Woman C.P.A.

Volume 37 | Issue 2 Article 5

4-1975

Electronic Data Processing: Implications of Real-Time Systems for Accounting Records, Part 2

Elise G. Jancura

Follow this and additional works at: https://egrove.olemiss.edu/wcpa

 Part of the Accounting Commons, and the Women's Studies Commons

Recommended Citation

Jancura, Elise G. (1975) "Electronic Data Processing: Implications of Real-Time Systems for Accounting Records, Part 2," *Woman C.P.A.*: Vol. 37 : Iss. 2 , Article 5.
Available at: https://egrove.olemiss.edu/wcpa/vol37/iss2/5

This Article is brought to you for free and open access by the Archival Digital Accounting Collection at eGrove. It has been accepted for inclusion in Woman C.P.A. by an authorized editor of eGrove. For more information, please contact egrove@olemiss.edu.

Electronic Data Processing

Implications of Real-Time Systems for Accounting Records

Part 2

Dr. Elise G. Jancura, CPA
The Cleveland State University
Cleveland, Ohio

Technical Characteristics of Real-Time Systems

The basic nature of a real-time system requires that a transaction be processed immediately as it occurs. This leads to systems that must be able to accommodate randomly occurring peak loads in an acceptable period of time. Some technique must be developed to handle and queue these transactions from various input sources without loss of any of the transactions. Further, some technique must be developed to identify and recognize priorities within these transactions, for if the real-time system is in fact controlling a physical process there may be a need for recognizing during a peak load the necessity of handling a transaction on some basis other than first-in, first-out. The need to handle simultaneous input from many parts of the system and to accommodate peak loads frequently results in a total system capacity that is not fully exploited by the real-time demands except during the peak-load periods. As a result, many real-time systems combine the processing of real-time applications with batch-processing applications, which occupy the system when the demands of the real-time transactions permit. This means that the control programming on the computer system must be able to accommodate more than one application program at a time and leads to the programming complexities present in a multiprogramming environment.

Multiprogramming is a mode of operation in a computer system in which more than one task is being processed concurrently. The internal control and scheduling problems in a multiprogramming system are quite complex but not peculiar to real-time systems. Many batch-mode systems, with or without remote input, operate in a multiprogramming environment. Multiprogramming introduces the need for the auditor to identify the actual program used in processing a given set of application data.

A multiprogramming environment introduces the possibility of concurrent updating of a master record. The control problem occurs in those instances where two or more programs access the same master record and simultaneously try to update it without benefit of the updating done by the other program or programs. The solution to the problem of concurrent updating is to prevent access to a given record by a second program until the first program has replaced its updated version in the master file. This can be accomplished by organizing access to the files through a file management, or supervisory, program that requires each application program to request permission of the supervisory program before accessing and updating the record involved. This is sometimes referred to as exclusive control. Because real-time systems involve simultaneous input from many parts of the processing system, the potential for attempted concurrent updating of a master file always exists. The problem of concurrent updating is not exclusive to real-time systems but exists in any kind of multiprogramming environment. It is, however, an additional consideration that must be taken into account in real-time systems.

In addition to a multiprogramming environment, many real-time systems operate in a multiprocessing environment. Multiprocessing refers to a situation in which more than one central processing unit operates in parallel. Multiprocessing can occur either with central processing units that are similar in nature or with central processing units that differ from each other. For example, in one configuration a small processing unit may schedule communication lines and then transmit the collected data to a larger processing unit. In another environment two processors of equal size and nature may share common memory or communication networks. The characteristics of heterogeneous input requiring the availability of many different kinds of processing programs, of a multiprogramming environment requiring the same variety of programs, and of a multiprocessing environment all dictate the need for a fairly complex operating control that can select processing programs and load them into core as indicated by the appearance of a particular form of input and that can coordinate the access to data files from multiple programs within a single central processing unit or from multiple programs within multiple central processing units.

The potential complexity of these kinds of systems requires a great deal of knowledge and technical sophistication on the part of the auditor and on the part of the installation as it designs its application programs and makes provision for control of its processing system. The implications for the auditor in testing the system are

significant, for the physical and economic impact of shutting down one of these complexes to test an individual program on a stand-alone basis is such that it is not a feasible approach to take. In order to use the client's processing facilities in this kind of an environment, the auditor must have an adequate knowledge of the operating system and the characteristics of the system, so that the audit programs can execute concurrently with continued client processing. If the system shuts down during the night shift, the auditor can have access to the system on a stand-alone basis. But even in those circumstances, the auditor may still find it useful to use the client's operating system, thus requiring a knowledge of the job control language of that operating system.

Most advanced data processing systems and real-time systems maintain their master data in integrated files. The concept of file integration results in the combination of the data records for several different functions with similar information into single comprehensive sets of records. This process of creating single comprehensive records and thus a single comprehensive file minimizes the necessity for duplicate operations and duplicate records. These integrated sets of data records, frequently called a data base, become the master file for a number of different applications. The processing for an integrated data system or data base is characterized by the fact that a single source document describing a transaction is used to initiate the updating of all records associated with that transaction and affected by it. Although this results in an elimination of duplicate data within the master files and more efficient handling of all the facets of the transaction, it places a very heavy responsibility on the installation for maintenance of that single data base. Under this approach all of the pertinent master information and historical data is contained within one single master file or data base, and erroneous processing or inadvertent destruction of that single data base can have more serious implications for an organization than the destruction of a single master file that is only one of several master files for a firm. Integrated data systems need not be on a real-time basis, but many are so organized.

System Recovery Procedures

By definition, a real-time system is one that must be available for processing whenever transactions occur. Failures tend to be much more critical in a real-time system, because the computer is interacting with and to a large extent controlling its environment. Under these circumstances the need for system reliability and immediate, or at least timely, recovery becomes critical. At the same time, real-time systems represent much more complex combinations of hardware and programming, which create a difficult process for restarting the system. In addition, the fact that most real-time systems involve large data bases that usually are updated in a destructive mode introduces the requirement for determining the status of the data base and protecting its integrity during the down-time and restart procedures.

System restart in a real-time environment can rarely be accomplished merely by reloading the programs, remounting the data files, and restarting the system at the beginning. It usually requires instead the execution of specially prepared programming routines that are designed to search out the data files to determine the exact status of the processing previous to the error and to reposition all of the various elements of the system. Sometimes the most difficult part of a restart procedure is determining exactly which part of the system has failed and the status of all of the concurrent activities occurring within the system. The approach taken to recover from an error condition will vary depending upon the cause of the error. Thus, for a large real-time system with many remotely located terminals and communication lines, the procedure for an error in one of the terminals varies from the procedures for errors within the central processing unit or in the device containing the master data files.

Provision for reliability in a real-time system includes more than just restart and recovery procedures. It is equally important for the installation to include those physical and programming procedures necessary to test the system continually and to detect errors as they occur. Many errors (some caused by operator error, some caused by programming failure, and some caused by hardware failure) do not create system conditions that prevent continued operation of the system and therefore are not always readily apparent as they occur. Failure to detect these errors can result in data files becoming interlaced with erroneous information that is not apparent until some later time. Failure to detect errors promptly can also cause loss of background information that would have been helpful in diagnosing the cause of the error condition.

Despite the best planning, emergency situations will occur that have destructive effects on the data base. Interruption of processing in the middle of an update operation can introduce errors into the record currently being processed. Other error conditions, such as malfunction within the input-output device containing the data file (for example, a head crash in the direct-access file), can very effectively destroy the data stored there. Thus any recovery procedure must provide for verification of the accuracy of the data base and for reconstruction of any file information adversely affected by a systems failure. A critical requirement of an effective recovery system for a data base is the necessity to keep track of the consequences of every updating transaction. This is particularly important in the integrated file system, where one transaction affects several logically related records. It is not enough to know just which transaction was being processed at the time of the failure but the full impact that that transaction has had to the point of failure on the master records. A technique must also be developed to notify the user initiating a transaction (usually at a remote terminal) of the results of that transaction, so that the user is aware, in the case of a systems failure, whether that transaction has been processed or not and therefore whether it should be retransmitted or not.

One approach to reconstruction of the data base is to maintain a dual recording of the file. Updating transactions are used simultaneously to update both files. Error conditions caused by an input-output device (for example, either the disk file or drum) can be handled by this method, for the second device is unlikely to have a similar malfunction at the same time. Under these circumstances the recovery procedure would be simply to use the remaining good copy to duplicate the corrected data file. This approach has a weakness, however, for error conditions within other components of the system, such as failure of a central processing unit, an environmental failure, or an unexplained error in the application program, will produce the same erroneous condition in both copies of the data file.

Another approach, and the one most commonly used, is periodically to dump the data file onto another device, on a regularly scheduled basis. The most frequent technique is to dump a file contained on a disk to a tape file. This approach requires that all of the transactions that have occurred since the last dumping operation also be saved. In the event of a failure, all of these transactions must be reprocessed. To minimize the reconstruction time the dumping can be done on a

(Continued on page 29)

EDP
(Continued from page 15)

more frequent basis, so that the intervening processing period for which transactions must be saved can be kept to a minimum. This approach to reconstruction has the disadvantage that the dumping operation itself requires time and, of course, that the reconstruction operation can be time-consuming, since it requires a reexecution of all of the transactions occurring since the preceding dump. Where possible, it is desirable when using the dumping technique to incorporate in the dump operation additional processing benefits. Thus, while dumping records, an edit routine can be executed to perform edit and reasonableness checks on the logical fields within each record in order to determine, where possible, consistency within the records. In addition, inactive or logically deleted records can be recognized, so that file rearrangement and compaction can be obtained as a by-product of the normal dumping procedure.

The third approach, which can be used to good advantage in those instances where a systems failure destroys only a few records rather than the entire data base, is an approach that can be referred to as an audit trail approach. Basically this technique keeps a record of all transactions that occur as well as the contents of each master record both before and after updating by a specific transaction. The audit trail log, or reconstruction log, can be recorded on any medium although the most efficient is some machine-readable medium that can be accessible to the recovery routine. By copying the contents of the data base master record before updating, the full text of the transaction, and then the contents of the data base record after updating, the reconstruction log makes it possible for the recovery routine to determine all transactions that were in the process of updating when a failure occurred. The records containing information regarding the contents of the data base records allow restoration of any records involved in the failure. Serious failures, in which the full file is destroyed, are still best handled by the latest file dump. Then the reconstruction log can be used to merge in the after copy of those master records that have been updated by transactions since the last dump.

Whichever recovery procedures an installation decides upon, it is essential that those procedures be carefully planned ahead of time and just as carefully documented. This means that the installation must investigate very thoroughly all potential sources of systems failure and consciously attempt to provide for recovery from each individual type of potential failure. In addition, it is important that operating personnel be made completely familiar with the proposed recovery routines and be carefully trained in the implications of those activities. Recovery plans should be periodically reviewed in order to determine that they are as applicable as when first designed. As part of the evaluation of the processing system, the auditor verifies that the client has made adequate provision to protect the data base and to provide for recovery routines that will facilitate continued operations. This is particularly true in those instances where the operations literally depend upon the successful functioning of a real-time system.

Tax Forum
(Continued from page 25)

3. Consideration should also be given to amending the governing instruments to provide that the members may vote each year to determine if the assessments in excess of disbursements for the year (excluding the separate fund discussed above) should be refunded to the members or applied against the assessments for the future year. The Internal Revenue Service has ruled in Revenue Ruling 70-604 that if the above provisions are present the excess of receipts over disbursements for current maintenance may be excluded from taxable income.

Remedial legislation for homeowners' associations is being considered by the House Ways and Means Committee, but even if such legislation is enacted it may not be effective for prior years.

In conclusion, it appears that until homeowners' associations are given specific exempt status under the Internal Revenue Code, steps should be taken to avoid future taxation on the accumulation of funds and to minimize or eliminate potential deficiencies for prior years.

The Search Is On For The

MANAGEMENT WOMAN . . .

YOU are wanted NOW in finance, controllership, banking, taxation, auditing, compliance and consolidations to name a few specific areas. If you have first-class qualifications with sound experience in your field, are results-oriented and want to climb the corporate ladder, then send us your resume in strictest confidence. Interviews by appointment only.

MANAGEMENT WOMAN, INC.
Suite 549/The Plaza at 59th Street
New York, New York 10019
Telephone: (212) 751-9290