Management Services: A Magazine of Planning, Systems, and Controls

Volume 5 | Number 2 Article 3

3-1968

New Generation EDP Control Considerations

Robert F. Moloney

The on line real time EDP system creates a whole new series of control problems. How is privacy ensured when a whole series of stations have access to the computer? What can be done about overloads on computer capacity? Here are some of the answers —

NEW GENERATION EDP CONTROL CONSIDERATIONS

by Robert F. Moloney
Xerox Corporation

FAST-RESPONSE data processing systems, generally referred to as "real time" or "on line systems," have made rapid progress over the past several years. Much of this progress can be attributed to the introduction of third generation computer equipment and supporting software capable of handling the complexities inherent in these systems.

The traditional control concepts, painstakingly built up over the past decade for batch-type operations, are no longer adequate. The use of communication facilities operating in an on line mode, duplex equipment configurations, multiprocessing, and multiprograming necessitates the development of additional control techniques if these systems are to process all data accurately and efficiently.

The purpose of this article is to discuss some of these new control requirements which systems analysts, programers, and auditors should be considering in the design of any real time system. These controls are considered in this article in four categories: on line controls, data protection controls, diagnostic controls, and emergency procedures.

On-line controls

The use of communication lines to transmit data in systems of this type requires the use of an on line terminal device by which messages are transmitted to or received from a computer. The most important consideration when operating in this mode is to ensure that the data being transmitted are received and

properly processed. It is always possible for messages to be lost or garbled or perhaps for a line or terminal device to go out of order during transmission. To prevent this from happening the system should provide program routines to check on messages sent through the system. These routines should provide for at least the following:

Message Identification Handling Procedures to ensure that each message is properly handled—Every message received at the data center should be identified by a message header showing such information as message number, terminal, date, and action code. This information is necessary for the initializer routine to route the message to the proper program for processing. If a message with an incorrect header is received, it should be routed to a control group for corrective action or rejected with a request to the originating terminal for retransmission of the entire message.

Message Transmission Controls to determine that all messages transmitted over the lines are received—This is done by assigning each message a number, usually within a block, and subsequently verifying the sequence of the message numbers received. Unaccounted-for numbers are considered exceptions to be investigated. Another control is the confirmation by the computer or terminal of all messages received.

One such system used a combination of these two methods. Each location was assigned sequential numbers 000 through 999. As an item was transmitted to the computer, it was assigned a message number, which was stored internally by the computer. Periodically a routine checking of these numbers would produce a report of out-of-sequence numbers and those remaining unused. These were forwarded to each originating location, which checked them against a log maintained for that purpose. Lost messages were retransmitted. Additionally, every message received by the computer was confirmed. This told the operator downline that it had been received by the processor and enabled him to verify the accuracy of the data.

Rerouting Procedures to handle messages sent to downline stations which are not operating—When this happens it is necessary to provide routines to reroute the message to another terminal or store the message internally in the computer until such time as the down terminal restarts operation. The procedures for dealing with this type of operation are sometimes called "willful intercept." (The action to be taken when the computer breaks down is not discussed here since this topic will be covered under the category of emergency conditions.)

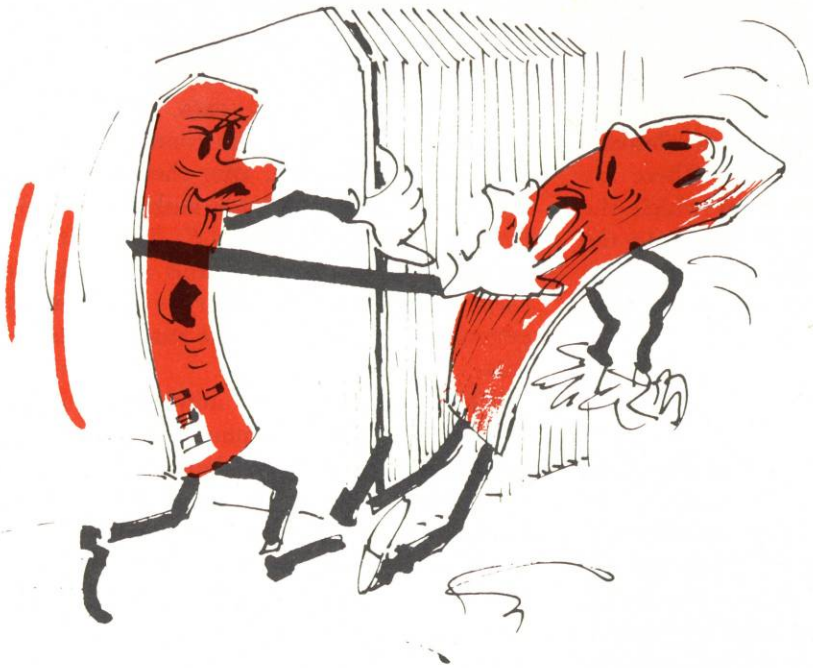
Check Characters to detect transmission errors—Two data verification procedures generally used to check the accuracy of transmitted data are the character and message parity checks. The character parity check, with which most accountants are familiar, verifies the accuracy of each character transmitted. The message parity check is a check digit compiled at the originating terminal, based upon the number of bits in the message sent, which is tacked on to the end of that message. The receiving terminal, similarly, compiles a check digit based on the number of bits received, and both check digits are compared. If they agree, a "transmission correctly received" signal is sent to the terminal or processor. If the communication lines have introduced errors, the system should provide for an alert signal to the terminal notifying it of the error or, preferably, for the automatic retransmission of the entire message.[1] In the latter case, if the error condition still exists the terminal should be alerted.

The system can verify that all messages transmitted have been received by giving each message a number; the computer checks the sequence to ensure that all are in.

Data protection controls

The on line capability inherent in a real time system creates a problem of data protection. Information stored in the files may be made available to terminal operators at their request. This raises the question as to what procedures are necessary to prevent data from

[1] For a more thorough discussion of this topic refer to IBM Corporation, "Message Control System Concepts," Reference C20-1609, p. 5.



Unless some provision is made to ensure that only one transaction can update a file at a time, file changes created by transaction A will be lost when transaction B, occurring simultaneously, is recorded.

being accidentally destroyed or used by unauthorized personnel. The question is especially critical when the information involved is highly confidential or the system operates in a time sharing mode. The problem becomes even more complex in a multiprogramed system: What assurance is there that program segments read into core storage will not be accidentally loaded over data currently being processed?

For purposes of this discussion data protection includes those measures incorporated in the system to prevent concurrent updating of stored files, unauthorized use of stored files, and accidental destruction of data in core storage during processing by the computer. Applicable control considerations are discussed in the following three sections.

Concurrent Updating—The normal types of errors which can occur during the transmission of data over the communication lines do not pose serious problems as the majority of these can be discovered through the use of a strong edit routine and the incorporation of the on line controls previously discussed. But what about the problem of concurrent updating, as when two transactions in a multiprogramed system attempt to update the same file simultaneously. For instance, transactions A and B retrieve the same file and update it. If the updated version of the file resulting from transaction A is stored first, it will be lost when transaction B is stored. What is needed is a procedure which permits only one transaction to update a file at a time. In the IBM system software package this is referred to as "exclusive control."

"Exclusive control" can be achieved by requiring each transaction to "request" permission of the supervisory program to update a file. If the file is available, the supervisory program grants the request and the transaction updates it. During this time no other transaction is permitted access to this file.

Data Security—The prevention of unauthorized access to stored data can generally be accomplished by the use of lockwords, authority lists, and dedicated communication lines. Lockwords, sometimes referred to as "keywords" or "passwords," consist of several characters in a data file which the input transaction or inquiry must match in order to gain access to the file. The use of this device to control file references may be further refined by supplying several lockwords. For instance, one set of characters may permit the file to be retrieved for reading purposes only (read-protect), and still

ROBERT F. MOLONEY, CPA, an accounting specialist at Xerox Corporation, has served as a staff assistant—EDP at Union Carbide Corporation and manager, budgets/cost at American Airlines. A graduate of Niagara University, he received a certificate in EDP and system analysis from New York University. He is a member of the Data Processing Management Association, the Institute of Internal Auditors, and the New York State Society of CPAs.

another set may permit both reading and writing.

When using lockwords, consideration should be given to the type of terminal used. If the lockword appears on each document printed by the terminal, it may defeat its own purpose since it becomes relatively easy to compromise it. There are devices available from which the lockword can be entered into the system in a non-print mode. This type of terminal should be used where feasible.

Authority lists

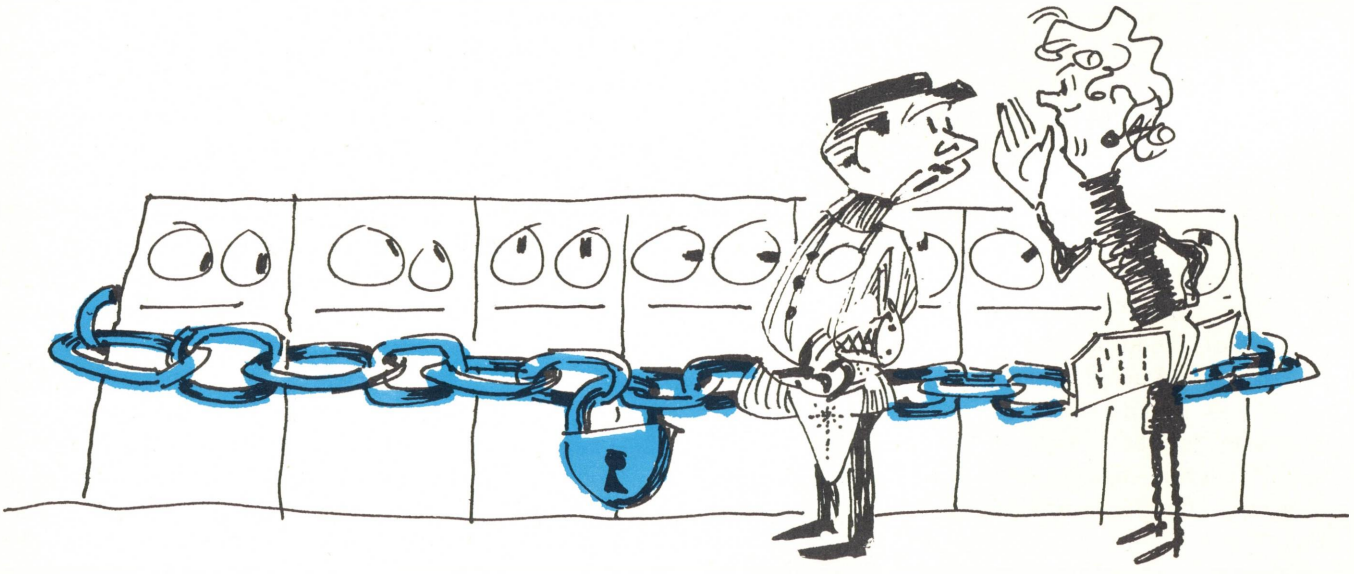
Authority lists are another form of protection. In this instance, the lockword is used to identify the person transmitting from the remote location. After the initial identification has been established, reference is made to an authority list which indicates which type of data the sender is authorized to receive. As with lockwords, the authority list may classify references to the files as read only or both read and write.

The previous discussion regarding the use of these two techniques does not exclude the use of other approaches to accomplish the same ends. For instance, "if an inquiry is received from a remote station. . . . . asking for confidential data, the computer might break the connection and then redial it, after checking to see that the station is authorized to receive the data.[2]" The extent to which these controls are used will depend on the type of system and the nature of the stored data.

When lockwords and authority lists are not used, terminals may be identified by means of dedicated communication lines. This generally implies the use of only one terminal on the line. Where terminals are attached to a party line or a similar network, answerback would be used in lieu of a dedicated communication line. With answerback a signal is sent to the terminal, and the latter responds within a prescribed amount of time with its code identification. Because this system can be compromised, it is not recommended where the security of stored data is of major importance.

The first two methods discussed should successfully exclude unauthorized personnel from the stored data files. However, even the best coding system can be broken if an individual can work on it. What procedure, then, can be incorporated in a system of this nature to prevent an unauthorized individual from transmitting data to the processor in an attempt to decode the lockword and gain access to the system? A monitoring routine could be established which would count the number of unsuccessful attempts to enter the system and after a certain number had been reached, say, three times in succession, a message might be printed out at the data station with instructions to call the downline station for an explanation. The point to be made here is this: If the data are important enough to require the

[2] Corning Publications, Inc., *EDP Analyzer*, March, 1966, Vol. 4, No. 3, p. 12.



"Lockwords" can be the key to preventing file compromise. One set of characters entered through the transmitter will permit the file to be retrieved for reading purposes only; another more restricted group of characters can be used for both reading and writing into the file.

use of lockwords and authority lists in the system, it is important to ensure that these controls are effectively accomplishing their assignments.

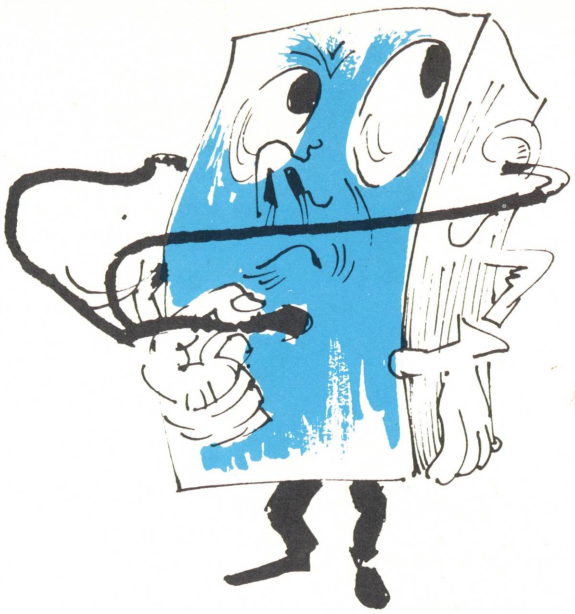
Boundary registers

Memory Protection—In a multiprogramed system, a number of different data elements will be in core storage at the same time. These will consist of the control program, a portion of which will permanently reside in core; a number of operational programs; and various queues of messages awaiting processing or in the process of being handled by the computer. Because of the possibility of a programing or a machine error, an operational program could address portions of core storage outside the limits of its own coding, work areas, or other applicable areas. The result could be the alteration or modification of another program; the destruction of data on tape, disk, or drum; or perhaps the creation of a series of endless loops. To keep this from happening the system should provide for some form of memory protection.

One technique currently employed is the use of boundary registers. This requires additional equipment in the form of an upper and lower boundary register. In the simplest form of this system the boundary registers are loaded with the upper and lower core storage addresses of the program when it is loaded into the processor. If during the course of the program the address portion of the instruction exceeds the boundaries indicated in the registers, an interrupt occurs, and control is passed to the supervisor program for appropriate action.[3]

The IBM and the Spectra 70 use storage and protection keys. Core storage in these systems is broken up into blocks. Each block has a four-bit storage key assigned to it. The program status word for each program has a four-bit protection key. Before any program references a block of core storage for the purpose of writing data, the storage key and the protection key must be matched; otherwise the operation is aborted, and control is passed to the supervisory program.

It is easy to see that there are a number of different approaches, depending upon the equipment used, to provide memory protection. Since this topic has been thoroughly discussed in various EDP publications,[4] it will not be discussed further in this article.



One of the characteristics of real time equipment is that it should keep operating without stopping if there is a malfunction somewhere in the system. This means that the machine must itself identify the trouble and either cure it or switch to another routine until the problem has been remedied.

Diagnostic controls

One of the characteristics of a real time system is that it must operate without stopping for fixed periods of time. The duration of time will depend upon the equipment configuration. A duplexed system can be designed to operate 24 hours a day, whereas a simplex system operates for only part of the day. Suppose, however, that there is a malfunction of some component part or that a programing error occurs while the system is operating. Should the processing halt, as was usually the case with second generation equipment, or should it continue processing after branching to some routine to handle the problem? The general rule is to keep the system operating if there is some way of circumventing the trouble. To accomplish this objective, it is necessary to build into the system some way of detecting and isolating error conditions so that appropriate action can be taken. This can be done through the use of diagnostic programs.

"Diagnostic programs are a tool used to test computers, isolate component malfunctions, and improve overall computer system operations."[5] We are interested only in those on line routines which detect the fact that errors are happening

[3] For a more thorough discussion of this topic refer to William H. Desmonde's "Real-Time Data Processing Systems, Introductory Concepts," Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1964, Chapter IV.

[4] See especially Corning Publications, Inc., op. cit., pp. 10-12.

[5] James Martin, *Programming Real Time Systems*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1965, p. 125.

and isolate, where possible, the cause of the error.[6] Once the error has been isolated it is up to the supervisory program to determine what action is necessary. According to Martin,[7] one of six actions can be taken:

1. Re-execute the faulty instruction and continue processing.
2. Restart the program in question.
3. Transfer to an exception routine characteristic of the program in question.
4. Initiate switchover.
5. Initiate closedown.
6. Halt.

Each of these alternatives will be discussed in the section on emergency procedures which succeeds this section.

The number and types of diagnostic programs in a particular system will depend on the design of the system and its equipment configuration.

For purposes of illustration, the following is an example of how these programs might work:

Consider a real time system communicating with a number of remote terminals. Suddenly a terminal breaks down. When this happens, a diagnostic program checks the communication network and establishes that there is a problem. Another diagnostic program checks each line until the down terminal is isolated. Once the error has been isolated, control is returned to the supervisory program, which might close down this line until repairs are made and route all interim messages to an adjoining terminal for manual handling.

It is always possible, of course, that errors may occur which will not be detected by such routines. The only protection against invalid data created by such errors is to have adequate file reconstruction procedures.

When a system is able to switch its operation from one computer an another, as is possible in a duplexed system, without changing its method of processing data, we say a "switchover" has occurred. On the other hand, if any equipment malfunction occurs in the system which requires the system to close down some part of the operation and modify its method of processing data to circumvent the error, we say it is in a "fallback" mode.

Emergency procedures

Once an error or malfunction has been isolated, it is up to the supervisory program to take appropriate action. As was stated previously, it has a number of choices. The first three of these—re-execute the faulty instruction, restart the program, or transfer control to an exception routine—do not present any unique problems. They have been satisfactorily tested in the traditional batch processing systems, and similar procedures can be incorporated into real time systems. The last three alternatives—initiate switchover, close down part of the system, or halt—are unique problems and require careful consideration if the system is to provide for all contingencies.

Failures that can cause a system to switch computers or close down a part of the system are generally due to hardware malfunctions. When a system is able to switch its operation from one computer to another, as is possible in a duplexed system, without changing its method of processing data, we say a "switchover" has occurred. On the other hand, if an equipment malfunction occurs in the system which requires the system to close down some part of the operation and modify its method of processing data to circumvent the error, we say it is functioning in a "fallback" mode. Whenever either of these conditions arises it is necessary to provide procedures to ensure that they are efficiently handled.

Switchover, as stated previously, assumes the use of duplex computers so that if the operating computer breaks down its supporting unit will take over processing. When the change from one unit to another takes place, whether it is automatic or manual, the machine operator should be informed via the console or printer what action he is required to take, if any, and

[6] Although there will be a number of off line routines peculiar to real time systems, we are not interested in them for purposes of this discussion since they do not affect the system while it is operating and are generally used only by the systems engineer.

[7] Martin, op. cit., p. 224.

the reason why the change was made. For an automatic switchover to take place, the on line computer should initiate action. However, if the malfunction is serious, this may not always be possible. To ensure that a changeover is effected when such a malfunction occurs, the standby computer should periodically check its counterpart. If it detects a malfunction, it should initiate the switchover. Whenever this transition takes place, a message notifying terminal operators of this fact should be sent downline.

During the transition it is vitally important that transactions not be lost. Messages that have not actually entered the system present no problems since these will be noted as missing when the sequence check (discussed under on line controls) is made. The problem lies with those that have entered the system and are awaiting processing on input, work, or output queues. This problem may be resolved by recording the message sequence number after a record has been updated rather than as each message enters the system, or perhaps both actions might be recorded and subsequently compared to each other. Another method is to post the message sequence number to the file as it is updated. In the event of a switchover the comparison of the transaction and file message number will prevent a file from being updated twice. It may or may not prevent messages from being lost depending on the system design.

Graceful degradation

Fallback or "graceful degradation," as it is sometimes called, occurs in a non-duplexed system when a part of the equipment configuration breaks down but the loss of the particular piece(s) is not serious enough to shut down the entire system. When this happens, the machine operator should be informed by the control program as to the current status of the system and what action he should take. In some instances terminal operators should also be informed. Procedures should be available to advise supervisory personnel what clerical action is necessary to support the system until it recovers and, finally, what action is necessary to restore the system to the condition that existed before the fallback occurred.

The results may be catastrophic when a real time system halts. If the halt is due to a complete breakdown of a major component the only thing to do is repair it as rapidly as possible. Procedures should be available for supervisors so that they may take necessary emergency action and guide clerical personnel in work which must be done while the system is down and initially after it recovers.

It is imperative that restart procedures be incorporated in the system. The restart is based on a complete checkpoint record written on a peripheral device such as a disk file. The checkpoint record, provision for which should be incorporated in the system, is a complete record of all messages, counters, logs, and status indicators in the system at that time. When a restart is necessary, the checkpoint record is used to restore the system to its condition at the time the checkpoint record was written. Each terminal is advised of the restart and the number of the last message properly received from the terminal at the time of the checkpoint. Subsequent messages are retransmitted, and the system is again operational.

If the halt occurs because of a system overload the problem is not as serious. System overloads can occur when a number of messages are read into core and subsequently it is discovered that there are not enough available core blocks to complete all the work the computer has started. This problem can occur in a multiprogramed system using random input. There must be some emergency procedures in the system to handle this dilemma.

It is possible to prevent this from happening, except in rare instances, by anticipating when the level of core blocks available for processing

Fallback or "graceful degradation," as it is sometimes called, occurs in a non-duplexed system when a part of the equipment configuration breaks down but the loss of the particular piece(s) is not serious enough to shut down the entire system.

When the level of core blocks in the computer available for processing data is reaching a danger point, the computer should shut down to the extent of refusing any additional input.

data in the computer is reaching a danger point. When this level has been reached, the computer should shut down and refuse to accept any more input. This means the computer must be able to control the volume to input during peak periods. This control is exercised in a system utilizing "polling" techniques by not "polling" the transmitting locations until the overload is ended. Another method that may be used is for the processor to send a signal either requesting the operator to re-enter the message into the system or locking the terminal pending further notice from the computer.

If the overload does occur, however, the system should be able to handle the problem by first determining the application programs temporarily in core which are not currently being used and making their applicable blocks available for the further processing of data. If this doesn't solve the problem, the system may have to destroy messages in the system, preferably on a last in basis, and request the applicable terminals to repeat the messages.

Conclusion

Real time systems are here to stay. If systems analysts, programers, and auditors are to design and review these complex applications, it is necessary that they understand what a real time system is and, more important, the controls necessary to ensure that the desired results are produced. An understanding of the traditional controls is not sufficient when operating in a real time environment. Although these controls, such as matching and batching or validity and limit checks, are just as necessary to the effective operation of a real time system as to a batch system, the nature of the real time system has necessitated additional controls to ensure that the system operates as desired.

This article discusses many of those control considerations. It is not intended as an all-inclusive statement of control requirements in real time applications, since these will be dictated by such factors as system design specifications, the equipment used, security requirements, and many other factors. It merely summarizes some of the more common control considerations basic to a well designed real time system.