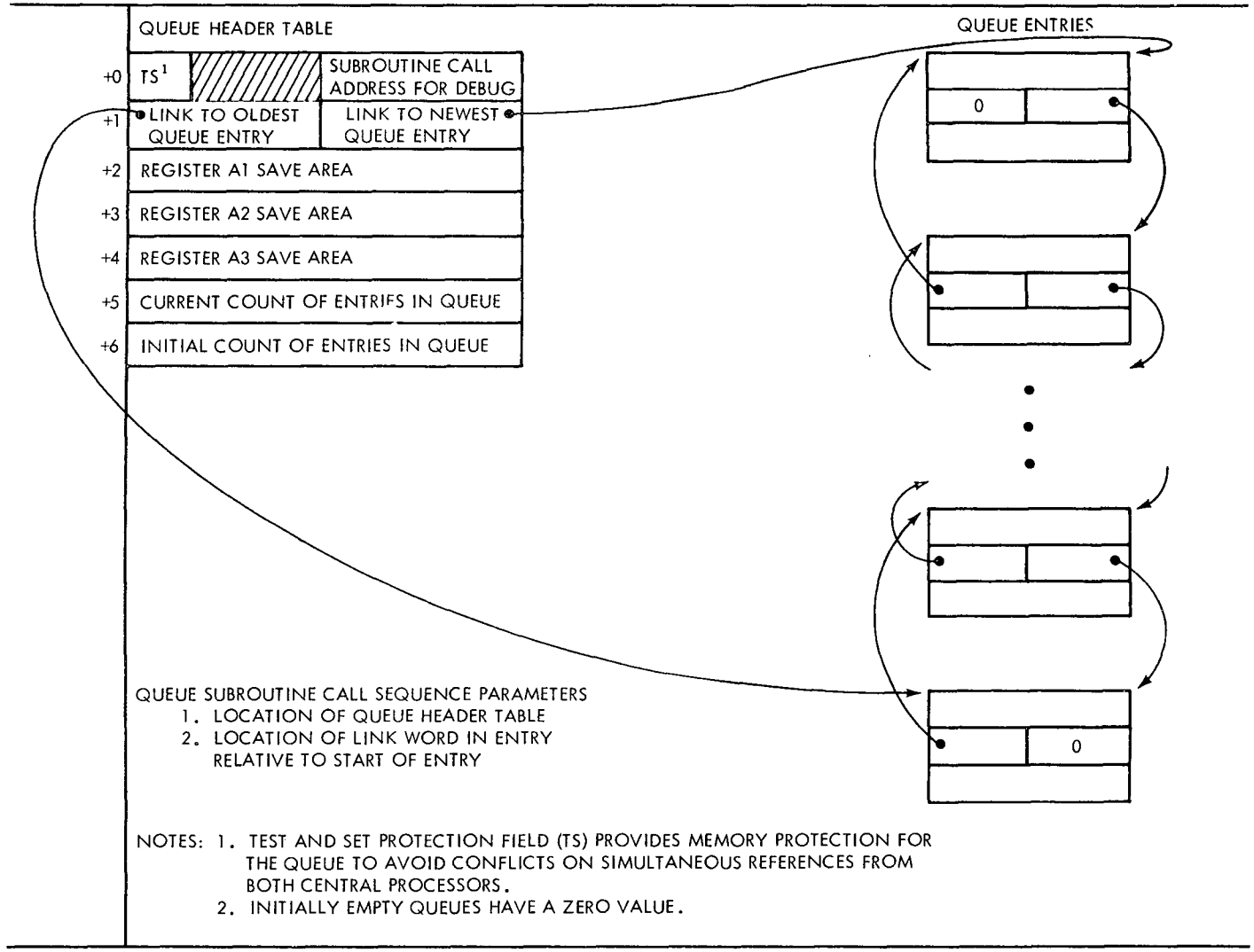


Figure 2. Queue Format.

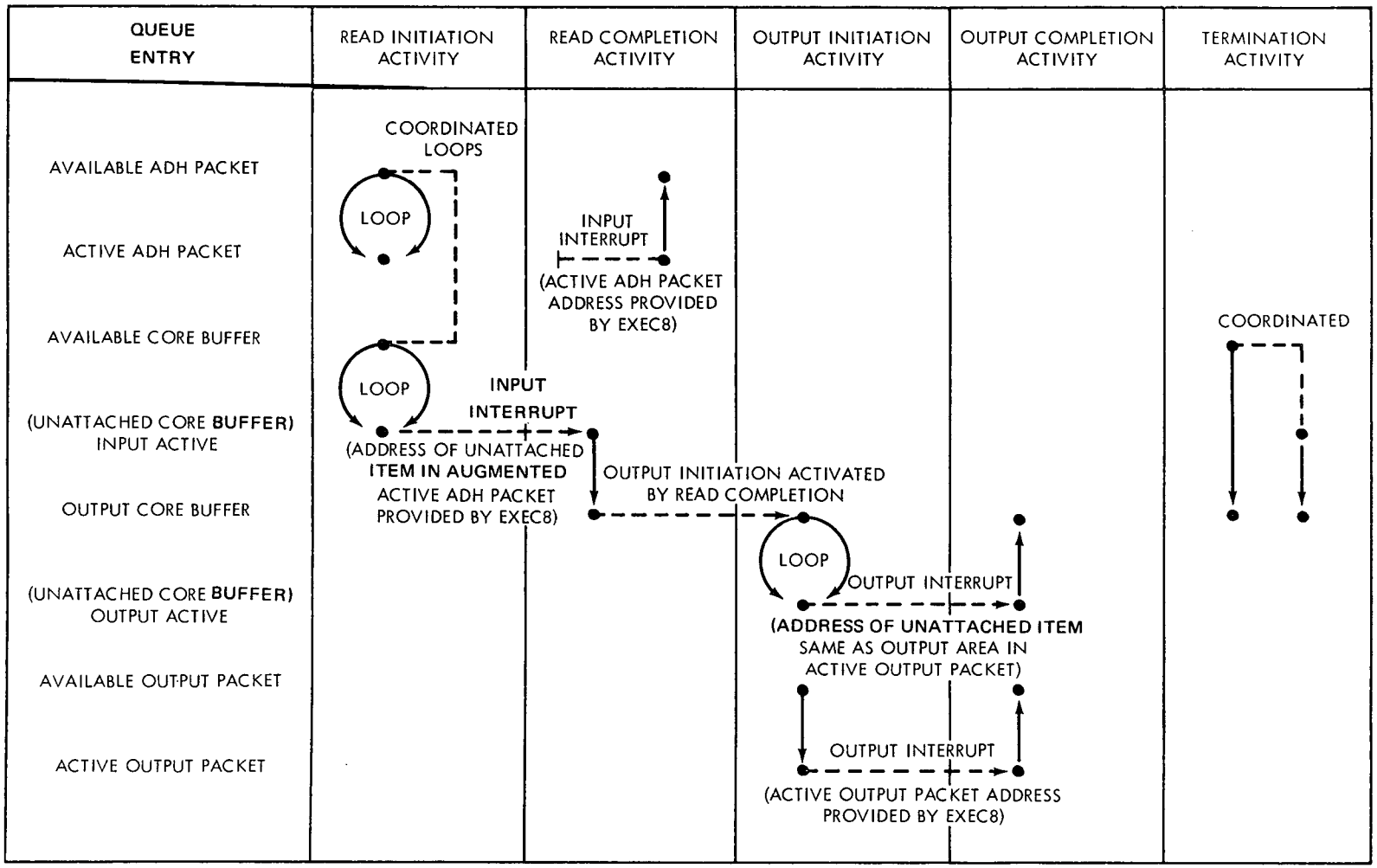


Figure 3. Queue Entry Motion.