Growing pains in the evolution of hybrid executives

by MARTIN D. THOMPSON

UNIVAC
St. Paul, Minnesota

INTRODUCTION

The development of a hybrid executive creates problems that go beyond those normally encountered in a digital executive. One of the biggest problems results from bringing two divergent human elements together in the same system. The analog user is “real-world” oriented with an extensive knowledge of the hardware involved in his problem. The digital user is often concerned with batch data processing and scientific computation that has been written in a high level compiler language.

While the digital user has some critical high priority programs to be scheduled, the main emphasis is directed toward completing the execution of as many programs as possible. Because of this, most digital executives stress orderly allocation of new programs, a minimum amount of re-allocation and effective utilization of all parts of the system.

The analog user generally wants control of the system on a demand basis and will retain a high priority level throughout the run. He occasionally requires direct contact with the operating system and will delay the run while deciding what action should be taken.

Part of the problem facing the hybrid community is to persuade the analog user to understand and appreciate the tools that are offered in a digital system. This will become a lesser evil as additional hybrid experience is obtained by the analog user.

The other problem is educating the digital systems programmer in the needs of the hybrid user. This is of immediate concern since the systems programmer will have the responsibility of modifying the digital executive in a way that will satisfy these needs and still retain an efficient operating system. In some situations, both worlds can be satisfied, but often there is a need for compromise.

The object of this paper is to show what happened when a digitally-oriented software group attempted to bend an existing capability to provide a sophisticated hybrid executive.

A hybrid education

When first given the task of providing the necessary software to control a Hybrid configuration, we began a study effort to determine what would be needed to provide a good software package. Obviously the first move was to peruse any information obtainable that was related to existing software in an attempt to start as close to the “state of the art” as possible. The next step was a survey of a number of hybrid users and designers in an attempt to determine their future needs and desires as well as what objections they had to the systems that they had developed.

A return to the drawing board ensued in an attempt to determine how best to adapt standard digital hardware and software systems in a way that would maintain the capabilities that existed for digital processing and still provide the real world needs of a hybrid computer. Since the system to be developed would be used by yet-to-be-named users to solve problems that could be quite diversified, it was mandatory that the chosen executive would be very generalized and offer modularity such that the basic philosophies could be transferred from one user to another, or from one digital system to another without major overhaul.

Some of the more critical needs for a hybrid system were itemized. Primarily, there was the need for high speed initiation of hybrid communication as a result of some stimulation that had occurred on the analog side of the system. Examples of this would be a time-out from an interval timer or a fault condition. In addition to fast reaction time, it is imperative that any variations in this time should be minimized and, if possible, completely removed.

In many of the critical real-time problems solved on a hybrid computer, the digital portion can easily become
compute-limited. Therefore it is very important that the available compute time in a given cycle be as predictable and liberal as possible. The chaining of I/O commands and independent data transfer to memory seem to be essential for maximizing and stabilizing compute time. Also, special consideration must be given to any possible reduction in executive overhead during a real-time operation.

Since most simulation laboratories are very cyclic in their needs for a large scale hybrid capability, justification for a large scale digital computer can rarely be made if the machine is completely dedicated. It is therefore necessary to retain a powerful batch processing facility under executive control within the digital portion of a hybrid system to supplement the high priority hybrid operations.

The hybrid portion of the executive should be modular to allow a user to specify only those segments that will be required, thereby conserving memory and allocation of peripherals. It should be general enough to allow the user to adapt it to various standard digital executives without major modifications. It should also be easy to adapt the software to variations in the hybrid system configuration such as changes in the number of channels used, or additions and deletions of hybrid devices or components related to these channels.

In some installations there will be multiple analogs operating independently, and each must react as though it were linked to a dedicated digital computer. To maintain this type of environment, it is necessary for all hybrid routines to be re-entrant, which would also reduce the reaction time for processing high priority interrupts. Another useful tool for a multiple analog environment would be a remote console located with each analog and providing the user full communication with the executive. This could be used for initialization of hybrid runs, normal debugging procedures or communication with the analog computer, as well as other operations normally offered on an I/O console.

In order to retain the full flexibility of use by various systems, all hybrid software routines must be as equally accessible from a high level language such as Fortran as they are by the assembly language. This would include any executive support routines such as those provided on a system program library. The program library would have to contain an assortment of standard hybrid routines such as an arbitrary function generator, a continual simulation language, and fast-response math routines, and be expandable to allow the user to add any routines that would be used frequently in his system.

At another level beyond the normal priority structure in a digital system, there is a need for a secondary priority network that is flexible and related strictly to the hybrid environment. This network should incorporate the speed of hardware and at the same time allow modification and interpretation under program control. To satisfy this requirement, as well as others previously mentioned, leads to the investigation of more effective design of the potential capabilities of the hybrid interface box.

In surveying our resources, it was learned that a very solid "UNIVAC 1108 hybrid system was in operation at the Naval Undersea Warfare Center (NUWC) in Pasadena, California. (See Figure 1). Although their system was quite specialized for solving their particular problems, it was evident that a considerable amount of planning and insight to the needs of the hybrid user existed. They had grown from a smaller-scale digital and minimum analog configuration to a sophisticated large scale system, and had expanded their executive development during the transition. NUWC and Univac personnel had modified the standard 1107/8 executive, which was basically a batch processing executive, to provide the real-time features necessary for their hybrid operation.

The most difficult problem encountered was to retain an effective batch processing capability on a machine that was to be dedicated to hybrid simulation on demand. The NUWC approach to the problem was to develop a "batch shelving" routine to provide high speed swapping between the batch and real-time mode of operation.

The first function of the shelving routine is to load the hybrid simulation program into the resident area of the drum, set appropriate retrieval information in the executive and issue a notification of a successful load completion to the console typewriter. The loading routine will then terminate, allowing the system to continue routine batch processing.

The second function of the shelving process is to respond to a type-in or an external interrupt on a hybrid channel that indicates a desire to execute the hybrid simulation program. A number of error checks are made at this point. The status of the operational program is
checked so that an attempt is not made to shelve a running simulation program and the status of the drum area is checked to verify that a simulation program is available for execution. If any errors are detected, the appropriate message is typed, the shelving request is ignored and batch processing continues.

If the verification is successful, the next step is to store all volatile flags and tables and control registers such as accumulators and index registers. After the environment is stored, the contents of user core are written on the drum and the simulation program is placed in user core. The console is notified that the real-time program is started and control is transferred to the starting address of the simulation program. The required time for the initiation process was determined to be less than two seconds.

The final function of the shelving process is the response to a type-in or an external interrupt requesting the termination of the simulation program. When this occurs, the simulation program is overlayed with the batch program and the batch environment is reset. The computer time used by the simulation program is calculated in a way that no simulation time is charged against the batch program. Less than one second is required from when the termination routine is entered until the batch program obtains control.

All device to device activity such as printing or punching cards from drum files is allowed to continue during the real-time operation. It is not necessary to suspend these operations since the CPU time used does not have a significant effect on the timing requirements of the real-time program.

In order to amplify the interface and provide a straightforward, timesaving software approach, the hybrid world was dispersed over five 1108 I/O channels. The dispersion was made as follows: the interval timer was placed on channel 0, unsolicited hybrid external interrupts were received on channel 1, the analog to digital (A/D) conversion system was on channel 2, the digital to analog (D/A) conversion system was on channel 3 and the discretes were on channel 4.

The result of this breakdown was a reduction in the interface requirements as well as a reduction in the communication control and command formatting requirements placed on the executive system. Examples of this would include D/A or A/D conversions. The command would indicate which channel should be used, and no directional commands would have to be formatted for transfer to the interface, since the channel used would be self-explanatory.

For the type of operation existing at NUWC their system was very satisfactory, and it was felt that this system would provide a solid foundation for building a generalized hybrid executive.

Subsequently, the NASA Manned Spacecraft Center in Houston, Texas indicated a desire to develop a hybrid facility using the 1108 multi-processing computer. See Figure 2. Having designed several dedicated systems, this type of operation offered many new challenges, but also provided the tools needed to create a much more powerful hybrid system.

The first and most obvious challenge was to blend the known hybrid software needs into the more sophisticated multi-processing executive. Although the pitfalls became more numerous in making modifications to an executive of this magnitude, it provided many solutions to system problems that had existed in converting a batch processing executive. It was a real-time executive with existing procedures for handling remote consoles while a real-time program was being executed.

The projected hybrid system at NASA will have up to six analogs operating simultaneously and independently. With this type of system, scheduling and allocation of facilities can become very critical. So it was necessary to extend the existing features for the assignment and release as well as the continual verification of the hybrid facilities.
The hybrid facilities were identified by labeling, as devices, such things as analog computers, groups of A/D and D/A channels, the interval timers and discrete devices. By identifying "hybrid real-time" as one of the facilities, the hybrid worker program could be assured that it would be the only hybrid program with critical real-time demands using the hybrid channel. By carefully assigning the proper priority to his real-time operation, and dynamically assigning the facility "hybrid real-time" only when the timing becomes critical, the user has the tools needed for a very efficient system.

The assigning and releasing of facilities can be done at run time through the control stream or dynamically by the operating program. The "Communication Routines" that verify and format the I/O packet establish the facilities to be used to execute the packet. The Hybrid Handler, the routine that executes the I/O packet, will validate the availability of facilities before execution. If the facilities are not available, the worker program will be notified and corrective action can take place. If a job is terminated for any reason, the standard executive procedures would release all facilities that had been allocated.

As would be expected with a multi-processing executive, most standard operations must be executed in generalized routines so that the executive will always have complete control of the system. Operations to be replaced in a batch type executive are the verification of an I/O packet, establishing memory restrictions and converting access control words from relative to absolute addresses, the saving of required environment in response to an interrupt, the queuing of interrupts when their priority is not sufficiently high enough to warrant immediate execution, and the queuing of I/O requests when the facilities required are not available.

By designing the executive modifications in such a manner, the hybrid runs will have no adverse effect on the operating system, and the central control structure of the executive will contain all necessary information on the status of all system components. This is especially desirable when there is external interrogation from conversational use of a remote device or when a programming bug or computer fault force an untimely abortion of the run.

It was during the NASA design effort that the technique was developed that made the greatest impact on the effectiveness of this over-all hybrid operating system. The 1108 has an interrupt lock-out feature that is set by hardware immediately after an interrupt is received. Any succeeding interrupts are suppressed until the executive does a minimal amount of housekeeping and releases the lock-out. This would normally take 40-60 microseconds. In addition to this time, an interrupt will not be recognized until the current instruction is complete. In most cases the instruction time is 750 nanoseconds, but in an extreme case of double precision floating point divide, the instruction time is 17.25 microseconds.

The composite of these times created a variation in the interrupt response time and, in situations such as setting analog components in response to an interval timer, this variation represented an uncomfortable margin of error. In designing a dedicated system, this situation was tolerable since the hybrid program could usually keep enough control over the entire operation to be waiting for any cyclic interrupts to occur. In response to unsolicited interrupts, the variation was usually tolerated.

By placing a small amount of capability in the hybrid interface, a method evolved that provided immediate initiation of I/O under the control of the interface. This method included a variation on externally specified indexing (ESI). Details on the technique are included in the description of the basic executive.

**General hybrid executive**

The out-growth of these previously-mentioned experiences was a modular set of routines that could be merged with an existing digital executive. The primary goals were twofold: to create a system that would allow the hybrid user the means for tailoring and expansion, and to retain the capabilities offered by the digital
executive. In order to accomplish these goals, the software was designed in two sections, Program Library and Executive Control. See Figure 3.

The first contained an executive-callable group of subroutines that may be required to support the hybrid operations. These subroutines would be retrieved from mass storage by the executive when required for execution of a hybrid program. Modification of these routines or the addition of new routines could be accomplished with ease.

Since the Fortran program library already contained most of the mathematical routines required for hybrid computation, it was decided that the first routines for the library should include the use of these mathematical capabilities for hybrid application tools. The first subroutine was a completely digital arbitrary function generator for the evaluation of functions with a maximum of four independent variables. The next effort was a continuous system simulation package that would utilize, in addition to the conventional math routines and the function generator, routines which simulated the various analog components. The language syntax was chosen so as to improve its flexibility and usability for a wide range of problems and yet retain the simplicity of a problem-oriented language such as Fortran.

To supplement the application routines, a number of operator-controlled routines were needed to assist the operator in debugging or initializing the hybrid programs. Included among these was a "hybrid interface simulator" that would overlay the Hybrid Handler and function generator, routines which simulated the various analog components. The language syntax was chosen so as to improve its flexibility and usability for a wide range of problems and yet retain the simplicity of a problem-oriented language such as Fortran.

To allow the operator to modify and interrogate component values in the analog computer, a routine was designed to interpret commands from the operator and, using the Communication Routines and the Hybrid Handler, execute the proper I/O packet and continue communications with the operator.

Another routine to improve the flexibility of the system provides the operator with a means to enter tables that indicate the initial conditions that the analog should have before the hybrid program can be executed. All values are set, a timeout is used when necessary to establish success, and the components are read to verify a normal completion. This routine can be executed during a batch processing mode of operation.

To make the basic debugging routines more comprehensive, a routine was provided that would, on breakpoint command from the worker program, read all of the components into a table of the same format as the one used to set up initial conditions. When the dump is complete, it is labeled and written on a mass storage device. This table may then be used as a restart feature at a later time or may be printed for perusal during post-run analysis.

The Hybrid Executive Control Routines include an initialize package, communication routines, and the Hybrid Handler.

The initialization portion of the executive control has the responsibility of answering the command from an external source that calls for allocation and execution of a hybrid program. In addition to retrieving the hybrid program from mass storage, parameters must be read and verified that will establish the hybrid environment required for the program to run. This includes the setting of general purpose access control words, error recovery routines with associated entry points, clearing all priorities and activating the hybrid channel(s) if necessary.

The Hybrid Communication Routines (HCR) have the responsibility of verification and formatting of the I/O packet that will later be executed by the Hybrid Handler. It was determined beneficial to the system development to adapt these routines to those of the analog manufacturer, since most manufacturers have existing Fortran programs written to perform diagnostic and maintenance tests on the conversion and analog equipment. In addition to these routines, if the digital computer is to be interfaced with existing hybrid equipment, the terminology used is generally well-entrenched in any operational programs the user has developed.

Analog manufacturers are placing greater emphasis on providing a compatible interface to the digital computer. An example of a large scale analog computer with a digital interface that is easily adapted to the ESI philosophy for hybrid communication is the Applied Dynamics AD4. The digital I/O commands necessary for control and operation of the analog are designed to make the analog appear as a standard peripheral device. ADI had previously formatted a set of 43 routines that could be called by Fortran or assembly language programs. These routines provided the means for initialization, data transfer, sensing and setting of discretes and mode control.

Provisions were made to use the Hybrid Communication Routines in either a test mode or a run mode. When operating in the run mode, no error detection takes place since the emphasis is placed on speed. If the test mode contains an error return argument, the error detected is noted and control is returned for user processing. If the test mode does not contain a return argument, the error is printed on the I/O console and continuation of the run is left to the operator.

The executive will make all normal checks on the parameters for an I/O packet. It will also validate the hybrid addresses referenced, and will make special
checks to determine if the argument type is correct for the request, and determine if enough data is available to satisfy the request, or if the range of the data is exceeded. Another condition that may be considered an error is the busy mode from the analog console referenced.

The I/O packet created by the Communication Routines will contain the location of the output buffer, any input buffers, an executive scratch area and the return entrance for normal and error conditions. Also included are indicator fields for the buffer length, priority numbers, error options, status of operation and a bit-encoded word indicating the facilities required for this operation.

The output buffer must contain not only output data but also the I/O commands required to control all communication over the hybrid channel. The interpretation of the request parameters for creating the output buffer must be listed as the primary function of the Communication Routines. Since the packet will not always be executed immediately, all of the verification cannot be done in the one set of routines. Because of this, fields such as the buffer length, facilities used, and the error options must be forwarded to provide the execution routines the information required for assembling the absolute access control words, facility allocation and error recovery procedures. When an I/O packet is to be executed as soon as it is formatted, the Communication Routines will select the proper entrance and transfer the I/O packet directly to the Hybrid Handler.

The Hybrid Handler is the keystone of Hybrid Executive. This is an assembly language routine that provides the interface between the normal digital executive procedures and the hybrid world. It is the only new routine that is keyed to the hardware restrictions of the digital system and hardware capabilities provided in the interface. Because of this, portions of the Hybrid Handler become very specialized and would have to be rewritten for any major change in the interface or if a different digital computer were used. Even though the coding would change, the philosophies should remain stable.

The Hybrid Handler is a composite of re-entrant software routines that are necessary for the execution of the I/O packets developed by the Communication Routines for utilization by the worker programs. These routines control any resulting communication between the hybrid world and the worker programs. The following includes the general task that will be accomplished by the Hybrid Handler:

1. Initialization
2. ESI environment control
3. Verification of the I/O packet
4. Activation of internally-stimulated I/O
5. Programmable priority network
6. I/O completion
7. Error analysis and recovery
8. Interval timer

The initialization portion of the Hybrid Handler has the responsibility of establishing the entire ESI environment for the hybrid channel. This routine is used by the Executive Control initializing package to set up the access control words and error recovery routines for system-dependent error conditions and clearing all priority flip-flops related to the initializing program. The hybrid channel will be activated if necessary.

A special phase of the initialization will allow the program to set up any error or system priority routines needed for the program.

A definition of the hardware activity is necessary to show how the Externally Specified Index (ESI) channel is controlled by the Hybrid Handler. (See Figure 4). When the hybrid interface detects a priority (priority < 128) from the analog world, it will add this unique priority value to the hardwired ESI bias for output (2000A), place this information in the lower 15 bits of the 36 input data lines, and raise the output data request line (ODR). This will cause the 1108 I/O control to form the appropriate pointer to the main memory access control word (ACW) related to this priority in the lower half of the input control register for the hybrid channel. The 1108 I/O will then transfer the appropriate pre-set block of memory to the hybrid interface. This block of memory may contain commands and output data.

If the commands in the output stream indicate input, the hybrid interface will set the lower 15 bits of the 36 input data lines to the hardwired ESI bias for input (2200A) plus the priority number, and the requested input data will be entered in the upper 18 bits with sign exten-
Growing Pains in Evolution of Hybrid Executives

From the collection of the Computer History Museum (www.computerhistory.org)
Handler. A routine provides for initialization of the priority system as well as entries for arming or disarming any particular priority or a range of priorities.

To supplement the priority hardware, the Hybrid Handler contains a programmable breakpoint priority where any monitored interrupts occurring with a lower priority will be queued until a command is received to lower the breakpoint to the level of the previous interrupt.

As previously noted, the normal completion of an I/O operation is activated by an output monitored interrupt. This causes the executive to store the priority and re-activate the hybrid channel. The Hybrid Handler will store status information in the I/O packet and for certain priorities the access control words are reset for succeeding interrupts. Intervals are enabled and control is returned through the executive to the location indicated in the I/O completion field of the I/O packet.

Another method for completion of an I/O operation is by the executive request, Hybrid I/O Cancel. This command will cause the Hybrid Handler to force the word count of the access control word associated with the given priority to zero. An "ignore state" is placed on the monitored interrupt for this priority and, after setting the appropriate status, control is returned to the canceling routine.

Two types of errors are noted by the Hybrid Handler, the parameter error in the I/O packet and an error condition that is detected by the hybrid interface. If an error is found in the I/O packet, control will be returned to the worker program. If the error is detected by the hybrid interface the Handler will check the error options offered in the I/O packet to determine if the operation causing the error should be re-executed a set number of times (with a time delay) or if the operation should be aborted. An error condition option field will exist for every priority where an error detected by the hybrid interface could exist. This will also include any priorities that are completely under the control of the Hybrid Handler. A bit corresponding to each type of error will indicate whether re-execution should be attempted. After detection by the Handler, the priority reserved for errors (177) is cleared.

An example of an error condition that could be detected by the hybrid interface is an illegal I/O command. Another is to have the termination command missing, which would happen if the word count in the access control word did not agree with the output buffer length.

Some situations, such as not having the power on the hybrid facilities referenced, will be referred to the operator.

The interval timer contains an active counter register and a holding register. The active register can be started, stopped and read. The holding register can be loaded under program control. When the active register is decremented to zero, the appropriate priority line to the interface is activated and, if a value has been entered in the holding register, the active register is reset and decrementation is initialized. A multitude of interval timers can be associated with the hybrid system.

The interval timer can be used to synchronize the updating of digital to analog conversions, placing the analog in the hold mode, timing programs or for timing the initialization of any other hybrid communication desired. It can be directed to decrement at seven different rates from one microsecond to one second.

SUMMARY

In conclusion, it is our feeling that we have designed a solid, basic hybrid executive that could easily be tailored to solve the problems of various hybrid users. The divergent philosophies of the analog and digital users had been successfully linked through the executive. The standard digital system, though occasionally compromised, remains a powerful tool. Additional efforts are always required to reduce the executive overhead and find ways to increase the effectiveness of the existing hybrid software. Hopefully, the program library will become more comprehensive as users expand their capabilities. As long as the hybrid executive remains dynamic, it can offer the hybrid user the capabilities that will be needed as hybrid systems are utilized in currently untapped fields.

ACKNOWLEDGMENT

The author would like to express appreciation to Eugene Guliani of Univac, Robert Roulette of EAI, J. T. McKinney of NUWC/Pasadena and Dr. B. B. Johnson of NASA/Houston, whose efforts greatly enhanced the design of the hybrid executive.