MULTICOMPUTER PROGRAMMING FOR A LARGE SCALE REAL-TIME DATA PROCESSING SYSTEM

G. E. Pickering, E. G. Mutschler, and G. A. Erickson
UNIVAC Division of Sperry Rand Corporation
San Diego, California

INTRODUCTION

The multicomputer programming techniques discussed in this paper were conceived and implemented in a large scale tactical data system developed for the U.S. Navy. Many of the details of this system are classified and cannot be discussed here. However, it can be stated that the subject system is a man/machine complex, primarily intended for fleet air defense and surface operations and maneuvering. The objectives of the system are: to provide commanders of forces afloat with a broad picture of the current tactical situation; to assist in directing operations in time to intercept and destroy potential threats; and to present the means to coordinate various weapon systems in a combat environment. These are achieved by automating, to a high degree, the collecting, processing, exchanging, and evaluating of large quantities of data through use of computers and digital data processing techniques.

The system objectives are accomplished in real-time. Data are received by the system from various sensing devices such as shipboard and AEW radars, sonar, IFF equipment and ECM equipment which are in continuous contact with the outside environment. Data entering the system are processed, analyzed and used by the system to influence or alter an event during the progress of that event, i.e., real-time.

Fundamental in the design philosophy of this system is the Unit Computer concept. In essence, the concept is a recognition of the need for computers of varying capability among ships of various types and operating modes. It solves this requirement by the use of standard computers operating in multiples to obtain increased capacity and functional capability, rather than use of several different computers each possibly of a different shape and size. However, in solving this varying computer capacity requirement, the Unit Computer concept established the need for multicomputer programs.

This paper concerns the programming problems encountered in designing and implementing operational programs requiring more than one Unit Computer and describes the techniques developed to solve two intrinsic problem areas:

- EXECUTIVE CONTROL IN A MULTICOMPUTER COMPLEX
- DATA TRANSFER BETWEEN COMPUTERS

A third problem is assignment and distribution of tasks between computers. To deal with this subject in the detail required would necessitate discussing classified material. That this is a problem of complexity should be obvious. However, it is evident that distribution of tasks between computers is a system design problem which must be uniquely solved for each multicomputer system.

---

*This paper was originally prepared for presentation at the 1961 Eastern Joint Computer Conference. Security regulations prevented its release until now.*
It should be emphasized that the multicomputer programming techniques discussed in this paper have been tried and tested for the last five years. Several multicomputer programs, employing two and three computers, have been delivered to the Navy. Thus, multicomputer programming, as discussed in this paper, is not theory, but actuality.

In preparation for the subject of multicomputer programming, it is first necessary to discuss pertinent characteristics of the Unit Computer developed by UNIVAC to satisfy the Unit Computer design philosophy.

I. THE UNIT COMPUTER

The Unit Computer is a general-purpose, stored program, solid-state machine.* Only those characteristics most pertinent to the subject of multicomputer programming are discussed in this paper.

Real-time Clock

Among the features that make possible the use of the computer in real-time applications is the real-time clock. The clock is contained in a special register within Magnetic Core Memory and can be referenced or set by the computer program. Time is maintained accurate to the nearest $2^{-10}$ second. The clock is incremented automatically by the computer upon signal of a $1/1024$ cps crystal controlled oscillator.

When the system is started, the clock is set to zero and time is automatically maintained relative to the start of the system problem. Although the modulus of the clock is approximately 7 days, the real-time modulus of the system may be somewhat less. The real-time modulus of the system is so defined that in scheduling tasks for execution (see discussion under Section II—Program Control in a Multicomputer Complex), the modulus of the clock will not be exceeded.

In a multicomputer program, the real-time clocks within all computers are synchronized. Upon initiation one computer transmits the content of its real-time clock to the other computers, which then set their clocks accordingly.

DATA TRANSFER LOGIC

A total of 14 input and 14 output channels is provided in the Unit Computer. Of these, two input and two output channels are especially designed for intercomputer communications. Each input channel and output channel consists of 30 data lines and 3 control lines. The special input/output channels differ from a normal input/output channel primarily in use of the control lines and in the associated channel logic.

The input/output logic of the Unit Computer requires that peripheral equipment request each input or output data word transfer. The computer then acknowledges receipt of the input or availability of an output. All input channels, including the special input channels for intercomputer transfer, and all normal output channels use Request and Acknowledge logic.

The control lines associated with request and acknowledge logic are shown in Figure 1. These are Input Data Request Line and Input Acknowledge Line for normal input channels and

---

* A more thorough description can be found in "Data-mation," December 1961, p. 45.
Output Data Request Line and Output Acknowledge Line for normal output channels. In addition to these two lines normal input channels have an External Interrupt control line and normal output channels have an External Function control line. The External Interrupt control line allows the external equipment connected to that channel to signal the computer of its immediate data transfer requirement which is honored by an interruption of the computer program. The External Function control line is used to specify a function desired of the external equipment connected to that output channel. An External Function Word to a tape control unit, for example, may specify *Rewind Tape Unit 2*.

The two special output channels use Ready and Resume logic for intercomputer data transfer control. The transmitting computer signals that it is Ready with an output data word which is interpreted by the receiving computer as an Input Data Request signal. The receiving computer upon accepting the transfer will send back an Input Acknowledge signal which is interpreted by the transmitting computer as a Resume. This reverse logic is necessary so that the transmitting computer will exhibit to the receiving computer the same control and timing characteristics as do the peripheral equipment.

The control lines associated with *Ready* and *Resume* logic of the intercomputer channels are illustrated in Figure 2. These are Input Data Request (or Ready) and Input Acknowledge (or Resume). In addition, intercomputer channels utilize an Input Buffer Status signal. The Input Buffer Status signal originates in the receiving computer and is defined by the state of the Input Buffer Active/Inactive Designator. The transmitting computer can sense the Input Buffer Status signal, thus determining status of the receiving computer's input buffer on that intercomputer channel.

A special output register (C1), incorporated in the computer, alternately services the two output channels reserved for intercomputer communication. All output intercomputer communications are time-shared through this special output register which holds data until a Resume signal is received from the receiving computer. However, if a Resume signal is not received within 32 to 64 seconds, the register is automatically cleared and the computer is notified of this condition by an Internal Interrupt called the *Intercomputer Failure Monitor*. This 32- to 64-second time interval is necessary to insure that intercomputer output with other computers is not suspended indefinitely. The Interrupt utilizes special entrance registers in Magnetic Core Memory for notification of faulty intercomputer output channels and to distinguish which of the channels is faulty.

Data transfer in or out of the Unit Computer, whether between computers or between computer and external equipment, is handled by buffered transmission of data with timing under control of the receiving computer or the external equipment respectively. The buffering process transfers consecutive words, starting at a given initial address through a given terminal address, on a specified input or output channel. A single computer instruction will initiate a buffer mode of data transfer. Once established, buffer transmission employs independent access to memory; the entire buffering operation proceeds to completion with no additional program references. Thus, the buffer mode of data exchange provides for input/output operating asynchronously with the main computer program; the computer continues execution of program instructions in the normal sequence.

The sequence of data and control signals for a normal transfer of data from one computer to another would proceed as follows:

a) Receiving computer sets Input Buffer Active Signal;

b) Transmitting computer detects Input Buffer Active Signal;

c) Transmitting computer places data on 30 data lines;

d) Transmitting computer sets *Ready* which becomes Input Data Request in Receiving computer;

e) Receiving computer detects Input Data Request;

f) Receiving computer samples 30 data lines;

g) Receiving computer sets Input Acknowledge line (returned to Transmitting computer as *Resume*);
Ordinarily the repeated instruction searches a table for coincidence. If coincidence does not occur, the repeat mode terminates and the instruction following the repeated instruction is executed. If coincidence occurs, the repeat mode terminates and the instruction following the repeated instruction is skipped. This method of searching a table for coincidence is used by the Executive Routine in controlling the execution of various tasks of the main program, as discussed later. A principal advantage of a repeated search is the reduction in instruction execution time since a memory reference is not required to read in a next instruction. As used in the Executive Routine, a repeat search takes $8 + 11.2 (N)$ microseconds where $N$ is the table item number on which coincidence occurs. A non-repeated search to accomplish the same end would require $8 + 40 (N)$ microseconds. Thus, the time required for program control is reduced by an approximate factor of 3 through use of the Repeat instruction.

**Compare**—This instruction compares the signed value of the operand with the signed value contained in either or both the arithmetic registers, $A$ and $Q$.Skipping the program's next sequential instruction is also allowed by this instruction if the Skip condition is met. The Compare instruction is used with the Repeat instruction to perform the search described previously. The execution time given in that description is for a repeated Compare. As will be described later in Section II of this paper, the time at which a task is scheduled to be executed is compared with the contents of the real-time clock.

**Return Jump**—The Return Jump instruction provides for transferring control to a specific set of instructions (subroutine), executing that subroutine, and returning control to the next instruction following the Return Jump. The Return Jump makes possible the modular program construction used whereby each separately defined task is implemented by a subprogram, which is referenced by the Executive Routine, and the identical subprogram is used in each program performing that task. A subprogram is a high-level subroutine; a subprogram usually references lower-level subroutines.
External Function—The External Function instruction has two meanings, one for intercomputer channels and another for normal input/output channels. If the instruction specifies an intercomputer channel, the connected computer is interrogated as to the status of its input buffer on the connected channel. If the interconnected computer's input buffer is active (i.e., an input buffer has been initiated and the connected computer is ready to receive data), the next instruction following the External Function is skipped. If the connected computer's input buffer is not active, the next instruction (which is normally a Jump) is executed.

For normal input/output channels, the operand of the External Function instruction (normally a control code) is transmitted to the external equipment connected to that channel.

Store Input Channel—This instruction stores the contents of the specified input channel at the address specified by the operand. An Input Acknowledge signal is then sent over the specified channel thereby informing the external equipment of the computer's availability to receive additional data and to indicate that the previous input was received. Since buffer mode of data transfer is normally used, the Store Input Channel instruction is used primarily to generate Input Acknowledge signals to External Interrupts.

Initiate Input Buffer (With Monitor) and Initiate Output Buffer (With Monitor)—These instructions establish an input or output buffer, respectively, via the specified input (or output) channel. Subsequent transfers of data, executed at a rate determined by the external device, are made directly into (or occur directly from) Magnetic Core Storage starting at the address specified by the operand. The storage address initially established is advanced by one for each individual transfer. The next current address is maintained throughout the buffer process in the lower half word of the Buffer Control Register for the specified channel; the last address of the transfer is maintained in the upper half word of the same Buffer Control Register. Each input channel and each output channel has a unique Buffer Control Register associated with it. The buffer mode will continue until it is superseded by subsequent initiation of a new buffer via the same channel or until the upper and lower half words of the Buffer Control Register are equal. Should the latter occur, a Buffer Monitor Interrupt is generated, causing the computer program to be interrupted and the Buffer Monitor Interrupt routine for that channel to be executed.

Multicomputer programs use the Initiate Input Buffer (With Monitor) and Initiate Output Buffer (With Monitor) instructions exclusively in transmitting data between computers.

II. EXECUTIVE CONTROL PHILOSOPHY

The system which made necessary multicomputer programming techniques has a number of functions that must be performed. In accomplishing these functions, the design objective is to use both men and machines to best advantage. Repetitive and routine operations are performed automatically by machines, whereas decisions of tactical importance are made by men. Although the focal point of the equipment complex is one or more computers, the system also employs high-speed digital data communication facilities, radar video processors, analog-to-digital data converters, digital-to-analog converters, and typical computer peripheral equipment such as magnetic tape and Teletype,* all of which provide for automatic inputs to the computer and/or automatic outputs from the computer. A complete display of complex and data entry devices allow men to monitor an ever-changing, yet accurate picture of the current tactical situation and to enter their intelligence and tactical decisions into the computer.

All equipment is connected directly or indirectly to the computers. For each function the computers must:

- receive data from the appropriate system equipment, consistent with data transfer rates of each equipment;
- correlate, process and (as necessary) evaluate data in performing the specified system function;
- present the results of processing and evaluation to men for their interpretation and decisions;

*Teletype is a trade mark of the Teletype Corporation.
transmit pertinent data and human decisions, as appropriate, regarding that data to the ultimate users, again consistent with data transfer rates of the equipment involved.

The system operates in real-time, therefore, operational requirements dictate the response times for individual system functions.

To relate a real-time system to real-time data processing then, one might say that real-time data processing is processing done immediately as a result of an input for the purpose of providing an essentially instantaneous response or output. Processing must, as a result, be completed at a rate greater than the input data rate or else the processing will fall behind so that eventually the system will be operating out of real-time. The system saturation point is reached when processing rate equals the input rate.

From the foregoing discussion, it is clear that any real-time system must have, at least, a data processing capability sufficient to handle the average input rate from all input functions. Moreover, since system inputs occur randomly, it must be assumed that all inputs could occur simultaneously. This would represent the peak load. The problem then, is how to schedule, or queue, the functions to be performed in such a way that under peak load conditions all functions would be performed within their real-time requirements.

An Executive Control Philosophy has been developed and implemented to solve the scheduling (or multiprogramming) problem. Executive Control Philosophy is a general term—explained specifically it means a recognition of the system’s tasks in order of system priority; distributing them among the system’s computers; and then controlling them in the individual computers of the system by an Executive Routine within each computer.

The Executive Control Philosophy is based on the fact that each function and concomitant handling of peripheral equipment defines a separate task (or set of tasks) that must be performed by the computers.

Each system task is performed by a computer subroutine (or group of routines) called a “subprogram.” By definition, a subprogram is any subroutine referenced directly by the Executive Routine.

An Executive Routine is contained in each computer of a multicomputer program and maintains complete control of tasks assigned to that computer. Over-all control of a multicomputer program by a Master Executive Routine is not employed. Rather a method of reciprocal control is used. One of the system tasks (i.e., subprograms) within each computer in a multicomputer system is concerned with intercomputer data transfer. The function of each intercomputer subprogram is to process the data received from an interconnected computer and upon completion of this processing to initiate the return data transfer. To effect reciprocal control, program logic must be such that the residual state of intercomputer transfer within each computer is input. Thus, the computer which has control of the intercomputer data transfer (i.e., next to initiate an output buffer transfer) actually has temporary control of the two-computer operation. Reciprocal control, therefore, means that control is assumed by one, then the other, of two interconnected computers. In systems utilizing three or more computers, reciprocal control would be used for each pair of interconnected computers. Details of intercomputer data transfer are discussed in Section III.

Subprograms within a computer are considered for execution on the basis of their priorities as system components. System priority for subprograms is relative, and is determined in the following manner: If tasks corresponding to subprograms A and B have response requirements of 100-milliseconds and one second, respectively, then subprogram A would have a higher system priority than subprogram B. System priorities for all other subprograms (or groups of subprograms associated with one system function) are chosen similarly.

Normally, tasks associated with processing inputs to the computer from system equipment (and outputs where stringent timing require-
ments are imposed by external equipment) have highest priority. This is particularly true if failure to accept inputs when available means loss of important, non-repeated data and possible system malfunction. However, in some cases where inputs are under more positive control of the computer or of a repetitive nature it is possible to lower the priority of equipment handling subprograms to insure processing of previously received data before new inputs are accepted. This has the advantage of effectively shutting off or delaying inputs during brief periods of saturation. Consider for example two subprograms associated with the display function: one subprogram periodically interrogates displays for manual action requests and the other processes these requests and generates the proper response. By assigning the processing subprogram a higher system priority than the interrogation subprogram, the rate of processing will automatically control the interrogation rate. Only under extreme saturation for a prolonged period would the operator even notice the delay; thus under normal operation the system response time for the display function would be honored.

For the reasons given above—to protect against loss of important non-repeated data and to insure processing of previously received data within the real-time response requirements—intercomputer subprograms are assigned a relatively high priority. Conversely, each computer might contain a subprogram of lowest priority to perform such operations as program checking when no other system task required execution.

Consider a list of all subprograms ordered by priority. The Executive Control Philosophy, applied in an elementary form, would dictate that this list be scanned sequentially, executing subprograms when they are needed. After each subprogram execution, the scanning process would be resumed starting with the highest priority entry. Thus, under a peak load condition where all subprograms may require execution, the top priority subprogram would be executed instantaneously, the next highest immediately following, and so on. The resultant response time of each subprogram would be the sum of previous execution times for all higher priority subprograms, recognizing that some of the higher priority subprograms may be executed more than once. The assumption must be made that sufficient computational capability exists to perform all assigned tasks. If this is not true then complete penetration of the list will not be realized and the only solution is additional computers. If computational capability is sufficient to handle the average input rate for all functions, then under peak load conditions higher priority tasks are performed first and lower priority tasks are delayed until time is available. Thus, a lower priority task may be defined as not having a critical response time so that late execution will not seriously degrade the system. An example of such a task is output to displays which might need to occur at a rate of 15 times a second to maintain a flicker-free presentation. However, during peak load, which probably would last for less than a second, reducing the output rate to 14, 13, or less would normally go unnoticed.

As stated earlier, the real-time requirements of any system task are such that processing is done within the required response time. The Executive Control Philosophy takes advantage of this characteristic by providing the option of assigning a variable priority to a subprogram corresponding to a particular task. Consider again the subprogram corresponding to task A. Subprogram A could be executed upon demand as a top priority item, thereby completely satisfying the response requirements. But recall that the maximum response time tolerable was 100 milliseconds. A response from subprogram A would therefore be acceptable as late as 100 milliseconds after the original demand. Since the present computer load may be quite high, it might be desirable to defer execution of subprogram A for, say, 50 to 75 milliseconds. This is done in the same priority list described earlier, except that a time element is included in the ordering of the subprograms. Multiple entries for subprogram A can be made in the priority list as follows: An entry is made as low priority item for execution immediately, as an intermediate priority item to be executed in 50 milliseconds, and as a top priority entry to be executed in 75 milliseconds. Thus, after a processing demand has been received, subprogram A will be executed:

From the collection of the Computer History Museum (www.computerhistory.org)
a) immediately out of the low priority entry, if the computer load is low;

b) out of the intermediate priority entry in 50 milliseconds, if the computer load is moderate; and

c) out of the top priority entry in 75 milliseconds, under peak load conditions.

This assures that under all load conditions subprogram A will be performed within the required response interval. It should be noted that the above procedure will result in automatic “smoothing” of the processing load, and will still allow execution of each subprogram in the absolute minimum time, consistent with system priority and existing computational load. A subprogram is executed only when its “flag” is set. By definition, a flag is set when the time at which the task is scheduled to be executed is equal to real-time.

2) When the Executive Routine detects that a flag is set, it reschedules the time when the next flag for this subprogram will be automatically set. This mechanism allows for tasks that must be executed at an approximate periodic rate, as mentioned earlier for output rate to displays.

3) One subprogram may set or partially set the flag of another subprogram. This mechanism is used when it is either necessary or desirable to execute certain associated subprograms in a definite sequence. For example, a subprogram which processes data placed in computer stores would be flagged by another subprogram which acquires that data through interrogation of an external equipment. Extending this same example by adding an additional processing subprogram, it may be necessary to complete both processing subprograms before reinitiating the interrogation subpro-

4) A particular subprogram may reset its own flag if, after receiving control, it determines that more than an allowable amount of processing time is required to complete the task. This mechanism is utilized only by lower priority subprograms and is designed to guar-

---

**Figure 2. Special Channel Configuration.**

Four mechanisms are provided to set these flags. Each subprogram normally uses but one of these mechanisms.

1) External Interrupts or Internal Buffer Monitor Interrupts set the flags of subprograms associated with servicing external equipment and most data transfers. Consider the task of transmitting data over a communication link. Communication terminal equipment would send an External Interrupt to the computer thereby signaling the start of a transmission period. The computer must have data to be transmitted available to the communication equipment within a certain time period. If the communication link is not operating, there is no need to execute the communication transmit subprogram.

2) When the Executive Routine detects that a flag is set, it reschedules the time when the next flag for this subprogram will be automatically set. This mechanism allows for tasks that must be executed at an approximate periodic rate, as mentioned earlier for output rate to displays.

3) One subprogram may set or partially set the flag of another subprogram. This mechanism is used when it is either necessary or desirable to execute certain associated subprograms in a definite sequence. For example, a subprogram which processes data placed in computer stores would be flagged by another subprogram which acquires that data through interrogation of an external equipment. Extending this same example by adding an additional processing subprogram, it may be necessary to complete both processing subprograms before reinitiating the interrogation subpro-

4) A particular subprogram may reset its own flag if, after receiving control, it determines that more than an allowable amount of processing time is required to complete the task. This mechanism is utilized only by lower priority subprograms and is designed to guar-
<table>
<thead>
<tr>
<th>EXECUTIVE FLAG TABLE (EFT)</th>
<th>EXECUTIVE TIME TABLE (ETT)</th>
<th>EXECUTIVE JUMP TABLE (EJT) (sub-program)</th>
</tr>
</thead>
<tbody>
<tr>
<td>n</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>24 seconds</td>
<td>A</td>
</tr>
<tr>
<td>2</td>
<td>24 hours</td>
<td>B</td>
</tr>
<tr>
<td>3</td>
<td>24 hours</td>
<td>C</td>
</tr>
<tr>
<td>4</td>
<td>10 seconds</td>
<td>D</td>
</tr>
<tr>
<td>5</td>
<td>24 hours</td>
<td>A</td>
</tr>
<tr>
<td>6</td>
<td>1 second</td>
<td>E</td>
</tr>
<tr>
<td>7</td>
<td>24 hours</td>
<td>B</td>
</tr>
<tr>
<td>8</td>
<td>50 millisec. + lockout</td>
<td>F</td>
</tr>
<tr>
<td>9</td>
<td>24 hours</td>
<td>G</td>
</tr>
<tr>
<td>10</td>
<td>24 hours</td>
<td>H</td>
</tr>
<tr>
<td>11</td>
<td>24 hours</td>
<td>I</td>
</tr>
<tr>
<td>12</td>
<td>24 hours</td>
<td>A</td>
</tr>
<tr>
<td>13</td>
<td>1 minute</td>
<td>J</td>
</tr>
<tr>
<td>M = 14</td>
<td>30 seconds</td>
<td>K</td>
</tr>
</tbody>
</table>

NOTE: 1. Subprograms D, E, F, J, and K are of a periodic nature.
2. Entries in the ETT are based on a 24 hour real-time modulus of the system.
3. The 24 hour entry in ETT represents a very large delay. This makes sure the flag is cleared automatically after the first reference.

Figure 3. Example of Executive Table Structure.
antee meeting the more stringent response time requirements of higher priority subprograms. When the Executive Routine again considers this task for execution on the basis of priority, control will be returned to enable further processing. The cycle will continue until all processing of this task is completed.

The Executive Control Philosophy is implemented through a unique method of table control. Three tables—Executive Flag Table (EFT), Executive Time Table (ETT), and Executive Jump Table (EJT)—contain entries for each assigned subprogram which are ordered by system priority. Each computer in a multicomputer program would have its own set of three tables with the subprogram entries unique to that computer. Figure 3 illustrates the one-to-one correspondence between the tables and also illustrates the use of a variable priority and lockout technique discussed earlier.

For a given subprogram, \( EFT_n \) contains the time (per real-time clock) at which the subprogram is to be executed, \( ETT_n \) the automatic delay before the subprogram is to be repeated, and \( EJT_n \) the address of the subprogram. If the subprogram is to be executed at other than a periodic rate, the content of \( ETT_n \) is a sufficiently large number (usually the real-time modulus of the system) so that the task will not be repeated until one of the other flagging mechanisms resets the flag. In operation, the Executive Routine sequentially compares the computer's real-time clock against the entries in the Executive Flag Table, starting with the task of highest priority. In the case of a “hit” (the clock is greater than or equal to \( EFT_n \)), the corresponding Executive Time Table entry is added to current time, the sum is stored in \( EFT \), and control is transferred to the corresponding subprogram as specified in \( EJT \).

If the “hit” corresponded to a multiple entry subprogram, all \( EFT \) entries corresponding to that subprogram would be reset. When control is returned to the Executive Routine, or if no hit occurs, the search of \( EFT \) is repeated, starting again with the task of highest priority. Figure 4 is a flow diagram of Executive Routine logic.

The description of Executive Routine operation, as discussed in this paper, is short—but complete. Main control loop of the Executive Routine consists of eight instructions, one of which is the powerful Repeat instruction used in the search operation. The efficiency of a repeated Compare and the extremely small number of additional instructions necessary ensure that only a minimum amount of computational time is usurped for the important area of program control.

III. DATA TRANSFER BETWEEN COMPUTERS

To accomplish intercomputer communications, stringent control must be exercised in establishing and terminating intercomputer buffers in each computer, thus insuring that data transfer between interconnected com-

Figure 4. Executive Control Program.
puters will stay synchronized. Interlocks and check points for intercomputer communication are required in the hardware and in the programs. Special characteristics of the computer are combined with the program to foment this multicomputer concept.

As can be seen from the delineation of hardware characteristics in Section I, significant and powerful means are available to produce an efficient program for intercomputer communication. Internal Interrupts provide excellent program efficiency, in that they obviate periodic sampling requirements for detecting “end of buffer.” Usually, within microseconds of buffer termination, the main program is interrupted to permit execution of program control tasks required by buffer termination.

In the following paragraphs, program characteristics and logic will be discussed to show the program’s use of the computer’s capability for data transfer.

PROGRAM CHARACTERISTICS

Elements of the logic governing the design of intercomputer programs are variable length buffers and reciprocal control. Fundamental to obtaining variable length buffers are the formats—message and buffer—and the method of controlling buffers. Reciprocal control means that control of the buffer operation is assumed by one, then the other, of two interconnected computers. In accomplishing this “ping-pong” operation, program logic is such that the residual state of the intercomputer channel is in the input mode.

INTERCOMPUTER FORMATS

Message Format

There are two types of information contained in the message formats: control bits that indicate the type of format and the validity of the information, and data bits which are the actual parameters and operands being transferred (see Figure 5). Basic message formats used are:

a) Two-word—This format provides an efficient control-bit/data-bit compromise and is used for most types of transfer; and

b) Multi-word—This format allows for any data configuration that will not adapt to the two-word format.

Both formats have a common control bit area in the lower 15 bits of the first word. These bits are used to: 1) specify exactly what action is required on the data contained in the format, and 2) detect bit errors in the intercomputer transfer. Note that if the format is specified as a two-word format, the next control word can be located easily without any further decoding. If, however, the format is a multi-word format, the next control word must be located by knowing the number of data words (found in the upper 15 bits of the first word) contained in the format. Use of the entire control word reduces data transfer efficiency. Up to 45 bits of data can be contained
in the two-word format, whereas only 30 bits of data can be contained in the first two words of a multi-word format.

Buffer Format

The message formats are superimposed on the intercomputer buffer format in chronological order. The output buffer format's first word contains the first and last address in which the data are to be put in the receiving computer (see Figure 6); next follow all the message formats. The last word is a unique end code to help determine whether errors have been encountered in control of the data transfer. This buffer format is thus variable in length, providing an efficiency in transmission time. Two buffer areas are provided in each computer for each transmission and one buffer area is provided for each reception. Thus, when an output buffer is active out of one area, the program may be packing the next buffer. In the case of “receive,” only one buffer area is required since (as will be discussed later) all processing of a previous “receive” buffer is completed before another receive buffer will be initiated.

BUFFER CONTROL

A major control problem encountered by the program is synchronization of the buffers in the communicating computers. This control problem arises: 1) when the intercomputer data buffers are initiated, and 2) when the buffers terminate. Between initiation and termination, the buffers are maintained in synchronization by the data control signals. The method of initiating the data transfer is the main key to establishing synchronization between com-

![Figure 6. Intercomputer Buffer Format.](From the collection of the Computer History Museum (www.computerhistory.org))
puters, because the buffer termination characteristics are defined within this initiation process.

The use of the Input Buffer Internal Interrupt by the receiving computer is predicated upon the receiving computer's knowledge of the length of the input buffer. The input buffer length must agree with the length of the output buffer from the transmitting computer. Therefore, with variable length buffers, the transmitting computer must define the input buffer length for the receiving computer.

The input buffer of the receiving computer is programmed to enter the first buffer word received into its Input Buffer Control Register. This is accomplished by executing an Initiate Input Buffer (With Monitor) instruction for the proper channel into the Input Buffer Control Register for the same channel. This instruction sets the initial address (the lower order 15 bits of the Input Buffer Control Register) the same as the address of the Input Buffer Control Register. The terminal address (the upper order 15 bits of the Input Buffer Control Register) must be different than the initial address for this method to work. Before the transmitting computer can initiate the output buffer, the receiving computer must have established an input buffer mode. The transmitting computer must determine this by testing the state of the Input Buffer Status Line with the External Function instruction.

When the receiving computer is ready to receive data (i.e., when all the specified conditions are met), the transmitting computer establishes the Input Buffer Control Word for the receiving computer. The first word of the output buffer data is the Input Buffer Control Word for the receiving computer. The lower order 15 bits define the initial address for data storage and the upper order 15 bits define the terminating address for data storage. Thus, the buffer area is defined for the receiving computer.

The output buffer mode is initiated by execution of the Initiate Output Buffer (With Monitor) instruction. As soon as this instruction is executed the first word of the output buffer data (the Input Buffer Control Word) is transmitted to the receiving computer. Data transfer proceeds under control of the Ready and Resume signals for the transmitting computer, and the Request and Acknowledge signals for the receiving computer. As soon as the transfer is completed each computer will be interrupted (as the result of the buffer monitor), allowing each computer to assume the opposite role—transmit or receive.

**PROGRAM LOGIC**

The intercomputer program logic is sufficiently flexible to allow operating any combination of multicomputer configurations. Three multicomputer configurations that have already been successfully operated are:

1) A three-computer system,
2) A two-computer system connected by one pair of cables, and
3) A two-computer system connected by two pairs of cables.

Refer to Figure 7 for illustration of these various configurations.

Other possible configurations for a three-computer system (connecting computers B and C) and systems utilizing four or more com-

---

**Figure 7. Multicomputer Complex.**
Computers could use the same program logic. A current restriction in the multicomputer system discussed here is that only two computers can be connected to any one computer.*

The discussion of program logic which follows will consider the simplest case—that of a two-computer system interconnected by one channel. Variations required for other configurations will be mentioned in the course of discussions.

Two subprograms are required for control in the two-computer configuration. These are A-B Intercomputer Control Subprogram (ABCON), located in the A Computer and the B-A Intercomputer Control Subprogram, (BACON), located in the B Computer.

In addition to the subprogram in each computer, there are three Internal Interrupt routines in each computer. They are the Input Buffer Monitor, Output Buffer Monitor, and Intercomputer Failure Interrupt routines. The routines are named by suffixing the subprogram name with MIN, MOUT, and FAIL, respectively. For example, the set of interrupt routines in the A Computer for the ABCON subprogram is ABCONMIN, ABCONMOUT, and ABCONFAIL (see Figure 8). In general, one subprogram and the three associated Internal Interrupt routines are required by each computer for each channel of intercomputer operation. In the three-computer configuration shown in Figure 7, Computer A would contain two subprograms (ABCON and ACCON) whereas Computers B and C would each contain one (BACON and CACON, respectively). In the two-computer system utilizing two-channel operation, each computer would contain two subprograms.

The program logic for the subprogram and three interrupt routines is now discussed for the A-to-B data transfer. Other subprograms and interrupt routines are identical except as noted.

*Since this paper was written, three-computer programs have been delivered connecting all computers to each other.

†The first letter in the code name indicates the computer in which the subprogram is located and the second letter indicates with which other computer the subprogram communicates.

The ABCONMIN program is initiated by termination of the intercomputer input buffer. In order to terminate the buffer the first word received must contain the proper input buffer limits as computed by the BACON subprogram. The ABCONMIN program checks the last word received (designated by the input buffer control register) to make sure it is a valid end code, thereby preventing an error in the control signals from causing erroneous data to be passed between computers. If the end code is correct, this program sets the ABCON flag in the Executive Flag Table.

The Output Buffer Monitor Interrupt in the A Computer indicates that data exchange from A Computer to B Computer is complete. The A-B Output Monitor routines (ABCONMOUT) thus entered initiates an input buffer (with monitor), into the buffer control register assuring that the residual state of the intercomputer channel is in the input mode, so that an output of the B Computer is accepted immediately when initiated.
The Intercomputer Failure Interrupt routine (ABCONFFAIL) is initiated when the data transfer to the B Computer has stopped for at least 32 seconds, but not more than 64 seconds (48-second average). This could occur if an input request to the B Computer were missed. The ABCONFAIL program alerts the A Computer program that a malfunction has occurred, allowing either automatic or manual remedial action to be performed, depending on the multicomputer configuration involved. For example, if both intercomputer channels are connected between the same two computers, entire data transfer is switched to the alternate channel. In any case, however, the proper information is given to allow rapid emergency maintenance whenever the faulty component is isolated within the system.

The A-B Intercomputer Control subprogram (ABCON) establishes control for the exchange of data between the A and B Computers in a multicomputer installation. This subprogram (ABCON) is flagged initially by the Input Buffer Monitor program (ABCONMIN) as described above. It is also reflagged by the EXEC on a 1-millisecond basis until the buffer is completely processed. The ABCON subprogram is referenced directly by the Executive routine in the A Computer. Operation of this subprogram is independent of other intercomputer subprograms. For example, ACCON would be contained in the A Computer in a three-computer complex.

The content of the input buffer contains both control and data information. Only the control data are processed by the ABCON subprogram. The first word of each intercomputer message format contains the control information.

The ABCON subprogram performs two basic functions: 1) processing of the data in the input buffer, and 2) control of the output buffer to the B Computer. The ABCON subprogram initiates output buffer control if all data have been processed from the input buffer.

If all data have not been processed, the ABCON program (see Figure 9) checks to determine whether the control word is valid. If the control word is not valid an error has occurred in the data bits on that intercomputer channel.

The recovery action, as described in the ABCONFAIL routine, can be used with the difference that the discovery of error has been made in the receiving rather than in the transmitting computer.

The next operation determines the data format type so that the data can be extracted from the control word (only for two-word format). Then the index to the next control word is computed. This index is stored to make the next control word available when processing the next format. The ABCON subprogram then turns control over to the processing subroutine which is specified by the format designator in the control word. This is one application of the indirect addressing feature of the Unit Computer. The processing subroutine executes its designated task and returns control to ABCON. ABCON returns control temporarily to the Executive routine to determine whether any higher priority subprograms are scheduled.

This is similar to flag setting mechanism number 4) described in Section II of this paper.

![Figure 9. A-B Intercomputer Control Subprogram (ABCON).](https://www.computerhistory.org)
If stringent timing requirements for higher priority tasks do not exist, all intercomputer messages could be processed before returning control to the Executive routine.

This cycle continues until all data formats have been processed. At this point the B Computer Input Buffer status is tested to determine whether the B Computer has established an input buffer. If not, control is again returned temporarily to the Executive routine. Normally this test is passed without delay and requirements for initiating an output buffer to the B Computer are begun. The unpacking index is reset for the next input buffer and the ABCON flag is cleared in the Executive Flag Table. The input buffer control word is packed in the first word of the current output buffer and the end code is packed as the last word. The output buffer control word is formed, the output buffer packing index is switched to the alternate buffer area, and an output buffer with monitor is initiated. Control is now returned to the EXEC until the next input buffer is received.

ILLUSTRATION OF OPERATION

In the discussion that follows, the method of control of the data exchange between the A and B Computers is described. The same process is used for controlling the A-C intercomputer data exchange in a three-computer system.

The computers are programmed to have equal control capabilities, but at any one time the computer that has initiated the Intercomputer Control subprogram is in control. The Intercomputer Control subprogram will retain control until all previously received inputs have been processed. When all inputs are processed, the computer in control initiates an output to the other computer. Control is relinquished to the other computer as soon as all the output data have been transferred. This type of reciprocating control is described as a “ping-pong” control system.

At the outset of system operation, the A Computer is selected to have initial control. The A Computer is forced into the output mode by an initialization subroutine. Data, transmitted from the A Computer to the B Computer in the first buffer exchange, would include the A Computer's real-time clock reading (in order to provide a common time base for the entire system) in addition to other initializing data.

The ABCON subprogram in Computer A tests the special input buffer status control line that indicates when the input mode has been selected in Computer B. Computer B, via the BACONMOUT program or the initialization subroutine, must have its Input Buffer With Monitor in such a manner that the first word received from Computer A automatically defines the area of Computer B's memory in which the succeeding data words are to be placed. When the input mode is detected, the ABCON subprogram in Computer A initiates an Output Buffer With Monitor. The first word always contains the first and last address of the input area in Computer B. Data transfer proceeds under control of the regulating signals which, in the buffer mode, operate independently of the program operation. The monitor logic in both computers compare, the current data word’s transfer address with the last address that should be transferred. When equal, each computer’s monitor logic interrupts normal program operation to allow immediate action by the respective computers.

Computer A, which was transmitting data, initiates the Output Buffer Monitor program (ABCONMOUT), which initiates an Input Buffer With Monitor into its buffer control register, and will remain inactive in the intercomputer transfer until Computer B initiates a transfer back to Computer A. In Computer B the monitor initiates the Input Buffer Monitor program (BACONMIN), which in turn initiates the BACON subprogram, thereby accepting the intercomputer control function from Computer A. The BACON subprogram will initiate the processing of all data, using the Control bits found in the first word of each format, to select the proper processing subroutine that is required. When all data are processed, an Output Buffer With Monitor is initiated to Computer A. This buffer consists of all output data accumulated while Computer B was in the input mode.
When this buffer from Computer B to Computer A is completed, the BACONMOUT program in Computer B initiates an input buffer to accept the next output buffer from Computer A. At the same time, the ABCONMIN program in Computer A flags the ABCON subprogram so that the inputs can be processed. When the inputs have been processed by ABCON, a complete round-trip intercomputer cycle is completed. The process repeats continuously, lending to the “ping-pong” effect in data transfer.

IV. CONCLUSIONS

Other systems, both commercial and military, have been and are being conceived which require more than one computer to solve the user’s problem. Some of these systems will connect two or more identical computers whereas other systems will connect a large scale central processor with smaller satellite computers. Consideration of multicomputer concepts in an original system design would allow for expansion of a single computer installation into a multicomputer installation as the problems to be solved increase or become more complex. Similar considerations would allow for greater flexibility in a multicomputer installation such that, depending on the problem or operational requirements, computers could operate separately on different problems or together to solve a common problem. The multicomputer programming techniques presented in this paper are usable as described or are adaptable to fit a particular system requirements.

The Unit Computer was designed specifically to facilitate extremely efficient intercomputer data transfer. However, such features as internal interrupts upon completion of a buffer transmission, although desirable, are not absolute requirements. The basic requirement is the ability to communicate directly with another computer. Since control signals associated with input and output to another computer differ from those associated with input/output to a peripheral device, these differences must be compensated for in the computer hardware.

The computer characteristic necessary to implement the real-time executive control philosophy is a real-time clock. This would be an internally stored, automatically maintained clock as in the Unit Computer, or an external clock capable of being referenced by the computer. The internal clock, repeat and compare instructions and external and internal interrupts included in the Unit Computer enable extremely efficient executive control; however, of these, only a real-time clock is an absolute requirement in the application of the Executive Control Philosophy to the control of a real-time system. If real-time is not a system requirement the real-time clock could be replaced by some other indexing mechanism under control of the program, thereby enabling use of the same priority logic to optimize executive control.

The Executive Control Philosophy presented in this paper is not restricted to multicomputer applications. The concepts are equally valid for a real-time system employing only one computer.

Several extensions of the techniques discussed will, no doubt, be obvious to the reader. Among these are dynamic changes in system task priorities as the operational requirements change during program operation. This merely means altering the order of the task list in the Executive stores. Included in this extension would be the ability to delete some tasks and add new tasks (subprograms) from a supplementary storage. This could be useful not only in a real-time command and control system when the nature of the battle may have changed but also in the scheduling of multiprogramming type problems in a commercial computing center. The fact that subprograms are designed to be independent of one another with any communication or control handled by means of the Executive routine and associated tables, makes the concept entirely feasible.

From a hardware standpoint, one modification considered is the ability of one computer to interrupt another computer. This characteristic has been incorporated in the later models of the Unit Computer. It has the advantage of allowing a computer to signal another computer that a buffer transmission is about to be initiated. This essentially eliminates the need for the Input Buffer Status signal and allows for more positive control of intercomputer data transfer.