REPORT NO. DOT-TSC-FAA-71-16

REFERENCE USE ONLY

SYSTEM RELIABILITY AND RECOVERY

RETURN TO
DOT/TSC
TECHNICAL INFORMATION CENTER

CHARLES A. DANCY
TRANSPORTATION SYSTEMS CENTER
55 BROADWAY
CAMBRIDGE,MA. 02142

JUNE 1971
TECHNICAL REPORT

Availability is Unlimited. Document may be Released To the National Technical Information Service, Springfield, Virginia 22151, for Sale to the Public.

Prepared for

FEDERAL AVIATION ADMINISTRATION
WASHINGTON,D.C. 20546

TECHNICAL REPORT STANDARD TITLE PAGE

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
DOT-TSC-FAA-71-16		

4. Title and Subtitle	5. Report Date
System Reliability and Recovery	June 15, 1971
	6. Performing Organization Code: TCC

7. Author(s)	8. Performing Organization Report No.
Charles A. Dancy, III	

9. Performing Organization Name and Address	10. Work Unit No.
DOT/Transportation Systems Center Computer Technology Division/TCC 55 Broadway Cambridge, MA 02142	FA-03
	11. Contract or Grant No.

12. Sponsoring Agency Name and Address	13. Type of Report and Period Covered
FAA-SRDS-RD 123 800 Independence Ave., S.W. Washington, D.C. 20546	Technical Report
	14. Sponsoring Agency Code: FAA RD 123

15. Supplementary Notes

16. Abstract

This study exhibits a variety of reliability techniques applicable to future ATC data processing systems. Presently envisioned schemes for error detection, error interrupt and error analysis are considered, along with methods of retry, reconfiguration, task rescheduling and system restart. Reliability data are accumulated on present and planned ATC data processing systems and on certain commercial, military, and experimental computers having features applicable to future ATC tasks.

Included as well are discussions of reliability concepts, methods of reliability determination and criteria for judging system reliability and capability for recovery.

This work is connected with FA-03-1, Large Scale Systems.

17. Key Words	18. Distribution Statement
Reliability, recovery, reconfiguration, multiprocessor, failsafe/soft	Unclassified - Unlimited

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	61	

Form DOT F 1700.7 (8-69)

ACKNOWLEDGEMENTS

The author acknowledges the contributions of Mrs. Judith Gertler to the discussion of software reliability found in Sections 4 and 5. He also appreciates assistance and data supplied by the NAFEC Test and Evaluation Division and by the commercial computer companies mentioned in this work.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

ADVANCED ATC COMPUTER SYSTEM RELIABILITY AND RECOVERY

1. INTRODUCTION

1.1 BACKGROUND

With the automation of the air traffic controller function in the next decade, high reliance will be placed on realtime computers. The role of the human controller will become a supervisory one, i.e., checking computer results. The load handled by the automated system will often exceed that capable of being handled manually. Thus the system must not be allowed to fail in ways that place unmanageable overloads on human controllers. The effects of failures must rather be controlled so that in the worst cases the system degrades slowly enough to permit loading to be scheduled downward or handed off to adjacent facilities. Clearly the system must be designed not only for high intrinsic reliability but also for fast recovery from failure and for high failure tolerance.

Early vacuum tube computers were plagued by faulty memory media and processing hardware. The mean time-to-failure was on the order of minutes. Programmers wrote their programs so that they could reconstruct what went wrong with the aid of elaborate built-in software debugging aids. They knew they had only a few minutes on a computer per day and had to capitalize on snap-shots of computer results. With frequent computer break downs it took weeks to finish a program.

The MIT computers such as Whirlwind and the Memory Test Computer, which were vacuum tube computers, had hardware marginal checking features. During off-hours the filament and screen-grid voltages of the vacuum tubes were reduced and computer test programs were run. This preventive maintenance eliminated many catastrophic failures during normal running hours. Paper tape inputs were sum checked and the ferrite core memories had parity bits for checking memory transfers. However, it was discovered then that the ferrite core memory was many times more reliable than the rest of the computer. Thus, when the Sage Computer was built, the processor was duplicated but not the memory. Whenever test routines determined a malfunction in one processor the other processor of the duplex configuration was manually switched, the memory remaining unchanged.

Memory parity checking has continued to this day. The IBM 370/165 recently revealed error checking and correction in its processor which corrects single bit processor storage errors and detects all double and some multiple bit errors. It is expected that computers in the late seventies will become even more reliable, with less emphasis on further increases in processor speed.

The NAS Stage-A (IBM-9020) computers and ARTS III (UNIVAC-1230) computers are configured in fail safe/soft configurations. When a processor is malfunctioning, a standby replacement is automatically substituted and runs in a degraded mode when not enough replacements are available. There is not much evidence on the effectiveness of this scheme. The system reliability drops off sharply whenever 250 or more targets are being processed and targets are actually lost at these heavy traffic rates, which is a fault of the software. Since the NAS Stage-A is not yet fully operational and there still exist software difficulties, there has been little experimentation to determine its reliability. A reliability check on the IBM-9020 was conducted at NAFEC in 1967 (see p. 10).

Certain general principles of system reliability seem clear. Obviously a system which is built out of highly reliable parts is preferable to an identical one which uses less reliable hardware. Similarly, a system with fewer parts will have fewer failures than one with a greater number of equivalent parts; this indicates that systems which are more general-purpose (without actual redundancy) are more prone to malfunction than special-purpose systems which do not have unnecessary functions. Systems with purposeful redundant elements or error correction features will operate even though certain faults have occurred.

Often computer simulation programs can be used to verify improvements in system reliability. In some instances pilot experiments are required; in other cases the full-blown system has to be tried.

1.2 SUMMARY

In Section 2, System Reliability Concepts, this report presents definitions of terms used in system reliability work (and specifically computer system reliability work) and how the concepts to which these terms refer relate to each other to produce highly available and highly reliable systems. A future reliability requirement for ATC systems is specified on the basis of current ATC system reliability and capability.

In Section 3, System Organization and Reliability, the features of several computer architectures are analyzed in terms of how they contribute to the system availability. Examples of parallel or array processors, associative processors, multiprocessors, and advanced uniprocessors were the architectures analyzed. Specific reliability figures are given and sample analysis presented for some of the architectures.

In Section 4, Error Detection and Error Handling, specific methods of hardware and software error detection and error handling are discussed. The value of error detection in an ATC system is shown to be related to the difference in "severity" between an undetected and detected error.

In Section 5, Recovery and Reconfiguration, specific methods used to aid in recovery and reconfiguration for various computer systems are discussed. A scheme showing the use of an associative memory for resource allocation and error recovery in a multiprocessor system is given.

2. SYSTEM RELIABILITY CONCEPTS

2.1 DEFINITIONS

Reliability is defined as the probability that a system will perform as required for a given period of time.

Availability is defined as the percentage of time that a system can perform as required.

The term "perform as required" is crucial; the reliability or availability of a system will vary if the performance requirements are varied.

Recovery is the ability of the system to continue operation after a failure of a portion of the system has occurred. Recovery may be thought of as the ability of the system to prevent a failure of a portion of the system from causing a failure of the entire system.

The term failure can have several meanings. A component failure is the failure of a component (a modular portion of the system) to perform as specified. A component failure can be temporary or permanent. A temporary failure would be an occurrence of a failure which was brief (relative to the time requirements of the entire system) and which was followed by normal operation of the component without any repair being necessary. A permanent failure of a component requires the repair or replacement of the component before normal operation will ensue. Intermittent failure is the continued occurrence of temporary failures. Any of these types of component failures could cause a system failure depending upon the system requirements for the performance of the component.

An example of temporary failure would be the failure of a data line to transmit data correctly because of noise. This would cause a system failure if there were no error checking in the transmission, or, upon finding the error, if the software in the system had no provisions for making a retry of the transmission. Systems which are able to correct temporary errors often refer to them as soft errors, and to uncorrectable errors as hard errors. The error checking described here is an example of error detection.

Error detection is the ability of a system to determine if a component performed or is performing as specified. This ability can be expressed as a proportion of the total number of errors occurring that are detected; however, the ability is usually expressed in terms of the types of errors that will be detected.

2.2 RELIABILITY CONCEPTS

Given the structure of a system and the reliabilities of the components of that system, the reliability of the system can be calculated. The reliability of a system whose components are connected in series is the product of the reliabilities of the components. The reliabilities of the components should be determined with respect to the requirements made of the components by the system. Thus, a system of 1000 components connected in series, each with reliability of .999 of operating 1 hour, would only have a reliability of $(.999)^{1000} = .36$. Thus in very complex systems, redundancy may be required to increase the reliability. However, with redundancy, the number of parts in the system increases and the maintenance time required and cost of the system will increase. Maintenance is often required to keep components of a system at their required reliabilities, as will be shown later.

A study of the occurrence of failure of equipment has shown that there are three types of failure: early failure, chance failure, and wearout failure. Early failure is caused by bugs in the manufacturing process. It can be prevented by a debugging process of sufficient length. Wearout failure is caused by wearout of the equipment and is usually normally distributed about the mean wearout life of a component. Most wearout failures can be prevented by replacement of components before wearout takes effect on the failure rate. Chance failure is called such because the nature of its distribution is statistically random; it refers to all failures that are due neither to early failure nor to wearout failure. Prevention of early failures and wearout failures will reduce the failure rate of a component to its lowest possible value, the chance failure rate.

The reliability of a serial system is only affected by the mean up time or mean time to failure of a system; Availability, however, is affected by both mean up time and mean repair time, since availability is the proportion of the total time that the system is performing as required. Hence, ease of repair and aids in finding the failing part affect availability, but not reliability.

In the case of a redundant system, however, reliability is affected by the mean down time because the system reliability depends on the availability of redundant units.

The following chart (Fig. 1) gives an idea of the factors that contribute to high system availability. The factors presented are both hardware and software features of a system, although the software features are dependent upon the implementation of corresponding hardware capabilities.

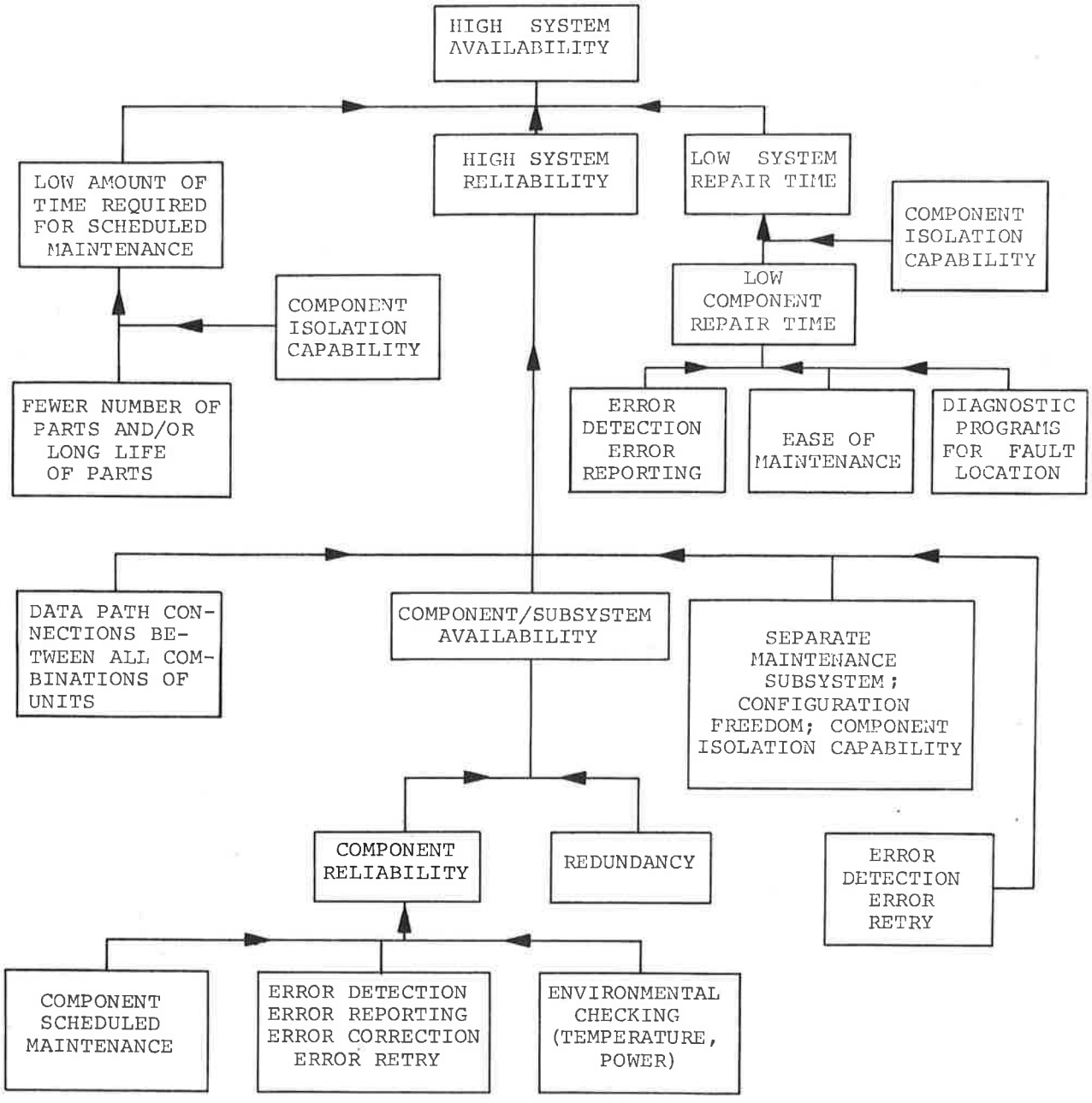


FIGURE 1. REPAIRABLE SYSTEMS WITH REQUIREMENT FOR CONTINUOUS OPERATION

2.2.1 Low Amount of Time Required for Scheduled Maintenance. This is a factor in system availability since a unit may not be used by the system while maintenance is being performed. It is also a factor in the minimum amount of time a unit must be unavailable. Unless the maintenance of a unit does not require the use of a nonredundant element of the system, the maintenance time contributes to system down time, if system requirements require system operation during maintenance time. This condition would be met if a system had no redundant elements and the system had to operate on a 24-hour per day basis.

2.2.2 Fewer Number of Parts and/or Long Life of Parts. In order to maintain a specified level of reliability in a part, that part must be maintained (cleaned) and replaced before the wearout probability becomes significant in comparison to the probability of chance failure. The wearout life of solid state electronic components is yet to be determined.

2.2.3 Component Isolation Capability. This is the ability to isolate a component from the rest of the system in order to perform maintenance without disturbing the rest of the system, e.g., the ability to power up and down independent of the rest of the system, or the ability to exercise a unit which has been removed from the control of the rest of the system. Without these capabilities the whole system has to be brought down to perform part of the maintenance.

2.2.4 High System Reliability. This contributes to availability by virtue of guarantying long times to failure of the system.

2.2.5 Low System Repair Time. Unless the system can be repaired quickly, high system reliability will be undermined.

2.2.6 Low Component Repair Time. The effect of this on system repair time is obvious.

2.2.7 Component Isolation Capability. This is once again a factor, as it was for maintenance, because of the similarity of maintenance and repair.

2.2.8 Error Detection/Error Reporting. This item is defined as the testing of the performance of a component by itself or another component and the reporting of all

errors found, including those which do not affect the performance of the system. The knowledge by the repairman of the type of error that caused a component failure often aids in quick location of the fault within a component.

2.2.9 Ease of Maintenance. The ease of maintenance means the ability to reach, to test, and to repair quickly and easily all repairable or replaceable portions of a component.

2.2.10 Data Path Connections Between All Combinations of Units. This means the ease with which a redundant portion of the system can be used to replace a failed portion of the system. For example, if one input/output processor of a system goes down, the compute element attached to that input/output processor may also be unusable unless it has connections to another input/output processor. In many cases the connections should be controllable, i.e., a unit should be able to use an alternate path on the flip of a switch or perhaps with the use of a specific signal or command. Control of this type is found in the 9020 system with the use of the configuration control register, which specifies to a unit which data path connections to itself are legitimate users of itself.

2.2.12 Component/Subsystem Availability. This merely states that the greater the amount of time the series components of a system are up, the greater the system reliability.

2.2.13 Configuration Freedom, Separate Maintenance Subsystem, Component Isolation Capability. These things contribute to the ability to repair or maintain a portion of the system while the system remains operating. Configuration freedom is the ability of the software to adjust to changes in which specific components are being used by the system. A separate maintenance subsystem is maintenance software which operates concurrently with, but independently of, the system software. The devices used by the maintenance subsystem may be recalled by the system software if needed.

2.2.14 Error Detection/Error Retry. This is an important factor in the system reliability because the system time requirements for an operation are generally much more lenient than the shortest time in which the operation can be performed.

Hence, if the system can detect a component error, it can retry the operation by using the time available for the system requirement. This is important when a component is prone to temporary failure.

2.2.15 Component Reliability. The effect of this on component/subsystem availability is the same as was the effect of high system reliability on system availability.

2.2.16 Redundancy. Redundancy contributes to subsystem availability by increasing the probability of the required number of components being up at any particular time.

2.2.17 Component Scheduled Maintenance. The importance of maintenance in keeping reliability at a specified level has already been mentioned.

2.2.18 Error Detection/Error Reporting/Error Correction/Error Retry. Error detection enables error reporting, error correction and error retry. Error reporting aids in component reliability by making repairmen aware of temporary failures and giving them the opportunity to correct the causes of a temporary failure before those causes cause a permanent failure. Error correction and error retry are similar. They both involve the correction of a temporary failure. The speed of the correction is much greater in the case of error correction. Error correction is important in the case where timing is critical. For example, the uses of error retry are involved with the use of secondary storage. It involves (1) read checking after every write operation and then retrying the write if a read check occurs and/or retrying the read check, (2) retrying a read operation if an error on the first read occurs. Item (1) is also a form of error detection. In the case of primary storage, the use of these techniques is generally too time consuming. The use of error correction, which means transmitting redundant information which is used both to detect and correct an error, however, reduces the probability that an error in a write in the past will cause a system failure or the necessity to restart system operation from some point in the past.

2.2.19 Environmental Checking. Environmental checking is related to error detection in that it involves monitoring external factors such as high temperature or power loss, which could cause component failure.

2.3 FUTURE RECONFIGURATION AND RELIABILITY REQUIREMENTS

It is hard to say exactly what the reconfiguration and reliability requirements for future ATC equipment should be. However, as the traffic increases, and automation assumes more of the human air traffic controller tasks, the dependence on automated equipment will increase. One would want the total system to maintain its current reliability in terms of preventing air traffic collisions or crashes.

The current NAS (9020) system seems good in that it is composed of subsystems with redundant elements. Since more than one element must go down for the system to go down, this enables restrictions to be made on the air traffic patterns by the controller before the entire system goes down.

In the year-long reliability acceptance test [1], the 9020 performed very well, producing a calculated Mean Up Time of 33,543 hours for the 325 flight A1 mode (failsafe) of operation. This is close to 4 years of 24-hour a day operation. The Mean Up Time for the 325 flight C1 (failsoft) mode of operation was 118,638 hours which is almost 14 years.

However, with regard to C1 mode of operation, if another IOCE was available, the Mean Up Time would be increased to over 1,000,000 hours, or over 100 years. At this high level of reliability, one has to begin checking the other elements of the system such as the power, radar equipment and airplanes themselves, in order to find the weakest link in the chain. Having the reliability of one of the elements of a serial system more than two or three orders of magnitude greater than the other elements does not materially affect the reliability of the entire system.

One also should begin worrying about keeping the same level of maintenance that was performed during the year-long reliability test. One should also worry about keeping the same quality repairmen. A reduction in speed of the repairs (mean down time) of a factor of 2 or 3 would result in a twofold or threefold reduction in the system mean up time.

For future systems, aiming for 10,000 hours mean up time for failsafe mode and 1,000,000 hours mean up time for failsoft mode does not seem out of reach nor overly expensive. These figures are a factor of 10 higher than those specified in the NAS reliability specification.

---

1. "IBM 9020 Data Processing System Reliability Determination," October 1966, reported under Contract FA 64 WA-5223.

3. SYSTEM ORGANIZATION AND RELIABILITY

In this section, the effect of computer architecture on reliability will be discussed.

3.1 PEPE, THE PARALLEL ELEMENT PROCESSING ENSEMBLE

PEPE is a special-purpose computer designed specifically for radar data processing. "The machine consists of an unstructured ensemble of an indefinite number of processing elements operating under common control. Each processing element contains the registers and adder logic of a single-address parallel arithmetic unit, and a random-access memory, and a minimum of control logic, the bulk of the control being concentrated in the common control unit. A major component of each unit is a special-purpose concurrent input unit, called a correlator, made up of a number of associative registers. The ensemble is content addressed and the input/output and control operations use associative techniques to provide novel capabilities for data input and retrieval." [1]

PEPE is designed to run under the control of a conventional computer called the host. Figure 2 shows the PEPE machine organization.

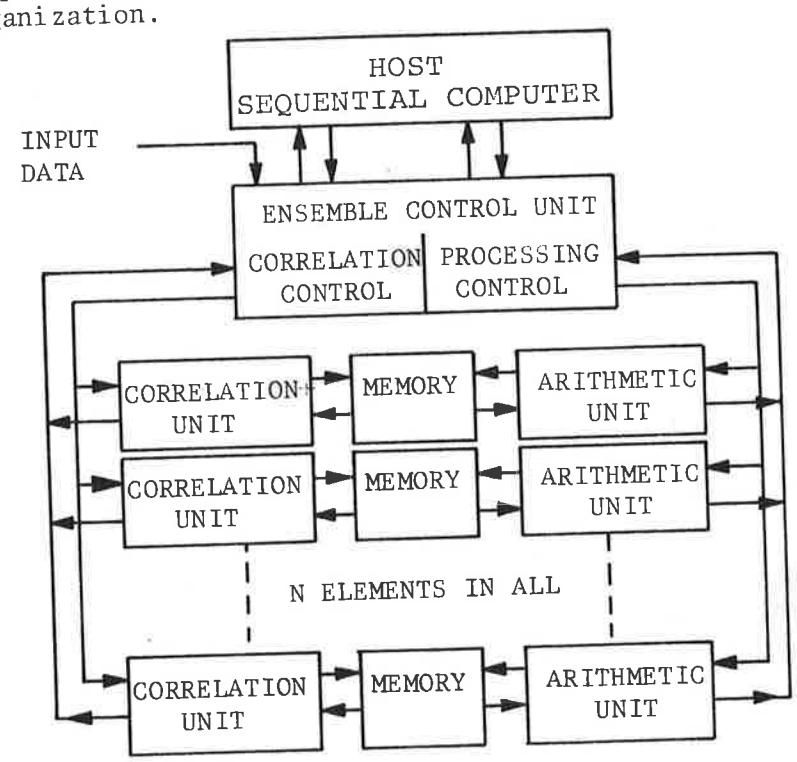


FIGURE 2. ENSEMBLE ORGANIZATION

In an actual application, each target being tracked is assigned a processing element. Computations are carried out in parallel on all targets at once. The data base structure is identical in all element memories; thus, "the element memory is location addressed with the same word location being selected in all memories in the ensemble."

The reliability aspects of PEPE are threefold. Since each processing element is content-addressed, any element can replace any other element. Any element "can be simply decommissioned electronically and left in place until removal and repair are convenient. This allows ensembles to be sized larger than required for the traffic load in anticipation of attrition due to failure. The excess capacity provided can be calculated using failure rate information to provide any specified level of availability."

Secondly, the design of PEPE, in comparison with other array or associative processors, uses a minimal number of parts since most of the control logic is centralized in the PEPE control unit.

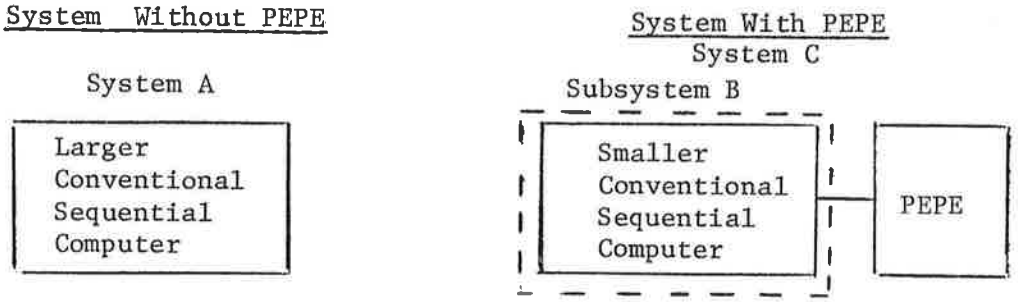
Thirdly, PEPE reduces the computational and storage requirements of the host computer. This is illustrated in Figure 3. Since reliability is a function of the performance requirements of a system, the reliability of the host computer portion of a system with PEPE would be greater than a system without PEPE. (The reliability of subsystem B is greater than the reliability of system A in the figure.) The reliability of the system with PEPE would be the reliability of PEPE alone multiplied by the reliability of the host computer portion of the system, which in almost all cases would still be greater than a system without PEPE. (System C has a greater reliability than System A in the figure.)

In order to get an estimate of lower and upper bounds of PEPE's reliability, one can assume the following: A parallel computer of the PEPE type is implemented with 60 processing elements each tracking one aircraft in each of thirty-two 11-1/4° sectors. (Each element tracks 32 aircraft, each of which is in a different sector of the 4-second radar scan. This scheme gives the computer a capacity of 32 X 60 = 1920 tracks with 4/32 = 1/8 second processing time per track.) Reserving 2 or 4 processing elements to be used as spares will prolong the life of the system to acceptable margins. The following chart shows the system mean up times calculated using Einhorn's method [2]. Three sets of assumptions are shown. The assumption of 1,000 or 10,000 hours mean up time for each processing element is reasonable. The actual value of the mean up time will probably lie somewhere between those two figures.

Assumptions	System Mean Up Time
$U = 10^4$ hrs. $D = 1/2$ 4 spares	4×10^{13} hours
$U = 10^3$ hrs. $D = 1/2$ 4 spares	4×10^8 hours
$U = 10^3$ hrs. $D = 1/2$ 2 spares	9×10^3 hours

where U = processing element mean up time
and D = processing element mean down time.

Given: A performance requirement; for example, track n targets.



	A	B	C
System or Subsystem Alone Meets Performance Requirement	YES	NO	YES
Reliability of System or Subsystem	x-d	x	$(x)(R_{PEPE})$

where $d>0$ and R_{PEPE} is the reliability of PEPE.

Assuming the reliability of PEPE can be made arbitrarily high,

$$x-d<(x)(R_{PEPE})<x$$

FIGURE 3.

Effect of PEPE on Reliability of ATC Computer System Because of Reduction in Sequential Computer Computational and Storage Requirements

3.2 GOODYEAR ASSOCIATIVE PROCESSOR

The Goodyear Associative Processor is an array processor capable of parallel arithmetic, parallel I/O and parallel search operations. The basic unit of the Goodyear Processor is the basic processing element, which consists of 256 bits of storage and a response store. Communication between neighboring elements is provided by a shift up or down capability in the response store. The basic hardware element of the Goodyear Associative Processor consists of 256 of these processing elements. The complete associative processor consists of "M" modules under global control of a single control unit as shown in Figure 4. The Goodyear Associative Processor is discussed more fully in the Large Scale Systems Report in the present series.

An estimate of the reliability of the Goodyear Associative Processor can be made as follows.

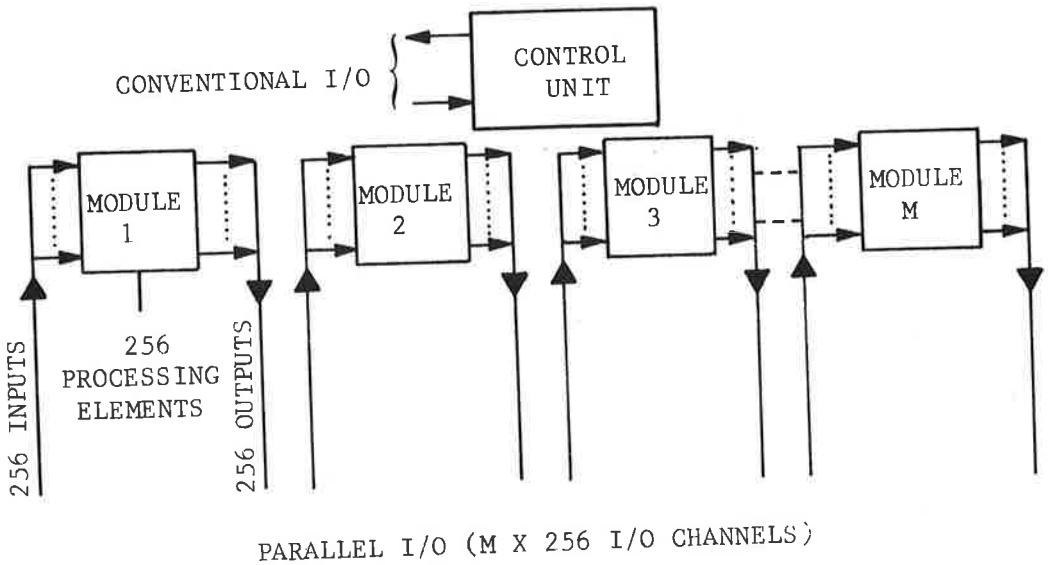


FIGURE 4. GOODYEAR ASSOCIATIVE PROCESSOR

According to the Goodyear Aerospace Processor proposal, the failure rates of the component parts of an Associative Processor plane (containing 256 processing elements) add up to produce an overall failure rate of 1 in 10,000 hours. The complete associative processor, containing 20 planes, has a failure rate of 1 in 500 hours, or a mean time between failures of 500 hours.

However, the associative processor has inherent redundancies which can be used to increase the effective mean time between failures of the system. One type of redundancy is being able to use any of the 256 words to replace any other since data is usually accessed associatively rather than by location. Thus, a certain number of words in a plane can be reserved to be used as replacements for words in that plane that fail. Another type of redundancy is being able to use any of the 20 planes to replace any other plane. Both these types of redundancies are "inherent" because they require very little programming to implement. One other possible type of redundancy is the apparent ability to use any of the 256 bits to replace any of the other bits. This ability occurs because there is no shifting of bits left and right within a word. However, in order to replace one bit by another a physical connection on the plane would have to be changed. Thus this ability could not be used to recover quickly from a failure in a bit strap.

The first two types of redundancies are proposed by Goodyear for increasing the mean time between failures of the system. Goodyear proposes reserving 20 of 256 words per plane as spares and reserving 2 out of 20 planes per processor as spare planes. Goodyear also proposes, in addition, to operate the system in duplex, with two complete associative processors processing identical data simultaneously. The value of reserving 20 words as spares is not evident unless most of the failures (80 to 90 percent) in a plane only cause failure of one word, and not, for instance, failure of the same bit in all words. The failure rate data needed to determine the value of reserving 20 words as spares has not been released. The value of using 2 planes out of 20 as spares is computed below and increases the mean time between failures from 500 to 1582 hours.

Using the analysis presented in Igor Bazovsky's Reliability Theory and Practice (p. 106) [3], we can compute the MTBF of the Goodyear Associative Processor as follows:

Let $R_{GAP}(t)$ = Reliability function for the Goodyear Associative Processor

$R(t)$ = Reliability function for a plane of the Goodyear Associative Processor

$Q(t)$ = $1-R(t)$ = Unreliability function for a plane

Since only 18 of 20 planes are needed for operation,

$$R_{GAP} = R^{20} + \binom{20}{1} R^{19} Q + \binom{20}{2} R^{18} Q^2.$$

Since each plane has an exponential failure rate,

$R(t) = e^{-\lambda t}$, where λ is failure rate.

Using the fact that

$$MTBF = \int_0^\infty R_{GAP}(t)\, dt$$

and substituting the equations for R_{GAP} and integrating, we obtain

$$MTBF = \left(\frac{1082}{6840}\right)\left(\frac{1}{\lambda}\right)$$

Since λ is $\frac{1}{10,000}$ hours,

$$MTBF = 1582 \text{ hours.}$$

The value of the duplex operation of associative processors can also be determined. If we assume that a processor will be repaired as soon as any failure occurs and that the repair time is a constant 1/2 hours, then an approximation to the mean time between failures of the duplex system can be made as follows:

> Failures in the duplex system occur, on the average, every 250 hours, since there are 2 processors with 500 hours mean time between any failure. The probability of the other processor failing completely during the repair period is approximately λt, where λ is $\frac{1}{1582}$ and t is 1/2. Thus, the MTBF of the duplex system is $250\ [(1/2)\ (1/1582)]^{-1} = 791{,}000$ hours.

The duplex system also provides the possibility of simpler failure detection. Once a second the response store contents of the second associative processor is transferred in parallel to the first AP and compared to the contents of the first associative processor. If a failure is detected, the word in which the failure occurred is known, and a test program is used to determine which associative processor failed. While the test program operates, the normal functions of the system can be carried on using the spare words in the plane. One should be sure that the failure could not have spread, or will not spread, erroneous data through the system if it is decided to continue operation after a failure is detected.

If a duplex system is not used, or during the repair of one of the associative processors, a test program will have to be used to detect failures. A suitable test program which checks

the operation of one word at a time and which has a run time of one millisecond can be run once per second in each plane to make a complete check of the associative processor once every four minutes.

The error detection ability in a single associative processor does appear to be less complete than that found in other types of processors. More study may be necessary to determine how additional error detection could be built in. Of course, a duplex system of associative processors offers excellent error detection, especially since every word in both processors is compared during each test each second.

3.3 IBM 9020, UNIVAC ARTS III: MULTIPROCESSORS

3.3.1 Introduction. The IBM 9020 and UNIVAC ARTS III computer systems are multiprocessors, although a single processor system is possible. The use of multiprocessing in these systems provides additional processing capability, but it also provides backup processing capability which can be used in case of failure of one or more processors. This backup processing capability is referred to as "failsafe" capability. In the case where failure of a processor or processors leaves less processing capability than is normally needed for operation, software can use this limited processing capability to provide completion of only the most critical tasks. This ability to perform limited processing is referred to as "failsoft" capability.

The overall scheme of providing reliability for the 9020 and ARTS systems is to build systems with multiple and redundant units and to provide hardware and software capability to use any of the units available. Prerequisites to this type of operation are error detection and reconfiguration, as was discussed in Section 2 of this report. This scheme is opposed to a duplex scheme of providing reliability where processing is duplicated and the results of the processing are checked. The basic difference between these two schemes from a reliability viewpoint is that redundancy is provided on a lower level (for smaller modules) in one system. Providing reliability on a lower level gives greater reliability since if one module fails the entire system does not fail.

A generalized view of the multiprocessing scheme is as follows: One can think of the processing as one or more separate processes. On the 9020 and ARTS systems there would usually be only one process. Each process is composed of many tasks or units of work to be accomplished. The process (not attached to any piece of equipment) allocates equipment for these tasks and schedules the tasks whenever the equipment becomes available. Thus any physical processor can execute that portion of the software known as "the system". More than one processor can be executing "the system" simultaneously, except for certain nonreentrant portions of code. Hence the process continues when a portion of the system, including a processor, goes down. Reconfiguration merely involves changing a table of available equipment.

In this section the reliability features of the processors of the 9020 and ARTS systems will be described. A

more general view of certain reliability features, error handling and reconfiguration, as applied to the entire systems are described in Sections 4 and 5.

3.3.2 IBM 9020 CE. The IBM 9020 CE has quite a number of reliability features. They can be grouped into three areas: logout capability, machine checks, and element checks. The logout feature enables a compute element to dump all important registers in a predetermined area within a storage element. A logout is performed whenever a malfunction occurs to enable later diagnosis of the problem. A machine check is an interrupt which alerts the CE to a hardware error in the CE or an element the CE is controlling. An element check is an interrupt which alerts a CE that a major failure has occurred. Element checks represent malfunctions which should be processed by a CE other than the controlling CE. The five basic types of element checks are out-of-tolerance condition, on battery standby, power off, logic check, and CCR (configuration control register) parity.

Direct control of one CE by another is an important reliability feature. One CE may direct another to logout or to perform an external start. An external start involves resetting the registers and obtaining a new PSW (program status word) from a particular storage location. Which CE has control of another is determined by the configuration control registers. Thus, a program error could not spuriously cause an external start.

3.3.3 UNIVAC ARTS III. The ARTS III system really has two types of processors. One is the IOP, input/output processor; the other is the CPM, central processor module. The IOP can perform input/output operations as well as most computing instructions. The CPM can perform no input/output operations but has more computational power than the IOP. An ARTS III system can contain up to 4 IOP's and 4 CPM's, along with sixteen 16K-30 bit word memory modules.

The fault handling and reconfiguration capability of ARTS is centralized in the Reconfiguration and Fault Detection Unit, which receives information by direct hard-wire from each of the IOP's, CPM's, and memory modules. The RFDU is interfaced to the system in three ways. Two "normal" interfaces, used for control, connect the RFDU to two IOP's. One IOP is referred to as the primary IOP and the other as the secondary IOP. The secondary IOP controls the RFDU only if the RFDU detects an error in the primary IOP.

One type of "special" interface connects the RFDU to each of the IOP's and CPM's. These special interfaces are used to transmit error indications to the RFDU as well as to prevent the honoring of "normal" and "high priority" device interrupts from a failed processor in all other system processors.

Another type of "special" interface connects the RFDU to the memory modules or a centralized memory access module. This interface can disable or enable all lines between any processor and any memory module. A configuration matrix (8 X 16) composed of flip-flops within the RFDU controls the disabling lines. Each row of the matrix corresponds to a processor, and each column, a memory module. The matrix may be changed manually or by program control.

By use of the normal and special interfaces the RFDU can electronically isolate any processor or memory module. Another function of the RFDU is to retain the power failure recovery location to be entered when powering up after a power failure.

The RFDU has one reliability feature built into its design. This is the requirement that two separate commands must be given in order to change the configuration matrix. This requirement prevents the software from issuing an erroneous reconfigure command.

3.3.4 Conclusion. The IBM 9020 and UNIVAC ARTS III multiprocessing systems use their expanded processing capability to provide additional processing capability and redundant processing capability. The redundant processing capability increases the reliability of the systems. In the 9020 system, the additional hardware for reconfiguring the system is built into each compute element; in the ARTS III system, a separate centralized piece of hardware has been added to the system, which can be controlled by either of two Input-Output Processors. The 9020 scheme is a more general approach and essentially more reliable since reconfiguring hardware is duplicated in each CE. The reliability of the ARTS III reconfiguration scheme is limited by the reliability of the RFDU; however, it may still be more reliable than the rest of the ARTS III system and hence not really a limiting factor to the ARTS III reliability. One problem with the ARTS III RFDU appears to be that there is no on-line checking of its operation, such as parity checking; an error by the RFDU may go unnoticed.

3.4 IBM/370

The new series of machines announced by IBM, the System 370 Model 155 and Model 165, contain many new reliability features. They are referred to by IBM as "Reliability, Availability, and Serviceability Features".[4][5] They include the following hardware modifications:

CPU retry of most failing operations.

Error coding and correction validity checking on processor storage to correct all single-bit errors.

Automatic deletion of malfunctioning buffer blocks.

I/O operation retry facilities, including channel retry data provided in the ECSW (Error Control Status Word) and channel/control unit command retry procedures to correct failing I/O operations.

Expanded machine check interrupt facilities to facilitate better error recording and recovery procedures.

The following software additions were made:

Recovery management support (RMS) to handle the expanded machine check interrupt and channel retry data.

Error recovery procedures (ERP) to retry failing I/O device and channel operations.

OBR and SDR routines to record statistics for I/O errors.

Environment recording edit, and print program (EREP).

I/O RMS routines, alternate path retry (APR), and dynamic device reconfiguration (DDR).

Checkpoint/restart and warm start facilities to simplify and speed up system restart procedures.

According to IBM engineers, the 370 was designed with reliability a major consideration. The following techniques were used in the design of the 370:

Improved component reliability. The greater circuit packaging densities used in System/370 result in greater reliability and a reduced number of interconnections.

3-Sigma design technique. The chains of logic circuits are designed to tolerate a wide variation in the values of the individual components. A statistical design technique is used, in which the total value for a chain of circuits is assumed to have a normal distribution about the design point. All chains are designed to continue operating properly when the actual value varies from the design point by three times the standard deviation. The probability of exceeding this deviation is less than 0.003.

Electromagnetic compatibility. System/370 is designed with improved tolerance to radiated electromagnetic fields, static discharges, and power line disturbances. This is accomplished by such techniques as plating the structural frame members and using matched transmission lines for interconnections. These measures also result in reduced radiation emanating from the system.

Vibration testing. Both the individual component and entire assembled units have been subjected to extensive vibration testing.

Improved internal testing. More frequent checking of data paths results in more rapid detection and better localization of errors. Sequencing and control circuits are more completely checked than in previous systems.

Exhaustive simulation and testing. All the logic circuitry has been extensively simulated to detect loading and timing design errors. The full functions of Operating System/360 have been available to thoroughly check the interactions of system components. There has already been extensive usage of System/370 within IBM. Before the first external shipments to customers, many systems will have had thousands of hours of use. Much of this usage will have taken place in a production environment similar to that of customers, not just in artificial test situations.

IBM also provides several new features for reducing the amount of down time with the following diagnostic and repair tools:

Logout. Data on intermittent errors is captured by logging on disk. A processor logout analysis program is available to the customer engineer. This program runs in an on-line test environment. It formats and prints the CPU and channel logouts and is capable of analyzing a series of logouts to localize fault to the most probable group of replaceable components. Input-output unit error recordings are formatted and printed by the error recording edit and print (EREP) program.

Microdiagnostics. Diagnostic programming for System/370 processors is done at the microprogram level. Microprograms provide more direct access to the hardware, with a resulting improvement in their localization capabilities.

Storage tests. In addition to simple ripple tests to provide quick checkout of storage units, transient software diagnostic routines provide complete testing and fault localization for main storage, buffer storage and local storage.

System test. A system test program is capable of rapidly checking out all local input-output devices and localizing most faults to the device level.

Diagnostic file. A small read-only file employing a single replaceable disk is contained in each System/370 console. This file is used to enter the microdiagnostics into the system. In the Model 155, it also contains the storage tests and the system test. This diagnostic file has a direct path into the processor data flow, reducing to a minimum the "hardcore" of equipment that must be operational to run diagnostic tests.

Console functions. Several new system console functions are included in System/370 to aid both the customer engineer and the system operator. Entry and display functions for main storage, local storage and internal registers now make use of the console printer-keyboard, with direct hexadecimal encoding. Direct control is provided for initiation of tests from the diagnostic file and presentation of results. The 3066 Console on Model 165 includes a cathode ray tube operator display, a microfiche projection display of internal registers, and a microfiche viewer for maintenance documentation.

RETAIN/370. The Remote Technical Assistance and Information Network has been greatly enhanced for System/370 by the addition of direct data transmission facilities. This field engineering facility brings to the customer engineer immediate specialist consultation to reduce the downtime resulting from difficult problems. The data link allows the remote specialist to initiate tests on the system and receive results. In addition, the specialist has direct access to a data bank of symptoms and solutions, which provides rapid information interchange.

The features mentioned above should substantially increase the availability of the 370 above that of the 360. The system organization of the 370 is not novel; however, reliability techniques have been used on the 370 design in order to attain the greatest possible availability from a uniprocessor computer system.

REFERENCES - SECTION 3

1. Githens, J.A., "A Fully Parallel Computer for Radar Data Processing," NAECON '70 Record, P. 290.

2. Einhorn, S.J., "Reliability Prediction for Repairable Redundant Systems," Proceedings of IEEE, Vol. 51, No. 2; February, 1963; pp. 312-217.

3. Bazovsky, Igor, Reliability Theory and Practice, Prentice Hall: Englewood Cliffs, New Jersey, 1961.

4. A Guide to the IBM System/370 Model 155, Form GC20-1729-0, IBM, 1970.

5. A Guide to the IBM System/370 Model 165, Form GC20-1730-0, IBM, 1970.

4. ERROR DETECTION AND ERROR HANDLING

4.1 GENERAL CONSIDERATIONS

4.1.1 Effect of Error Detection and Hardware Reliability Upon Accident Rate.

As mentioned in Section 2 on System Reliability Concepts, error detection and error handling are important features to a high reliability system. It is important to reduce the chance of an undetected error, or at least the spreading of erroneous data resulting from an undetected error through the system, to an absolute minimum. Otherwise, a catastrophic failure could result.

The following analysis of the relative worth of error detection and hardware reliability can be made. Given a computerized air traffic control system S, let p_0 be the average number of occurrences of an error in a given time t. Let p_1 be the probability of detecting any error that occurs. Let p_2 be the probability that an error is undetected; hence, $p_1 + p_2 = 1$. Now, in order to determine the effect of the detection rate, let S_1 and S_2 represent the severity of a detected error and an undetected error, respectively, where severity means the accident rate or probability of an accident resulting from an error. The following equation results from the above assumptions:

$$A = p_0[p_1S_1 + p_2S_2], \qquad (4.1.1)$$

where A is the average number of accidents in a given time t.

Applying the fact that $p_1 + p_2 = 1$,

$$A = p_0[S_1 + p_2(S_2 - S_1)] \qquad (4.1.2)$$

or

$$A = p_0[B + CD] \qquad (4.1.3)$$

where

$$B = S_1$$

$$C = p_2$$

$$D = (S_2 - S_1).$$

Assuming p_0 remains constant, there are two factors contributing to the accident rate. B, which represents the severity of a detected error, is generally a function of the reconfiguration and backup capability of a system.

This backup capability might even include the warning of pilots by voice channel that a computer breakdown had occurred. The product CD represents the effect of possible undetected errors on the accident rate. C represents the error detection ability of a system; as the error detection ability increases, C decreases and the accident rate drops. D represents the difference in severity of an undetected and detected error; unless the difference in severity is significant, error detection is not important. However, one would expect that the difference in severity of errors would be quite great. An undetected error could cause loss of enormous amounts of data and make recovery impossible and require a restart of the system from scratch. Thus, in order to improve the safety of a system, either the severity of a detected error could be reduced or the error detection ability of the system could be increased. Either the first or second course of action could be more beneficial depending on the values of B, C, and D.

4.1.2 Hardware Error Handling. Error handling procedures, which were once almost exclusively performed with software are now being performed with hardware automatically. These hardware error handling procedures do not increase reliability per se, since the procedures in most cases could be provided by software, as long as the same error detection facilities are in use. However, they do increase reliability in some cases by speedier handling of errors, thus providing more time for recovery.

Examples of hardware error handling are given in IBM announcements about the System 370. In the CPU, there is automatic retry of most failing operations. The processor storage control unit corrects all single bit errors using an error correction code. The input/output channel also has retry procedures for retrying failing I/O operations, although retry is not automatic since the program that issued the I/O command may be time-dependent on the operation.

4.2 IBM 9020 ERROR DETECTION AND ERROR HANDLING

4.2.1 Hardware Description

4.2.1.1 Introduction.

This section describes the error reporting methods used in the 9020 System.

4.2.1.2 Program Status Word. The program status word (PSW) of the 9020 System is used to control instruction sequencing and to hold and indicate the status of the system in relation to the program currently being executed. By storing the current PSW during an interruption, the status of the CE is preserved. By loading a new PSW or part of a PSW, the status of the CE can be initialized or changed.

The interruption system permits the CE to change status as a result of conditions external to the system, in input/output (I/O) units, or in the CE itself. Five classes of interruption conditions are possible: I/O, program, supervisor call, external, and machine check.

Each class has two related PSW's called "old" and "new". They are stored in unique main-storage locations in the Preferential Storage Area. An interruption causes the current PSW to be stored in its "old" position and the PSW at the "new" position to be made the current PSW. The "old" PSW contains all necessary status information about the system at the time of the interruption. If, at the conclusion of the interruption routine, there is an instruction to make the old PSW the current PSW, the system is restored to its status prior to the interruption and the interruption routine continues.

4.2.1.3 Interrupts. Errors are reported to the CE by an interrupt. All interrupts are not errors, but all errors are reported via interrupt.

Interruptions are taken only when the CE is interruptable for the interruption source. The system mask, program mask, and machine check mask bits in the PSW may be used to mask certain interruptions. When masked off, an interruption either remains pending or is ignored. The system mask may keep I/O and external interruptions pending, the program mask may cause four of the 15 program interruptions to be ignored, and the machine-check mask may cause machine-check interruptions to remain pending.

An I/O interruption provides a means by which the CE responds to conditions in the channels and I/O units.

The address of the channel and I/O unit involved are recorded in bits 16-31 of the old PSW. Further information concerning the I/O action is preserved in the channel status word (CSW) that is stored during the interruption.

Unusual conditions encountered in a program create program interruptions. These conditions include incorrect operands and operand specifications, as well as exceptional results.

A Supervisor-Call interrupt occurs as a result of execution of the instruction SUPERVISOR CALL. Eight bits from the instruction format are placed in the interruption code of the old PSW, permitting an identification to be associated with the interruptions. A major use of the instruction SUPERVISOR CALL is to switch from the problem-state to the supervisor state for supervisor service to the problem program.

The external interruption provides the means by which the CE responds to signals from the interruption key on the system control panel, the timer, and the external signals of other elements in the 9020 system. One type of external interrupt, the element check, is used to communicate between elements and is often a hardware generated error signal.

4.2.1.4 Machine Check. The occurrence of a machine check terminates the current instruction, initiates a diagnostic procedure, and subsequently causes the machine-check interruption. A machine check cannot be caused by invalid data or instructions. The diagnostic scan is performed into the scan area starting at location 128 of the Preferential Storage Area. Proper execution of these steps depends on the nature of the machine check.

Hardware problems are reported in the 9020 with a machine check interrupt and an element check caused external interrupt.

A machine check interrupts occurs after the automatic detection of a hardware error in the error detection circuits of the CE or an element the CE is controlling.

The cause of machine check interrupt can be determined by reading the registers of the CE with a Diagnose Instruction. The Diagnose Instruction reads the same information as is in words 56-62 and 66-69 of the Preferential Storage Area after a CE logout. Many machine check interrupts also cause an element check in the other computing elements.

4.2.1.5 Element Check. When a major problem exists in an element (not I/O device) in the system, it is communicated to a CE by an element check signal (ELC).

The element check causes an external interrupt with bit 31 set. This shows that the CE is to read its DAR (Diagnose Accessible Register). The DAR shows the element that caused the Element Check.

If the element is a CE or IOCE a Write Direct is used to logout the failing unit to provide more information on the failure. If the element is a SE the Diagnose Instruction can be used.

In the 9020 Multiprocessing system, serious problems cause two interrupts. For example, a parity error in a CE will cause a machine check interrupt in the failing CE and an element check to the other CE(s) configured to listen to the failing CE. On the other hand, an invalid instruction only causes a program check interrupt in the concerned CE.

This double examination is needed because the error analysis software may not be able to work in the failing element. It also presents the need to coordinate independent error handling routines.

4.2.1.6 Failure Types. In the 9020 System major failures result in an element check. Element checks are those checks made on malfunctions which occur within a CE, IOCE, SE or other major element and which cause flag bits to be set in a register contained within each CE known as the Diagnose Accessible Register (DAR). In the simplest terms, an element check is a malfunction that should be monitored by a computing element other than the controlling computing element. The five basic types of element checks are defined below.

OTC: Out-of-Tolerance Condition

Meaning: The temperature has risen in the indicated unit to within 10 percent of the Thermal Protection Temperature.

When the Thermal Protection Temperature is reached, power is sequenced down and cleared from the element to protect it from thermal damage.

OBS: On Battery Standby (CE, IOCE, SE)

Meaning: The indicated element has lost ac power and has switched to its own battery

Since some element checks are not uniquely identified in the DAR, additional information must be obtained via logout or sense data analysis.

CHECKS FROM:	SETS THIS DAR FLAG[1]
This CE[2]	
OTC	OTC
OBS	OBS
Other CE's	
LOGIC ERROR CCR PARITY POWER OFF	ELC[3]
IOCEs	
LOGIC ERROR	ELC[3]
CCR PARITY	OBS/pulse
POWER OFF	ELC[3]
OTC	OTC
OBS	OBS/word
SEs	
LOGIC ERROR CCR PARITY POWER OFF OTC OBS	ELC[3]
PAMs & TCUs	
LOGIC ERROR	
priority control (PAM)	ELC[3]
all other (PAM & TCU)	(via I/O check)
CCR PARITY	ELC[3]
POWER	
OTC	(via I/O check)

1. The flag is a single bit for every element except an IOCE; for an IOCE, the flag is a pair of coded bits.

2. "THIS CE" refers to the CE in which the DAR being considered is located.

3. The acronym "ELC" (from Element Check) is used to denote any DAR flag which is set by more than one kind of element check.

FIGURE 5.
DAR FLAGS SET BY ELEMENT CHECKS

4.2.2 9020 Operational Error Analysis Program (OEAP) (Fig. 6).

4.2.2.1 Introduction. The Operational Error Analysis Program performs on-line analyses of equipment failure indications concurrently with regular processing, and assesses the source and severity of each diagnosed malfunction. When required, the OEAP realigns the communication and control paths to functionally replace failing elements with redundant elements.

When an interrupt occurs in a subprogram, the 9020 control program gains control of the Computing Element executing the subprogram. Depending upon the type of interrupt and the subprogram priority, control is returned to the interrupted subprogram or another subprogram after appropriate action is taken by the control program. Error indications are passed to OEAP and the error environment is analyzed.

If the error continues, OEAP isolates the malfunctioning element and removes it from the operational system. All findings and actions of OEAP are reported back to the control program, and any additional configuration changes directed by the control program are executed by OEAP.

4.2.2.2 Error-check Analysis and Fault Isolation. The major portion of OEAP is devoted to analyzing the error-check environment and to isolating, if possible, the malfunctioning element or interface. Malfunctions cause one of three possible abnormal-condition signals (see Section 4.2.1.6). The main purpose of the error-analysis function is to identify the malfunctioning element or interface that is generating the error signal. The results of the analysis are reported to the error-control and statistical routine.

4.2.2.3 Maintenance of Error Statistics. Error conditions reported by the analysis routines are dynamically communicated to operational and maintenance personnel. A frequency count of the intermittent errors for each element is kept. Interface errors are also recorded when errors occur in both of two elements communicating

with each other. The operator uses the intermittent error count in deciding whether one or both of the interfaced elements should be removed from the system.

4.2.2.4 Error Environment Reporting. When a malfunction has been detected, the control program receives pertinent information about the failure. If a solid error occurred, OEAP reports that the element has been deleted; if the error was intermittent, OEAP requests a course of action. Relevant information is also reported via high-speed printer, typewriter and magnetic tape.

4.2.2.5 Reconfiguration. OEAP has sole responsibility for maintaining the system configuration. Except at initial program loading, it alone executes the reconfigure instruction.

4.2.2.6 Summary Comment. OEAP's design heavily depends on the convention that the Computing Element receiving a machine-check interruption should attempt recovery. This rule is fine if OEAP is resident in main storage. For applications with severely limited main storage, however, OEAP may have to reside in part on disk or drum; then the design philosophy becomes less desirable because a malfunctioning Computing Element necessitates an I/O operation before the work of analysis can start. Depending on the number of Computing Elements, the nature of the error, and the amount of main storage, trade-offs are obviously involved.

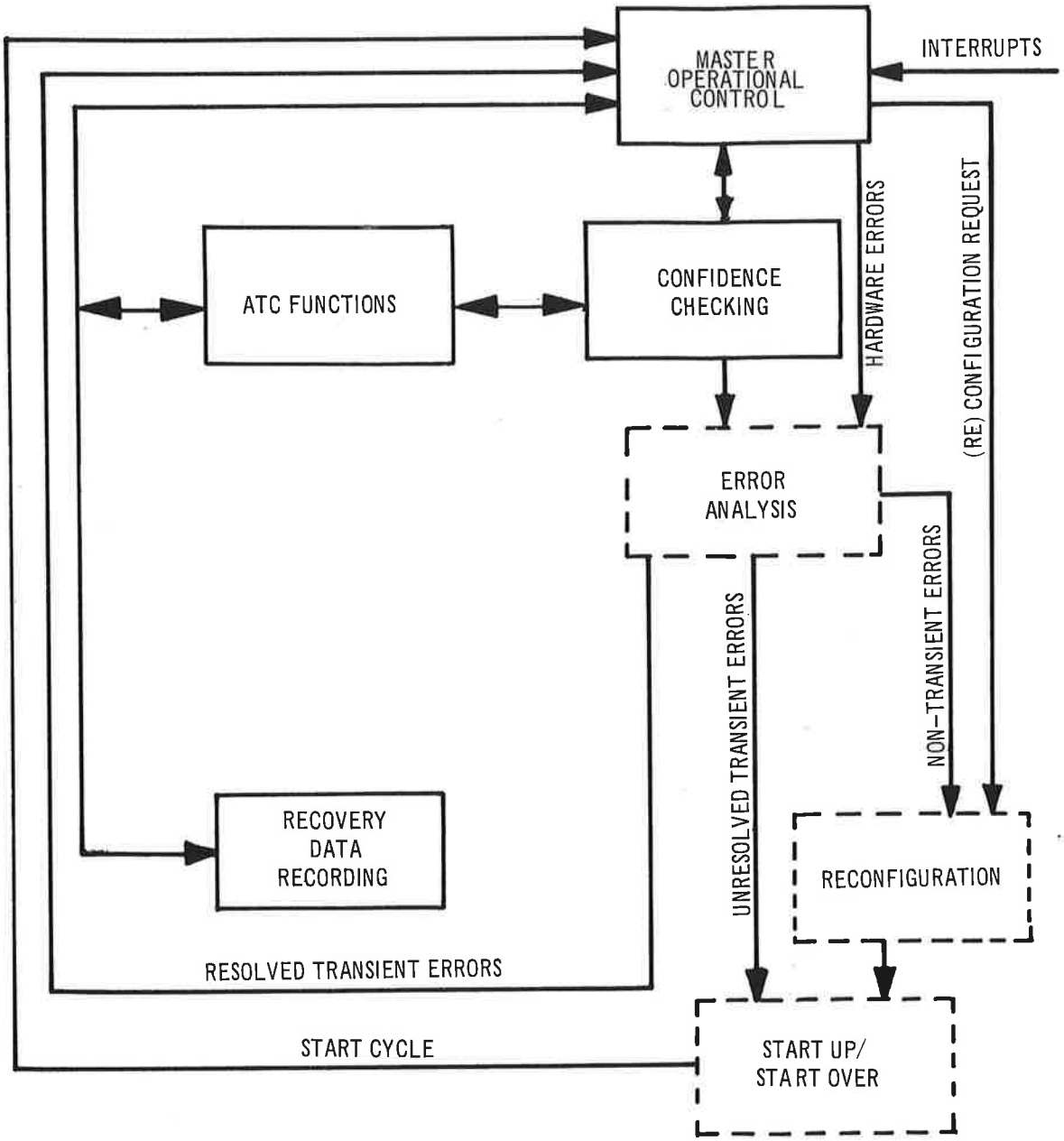


Figure 6. Relationship of OEAP to Other 9020 Programs.

4.3 DIAGNOSTIC AND MAINTENANCE PROGRAMS

4.3.1 General Concepts. Software is not used as extensively in error detecting as hardware is. Hardware is generally relied upon to provide most detection of errors. In some cases, however, software is used to detect the absence of a hardware signal, which may be an error. For example, a routine may check a clock to make sure that some device has not failed to respond to a command in the normal response time.

Software is used extensively in the area of fault location, however. Fault location programs are used to determine the specific area of a unit which is not performing properly. Maintenance diagnostic programs are often used as fault location programs. These programs exercise a unit in many different ways, and check the timing and performance of a unit. In the IBM 370, a new type of diagnostic program called microdiagnostics are used for fault location. These programs are written in the microcode of the machine so that they have more capability to discern exactly what is not working properly.

A diagnostic program is designed to test and verify the operational capability of computer logic that is accessible and available for test under operational conditions. Diagnostic routines may be assembled and loaded as a functional part of the operational programs. If adequate storage is not available, the diagnostic programs may remain on a bulk storage device until needed.

4.3.2 Software Checking of Hardware. A multiprocessor configuration lends itself to more thorough, software-implemented error checking than does a single processor. Cooperative processing checking and comparisons are possible during operation to provide system-type tests which hardware self-test programs alone cannot always provide.

4.3.2.1 Programming Techniques. There are a number of programming techniques which may be used in a diagnostic program. They are summarized below.

a. Check Sum - In the check sum method, a memory check is developed before operation is begun, i.e., all program instructions and constants are added together and the sum stored in a specific memory location, an invariance test. At specific intervals in the operational program, the check sum routine can be executed and the sum compared to the check sum in the previously defined location. Any noncompare will be regarded as an indication of failure. Hardware can be implemented to update the check sum in a writable memory element (ME), if desired.

b. Addressing Check and Instruction Test - This addressing check and instruction test routine would normally be utilized early in a series of self-test routines. Starting with a selected sequence of instructions, the processor hardware required for each instruction would be tested and verified for correct operation. They would be used to help verify the correct operation of other instructions until the complete set of instructions is verified. This is similar to the boot-strap technique, but implemented via software. During this process, the index registers, accumulator, and all other registers would be checked. In the addressing check, the addresses of contiguous storage locations would be stored throughout memory, and branches made to these locations. If the transfer instruction branches to a location that does not contain the specified address, a hardware failure in addressing has occurred.

c. Bit Check ("Memory Ripple Test") - A prestored set of selected constants can be used to manipulate each bit in storage. These constants can be addressed, stored in a location, complemented, and recomplemented; and a compare can be made with the original storage location. Any non-compare will indicate a hardware failure. This technique will test sense amplifiers and inhibit drivers of the ME's.

d. Interrupt Test - The interrupt feature can also be self-tested, e.g., programmed I/O commands can be used to force interrupts. Correct processing of interrupt sequencing and interrupt address generation can be checked and compared to a previously defined pattern.

e. I/O Test - A built-in I/O test feedback loop can provide a means of testing certain I/O data flow and I/O control logic. Programmed commands from the diagnostic routine can enable data feedback from one IOP or IOC to another IOP or IOC via a shared I/O device. Data patterns can be sent from storage through the I/O logic, back into storage and then compared. This will test both the I/O path and I/O commands.

f. Storage Protect - Storage protection for critical program areas and constants can be accomplished through a key or bit pattern set in the Program Status Word of each schedulable task. For storing in storage protected areas, the key must match or an addressing check will occur. A self-test program would attempt to violate storage protection keys by writing into protected areas.

4.3.2.2 Self-testing Programs. The purpose of self-test programs is to perform a periodic functional test of the computer system in the operational or checkout mode. Self-test programs employ the techniques discussed in Section 4.3.2.1. Hard machine failures can be detected through failures to execute portions of the program. Transient failures are detectable only if they occur during execution of the program.

C-P2 Computer Self-test Program. The IBM System/4 Pi Model aerospace computer utilizes a self-test program designed to functionally test and verify the operational capability of the computer system logic that is accessible and available under operational conditions. Both hardware and software self-test features provide the overall self-test capability for the system. The self-test features include:

(1) Main Bus/Parity Assignment
(2) Main Store/Parity Check
(3) Main Store/Storage Protect
(4) Input/Output Parity Assignment and Check
(5) Input/Output Test Feedback Loop

The CP Self-Test Program is designed to detect single-solid-fault type malfunctions as well as intermittent failures that are of sufficient duration to appear as a failure. However, isolation of the diagnosed malfunction will be at the computer level.

The user's requirements determine the test frequency for the Self-Test Program. The Executive routine controls the operation of the test program, which is composed of three segments:

(1) CPU Test
(2) I/O Test
(3) Interrupt Test

For a complete system check, all three segments must be executed. If it is not practical to execute all three segments every computational cycle, selection can be rotated to provide a complete test over several cycles.

The test sequence provides a functional build up of the computer system. Operations and data flow essential to the test process are tested first. Once an operation has been verified, the operation can be used as a tool to check more complex tests later in the test program.

A GO/NO-GO timer protects against improper instruction sequencing or program hangups. It is a self-incrementing counter and must be reset by the program at least once each 1.3 seconds or a NO-GO error condition will be set. A NO-GO condition is also created when a malfunction is detected by the test program. Isolation of the detected malfunction will be to the computer level.

4.3.3 IBM 9020 Maintenance Philosophy.

The philosophy behind the IBM 9020 maintenance software, which consists of both on-line and off-line programs, is outlined below:

a. Provide malfunction indications by detecting element malfunctions with built-in hardware and/or on-line diagnostic routines.

b. Record in storage (SE) the element environment at the time of malfunction (built-in hardware logout).

c. Interruption to an Error Analysis program for each malfunction to:

(1) Analyze the logout to determine the element or interface causing the malfunction.

(2) Count the malfunctions and record the malfunction rate.

(3) Record logout and other system environmental data on a maintenance history tape, and immediately furnish a hard copy printout.

(4) Retry/restart where practical.

(5) Request reconfiguration when the malfunction is nonclearing to exclude the malfunctioning element from the operational system.

d. Maintenance personnel analyze hard copy printout to determine localized fault area within the malfunctioning element.

e. Maintenance personnel request the Operational Program to provide the minimum maintenance subsystem which is required to run the off-line diagnostic programs determined by hard copy printout analysis. It is preferable for interface testing to include with the malfunctioning element in the maintenance subsystem the particular elements associated with the malfunctioning interface area.

f. If the elements (other than the malfunctioning element) required for the minimum-sized maintenance subsystem are not available because of other malfunctioning elements, use the stand alone maintenance facilities on the malfunctioning element to effect a repair.

g. Use the logouts as the source of information on intermittent malfunctions, thereby reducing the long repair attempts historically associated with intermittent malfunctions. This is accomplished by searching logouts for previous malfunctions made on the suspect element(s) and finding the common denominator of the intermittent malfunctions. This searching is done by off-line data reduction and correlation by maintenance personnel.

h. Use diagnostic fault locating tests (FLT's) and test programs to isolate malfunctions to a few replaceable cards. On elements where it is not feasible to write a fault locating type test, the maintenance programs only exercise the elements and produce outputs that allow rapid isolation of the failing cards by the maintenance personnel.

i. Use scheduled maintenance time to maximum advantage by:

(1) Planned hardware and software tests which serve to maintain element/unit operating levels.

(2) A maximum investigation effort of intermittent malfunctions and unresolved interface malfunctions to determine cause and resolution.

From this, it is evident that maintenance of the IBM 9020 System is built around the philosophy of:

(a) error detection and reporting hardware
(b) environmental saving and logout hardware
(c) off-line error diagnostic programs
(d) highly trained maintenance personnel.

4.4 ARTS III HARDWARE ERROR DETECTION

The hardware errors detected in the ARTS III system are indicated by the interrupt scheme as shown in Figure 4.3. The interrupts which correspond to hardware errors are these: power tolerance, RFDU #1, RFDU #2, memory address parity, memory resume, memory data parity, and background memory data parity. There are two other interrupt types which are used to alert one processor to an error or special condition elsewhere in the system. These are the high priority processor interrupt and the normal (priority) processor interrupt. The following will give a description of each of the above types of interrupts.

Power Tolerance Error

The power protection feature is implemented in both the IOP and the CPM by monitoring the -100 volt dc regulated converter input power. Power interruption up to and including 5 milliseconds duration have no effect on the operation of the processor; however, power interruptions exceeding 5 milliseconds duration cause the processor to generate a Power Tolerance Error interrupt and inhibit class II, III, IV and V interrupts. In the event of total system power failure the Power Tolerance Error interrupt will occur a minimum of 250 microseconds prior to master clear to allow emergency software operations to be performed.

Another effect of the power tolerance feature is that during the marginal power state (-97.0 to -94.0V dc), the manual jump instruction (f=61, j=0) acts as a No Operation instruction. A routine consisting of a manual jump instruction followed by a jump instruction (f=65, j=0) can be used to return program control to the point of interruption if power is restored before a power failure occurs. If the regulated converter input power drops below -94.0 V dc, a master clear will be performed by the hardware.

Another feature of power tolerance in ARTS III is the AUTO START switch. When the AUTO START switch is selected, a return of regulated converter input power to -97.0 V dc or above after a master clear will cause a power restart interrupt. All interrupts (including power tolerance) are inhibited until a jump instruction is executed. When the jump is performed all interrupt scans shall be enabled.

RFDU Interrupts

The RFDU interrupt is generated by the Reconfiguration and Fault Detection Unit when it detects an error in a processor or memory unit. The interrupt is directed to the primary IOP unless the failure was detected in the primary IOP; in that case, the interrupt is directed to the alternate IOP. The RFDU interrupt inhibits all Class III (except for Fault and Breakpoint), IV, and V interrupts.

Intraprocessor Hardware Error Interrupts

The memory address parity error, memory resume error, memory data parity error, and background memory parity error are classified as intraprocessor hardware error interrupts. The memory address parity error is caused by a parity error in the address received by the memory. The memory resume error occurs when the memory fails to respond to a processor instruction or operand request within the allocated time. The memory data parity error is caused by a parity error in memory data occurring during an instruction or operand fetch sequence. The background memory parity error occurs when a memory error occurs during the operation of an IOP hardwired background program (such as the automatic clock update operation).

The above error interrupts are all of equal priority and are serviced on a first come, first served basis. This type of interrupt inhibits all Class III (except for Fault and Breakpoint), IV and V interrupts. This interrupt also inhibits other hardware error interrupts of the same type; for example, when a memory data parity error occurs, other memory data parity interrupts are inhibited. Under program control it is also possible to inhibit all other types of intraprocessor hardware error interrupts.

High Priority Processor Interrupt

This interrupt is an interprocessor interrupt. It is intended to be used for failure detection and recovery functions. The high priority processor interrupt inhibits all Class III (except for Fault and Breakpoint), IV, and V interrupts.

Normal Processor Interrupt

This interrupt is used for interprocessor signalling and communication of a lessor priority than that used with the High Priority Processor Interrupt. The Normal Processor Interrupt inhibits Class IV interrupts.

Other Interrupts

There are certain other interrupts which are intended to detect software errors, but which also aid in the spread of undetected hardware errors. These are the Program Fault, Executive Instruction Error, Write Lockout Error, Read Lockout Error and I/O Write Memory Protect. The Program Fault interrupt is generated by hardware detection of an illegal function code. The Executive Instruction Error is generated by hardware detection of an attempt to execute an executive instruction when the executive designator is set in the status word. The Write Lockout Error is generated by hardware detection of an attempt to write into a memory domain (2048 words) that is write protected. The Read Lockout Error is caused by an attempt to read a read protected memory domain. The I/O Write Memory Protect is generated by hardware detection during an input operation if an attempt is made to write into a memory domain that is write protected.

	Type	Processor
Class I Interrupts		
Power Tolerance	Intra	Both
Class II Interrupts		
RFDU #1	External	IOP
RFDU #2	External	IOP
Memory Address Parity	Intra	Both
Memory Resume	Intra	Both
Memory Data Parity	Intra	Both
Background Memory Parity	Intra	IOP
High Priority Processor	Inter	Both
Class III Interrupts		
Breakpoint	Intra	CPM
Program Fault	Intra	Both
Monitor Clock	Intra	Both
Return to Executive	Intra	CPM
Exec Instruction Error	Intra	CPM
Write Lockout	Intra	Both
Read Lockout	Intra	Both
I/O Write Lockout	Intra	IOP
Class IV Interrupts		
Normal Processor	Inter	Both
Class V Interrupts		
Channel Interrupts	External	IOP

Figure 4.3

ARTS III Interrupt Scheme

REFERENCES AND BIBLIOGRAPHY - SECTION 4

1. "An Application-Oriented Multiprocessing System," IBM Systems Journal, Vol. 6, No. 2, 1967.

2. "Study of Automatic Recovery of Aerospace Multiprocessor Systems," Contract NAS 12-2161, Wynne A. Hyatt, M. Howard Kramer, Robert E. Lyons, and Clayton A. McAdams, IBM Federal Systems Division, July 1970.

3. UNIVAC Interim Report, "The Multiprocessor Executive Design Data," A.A. Westerhaus, J.T. Rye, G.L. Youngsma, W.D. Lahti, Contract No. FA70WA-2289, January 1971.

5.0 RECOVERY AND RECONFIGURATION

5.1 INTRODUCTION

Recovery and reconfiguration are means of coping with an error or failure in the system. Originally, recovery and reconfiguration were done entirely manually; the trend over the years has been to automate the recovery and reconfiguration process piece by piece, until now in advanced systems recovery and reconfiguration have become substantially automated.

5.2 HARDWARE

5.2.1 Error Correction. Error correction is one form of recovery. Error correction is incorporated in a scheme of error detection similar to parity checking. Enough redundant information is transmitted to determine not only whether there was an error in transmission, but also the location of the error. Of course, not all errors detected can be located. For example, one scheme of error coding might detect all one-bit and two-bit errors in the transmission, but only correct the one-bit errors. Error correction is being used in IBM 370 processor storage (main storage).

5.2.2 Error Retry. Error retry has been extensively used as a software recovery scheme. However, it is now being incorporated into the hardware.

5.2.3 Logout. Another hardware feature used in recovery is logout. This is the automatic recording of important registers of an element in a reserved area of memory. Logout is usually performed prior to an impending power failure or after an error or failure is detected in an element. The logout information is used to restore registers when restarting the system or to aid in the diagnosis of the error. Provision in the system is usually made for automatic redesignation of the reserved area of memory in case the memory unit containing the reserved area locations fails.

5.2.4 ARTS III RFDU vs. IBM 9020 CCR. A scheme of recovery in which redundant equipment is incorporated into the system for use when equipment fails has been used in air traffic control systems. These systems seem to be the forerunners of future continuously operating systems. These systems continuously monitor the performance and environment of all devices. Once it has been determined that a portion of the system has failed (because of bad performance) or that a portion may soon fail (because of critical environment), then the system electronically isolates the failed unit from the system, so that the failed portion of the system can be repaired while system operation continues. This electronical

isolation capability can also be used to reconfigure a maintenance subsystem to aid in repair of the failed unit. Electronic accessibility, or connections between all units of different types, is important also. This accessibility guarantees that any operating module can interface with any other operating module. The RFDU of the ARTS system has been discussed from its error detection aspect. The reconfiguration aspects of the RFDU enables it to configure a set of processors and memory in any desired manner, merely by enabling or disabling the line between a processor and memory. The RFDU does not provide control to any peripheral devices, however. Another reconfiguration device, the Level I Redundancy Feature, enables the peripheral devices to be shifted from one IOP to another IOP manually.

In the IBM 9020,Reconfiguration is enabled by the Configuration Control Register (CCR). Each major system element (CE, SE, PAM, IOCE) has a CCR. The CCR specifies to that element which compute elements or input output control elements are valid sources of commands to itself, including commands which change the contents of the CCR. The CCR scheme of providing reconfiguration is more general than the RFDU scheme since the reconfiguration ability is not attached to any one piece of hardware. Also, the CCR scheme includes provision for switching peripherals to an alternate IOCE, whereas this has to be performed manually using the ARTS Level I Redundancy Feature.

Of course, in all these schemes software has to be used to decide, based upon failure indications, whether a failure should be declared and how to reconfigure the system.

5.2.5 EXAM Crossbar Configuration. The configuration of a system is determined by the allowable paths for communication of data and control signals among elements of the system. In the NASA EXAM (EXperimental Aerospace Multiprocessor) System, crossbar switches are used to provide most of these paths. Interrupts do not use the crossbar switch, however; they are controlled by the interrupt director.

EXAM is similar to the IBM 9020 in that when an element fails, it is isolated from the system and a redundant element is used to replace the failed element. EXAM also has the ability of configuring a secondary system for use in the repair and verification of faulty elements.

A major difference between the IBM 9020 multiprocessor and the EXAM system is the data paths between elements. In EXAM, a configuration control register in each element of the system would not prevent a faulty element from interfering with data flow through the memory or I/O crossbar switch unless the CCR prevented the element from talking to other elements. If a "listen to" control in the element is used, a faulty element might slow or stop the system if it tried to communicate with other elements through the crossbar switch.

Configuration control in EXAM can be divided into two parts: the communication through a crossbar switch and an interrupt directed to a specific compute element through the Interrupt Director. In the IBM 9020 configuration control registers perform both of these functions.

In EXAM, storage elements are connected to the compute elements and to I/O processors by the memory crossbar. CE's are connected to the I/O processors and to I/O control units by the I/O crossbar. This connection of elements of the EXAM system is shown in Figure 7. The crossbars are used to decrease the number of wires needed for communication between elements. They contain the addressing logic and hence are a logical place to also put the configuration logic. Two additions have been made to the crossbar switch. They are: changeable logical addresses (to aid in rapid recovery from memory failure), and configuration logic (to allow element isolation and reconfiguration).

An important aid to fast recovery from an ME failure is the ability to change the logical address of an ME. For example, if during a store operation a parity error is detected in a word of an ME, the system can recover by:

a. Moving all of the other words of that ME to another ME.
b. Changing the address of the new ME to that of the failed ME.
c. Configuring the failed ME out of the operational system.
d. Re-executing the store instruction.

Without the ability to change the logical address of a memory element, a software procedure to resolve all the pointers and addresses to words in the replacement element would be necessary, and programs could not use an area of memory as logically contiguous if it spread into two ME's.

The second change made to the crossbar is the addition of configuration control capability. Configuration control was done in the crossbar itself for two reasons. The first is that if the configuration logic were in an element itself and limited to whom that element would listen, the crossbar could be filled with invalid communication, and the system throughput curtailed. If the configuration logic were in an element and specified to whom that element could talk, the system would be dependent on part of a failed element to insure that the failed element would not interfere with the system.

Each data path crossing within the crossbar is useable only when a control flip-flop connected with the intersection is set. A compute element can be isolated from the rest of the system by resetting all the flip-flops on its line to zero. Similarly, a primary and secondary system can be configured by setting to one the flip-flops on the intersections of lines of elements in the same system, and by resetting to zero the flip-flops on the intersections of lines of elements which are in different systems. This configuration scheme isolates the primary and secondary systems from each other.

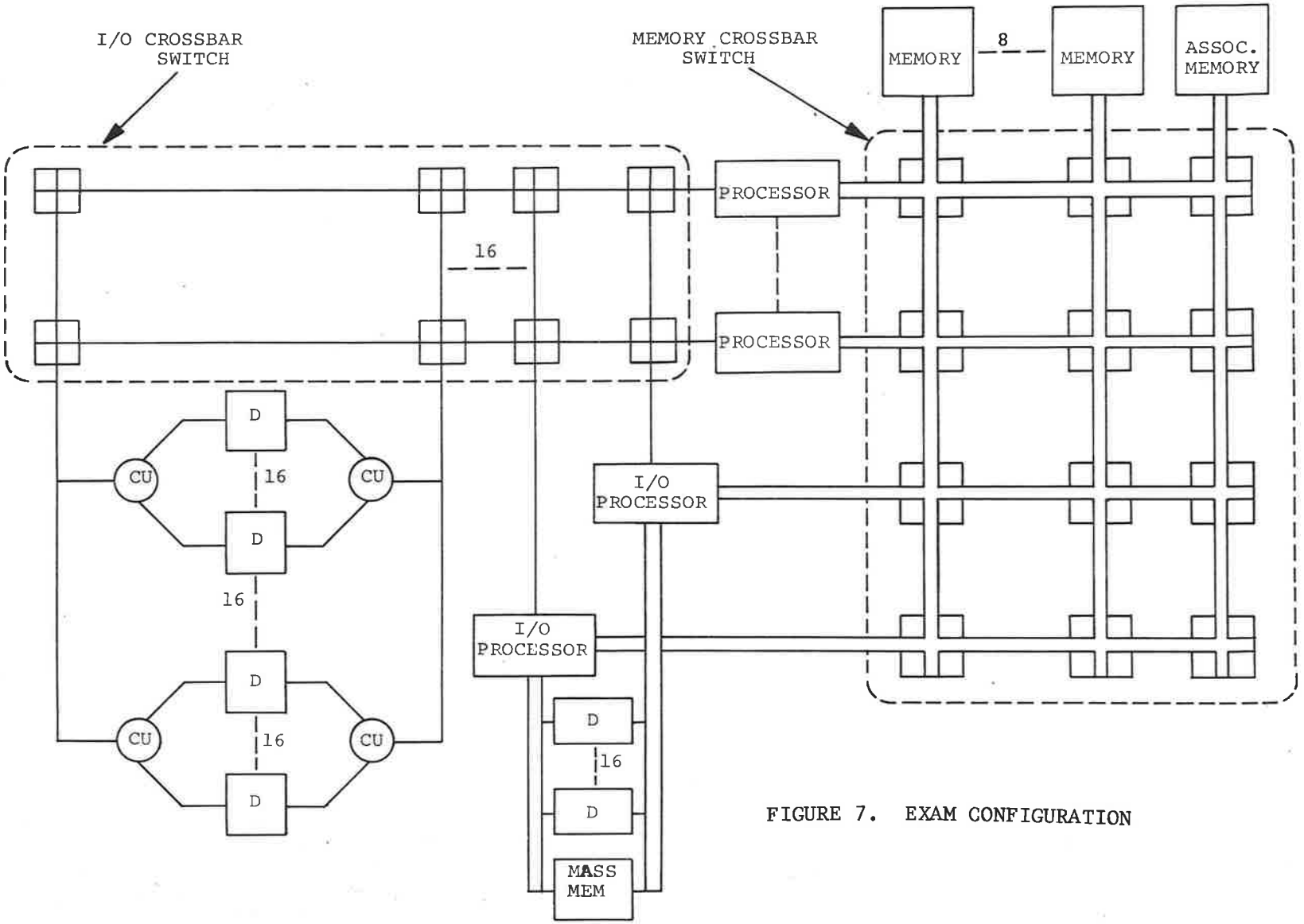


FIGURE 7. EXAM CONFIGURATION

The following restrictions should be placed on the CE issuing a set configuration instruction to insure against the misuse of the instruction.

a. The issuing CE must be in the primary system.

b. The issuing CE must be in the supervisor state.

c. The issuing CE's priority must be equal to the priority of an interrupt that might cause a reconfiguration.

d. The operand of the reconfigure instruction contains proper check bits.

e. Either this CE has not reconfigured the system for "X" seconds, was not the last CE to reconfigure the system, or a second CE vouched for the last configuration.

f. A sequence of instructions to reconfigure the system is used. For example, a set register is used that causes an interrupt in all CE's in the primary system. After a delay, an issue instruction can be accepted. If a CE disagrees with the reconfiguration, it can issue a reset instruction. Resets from two CE's void the reconfiguration.

The above restrictions should make it impossible for a faulty CE to reconfigure the system.

5.3 SOFTWARE HANDLING

5.3.1 Failsoft Strategies. The replacement of a failed system element with another system element of the same or similar type is referred to as failsafe. Under failsafe conditions, the system can continue to operate at full capacity. However, if no back up element is available, another strategy must be employed to keep the system at least partially operational (see Section 3.3.1 for a discussion of failsafe). As long as each of the major system elements remains at least partially operational, the system must be able to carry out some of its workload. Depending upon which element has failed, different failsoft strategies are available.

5.3.1.1 Storage Capacity Degradation. When a memory element (ME) fails, either permanently or intermittently, the storage capacity of the system degrades. If the error is transient, full storage capacity will be restored, but the contents of specific memory locations may be bad and will have to be reconstructed. If the ME failure is permanent, the replacement ME may be smaller than the failed element resulting in lost storage capacity. Finally, no replacement may be available. The following techniques are useful for handling this problem as well as recovering from other hardware failures:

(a) Check-point and restart. The task is re-executed starting at the check-point memory location which was recorded when the failure was diagnosed. Some tasks can start cold from a restart entry point.

(b) Program rollback and startover. This technique is similar to check-point and restart except that recomputation is begun by retracing several steps in the program to a rollback point. Rollback would be useful if a data error occurred.

(c) Recovery delays. The required recovery speed is a critical factor in designing recovery procedures. All tasks that need fast recovery from the point of an error should be designed to be re-entrant. For a slow recovery from the point of an error, slower auxiliary storage may be used for storage of the data needed by an application program.

5.3.1.2 Functional Degradation. If the system is forced to operate with less than its full configuration, selected functions may be eliminated. Some tasks may not have sufficient importance to require special recovery. These tasks may be cancelled or automatically rescheduled.

5.3.1.3 Periodicity Degradation. In real time systems there will be certain fucntions which are performed on a periodic basis. When the multi-processor is forced to operate in a degraded mode, the time between execution of these functions may be extended.

5.3.1.4 Response Time Degradation. If processor capacity has been limited due to a failure, it might be necessary to use re-entrant programs rather than having multiple copies. This would slow down the functions and consequently increase their response times. The extent to which this technique can be employed depends upon the priority of the task involved.

5.3.1.5 Procedural Degradation. Altering the external environment to ease the load on the system will also ease failsoft recovery. Procedural degradation includes such things as restricting the number of human initiated input and output messages, decreasing data rates or sampling periods of sensor equipment, and manually performing selected functions.

5.3.2 IBM 9020 System Recovery Philosophy. The general philosophy for recovery from and maintenance of malfunctioning equipment in the IBM 9020 System is to "replace on-line and repair off-line". The goals of on-line malfunction processing are to:

a. minimize the effect of transient malfunctions on operational processing,

b. minimize the time required for recovery of operational processing following nontransient malfunctions through prompt utilization of redundant elements,

c. acquire and provide timely and sufficient data on malfunctions to permit efficient preventive and corrective maintenance on the equipment used by the operational system.

If at the end of T ms. sufficient OBS checks have not been set to preclude automatic reconfiguration, recovery proceeds as specified for non-retriable checks.

If at the end of T ms. multiple OBS checks have been set which prevent automatic recovery of the operational system, the loss of system power is assumed. The response is to internally housekeep core storage in preparation for a start over, once system power has been restored. Since core storage is non-volatile, program and table reloading is not required.

Inter-Element Transfer Failures. A malfunction which occurs during an attempt by a CE or an IOCE to access storage may result in an element check flag being set for both the accessing element and the accessed SE. If the Error Analysis program is unable to determine which element suffered the malfunction, both elements are logged out. The order of logout preserves the contents of required registers until they have been logged.

If the checks prove non-retriable and if replacement elements are available for reconfiguration under program control, both elements involved in the unsuccessful transfer are replaced by Mode I Reconfiguration. This procedure insures operational recovery and makes the malfunctioning interface between the pair of elements involved available immediately for off-line maintenance.

Should only one of the two replacement elements required be available for automatic reconfiguration, operational recovery is attempted using the one available element to replace the corresponding suspect element. If unsuccessful, the appropriate check report is issued and operational processing terminates.

Multiple Failure Indications. The error analysis program takes action to minimize the propogation of check indications from a single malfunction. All operations which involve attempts to access a failed SE are terminated until operational recovery is achieved.

5.3.3 Use of Associative Memory for Resource Allocation and Error Recovery. One other proposed scheme of providing recovery and reconfiguration utilizes an associative memory to perform the resource allocation function. This scheme is proposed for the NASA EXAM System, which includes a 64-bit by 128-word associative memory. Because the AM (Associative Memory) performs this critical system function, it is an important element in an error recovery strategy, which declares that failed resources are unavailable for allocation, and it recomputes the resource allocation by reallocating nonfailed resources to the tasks requiring them. Critical tasks can then be performed despite resource failures. (Only lower priority tasks are affected by such failures.) This concept provides EXAM with a graceful degradation capability in the event of a resource failure.

Overall EXAM system control is vested in the Supervisor Program, which resides in a memory page and operates through an available CE. The AM resource allocation algorithm acts as a slave to the Supervisor Program. This program provides information (macroinstructions and data) to the AM. The AM in turn provides resource allocation decisions and advice to the Supervisor. The Supervisor can then effect or veto the allocation as appropriate.

Communication between AM and the Supervisor is effected by an AM input and AM output buffer and certain control bits (FF's), which are set by the AM or the Supervisor. Figure 8 shows the control and data paths required between the CE and the AM.

In the EXAM system there are 66 resources to be allocated by the AM. These resources are broken into two classes and four resource types. Units of work for EXAM are subdivided into tasks that must be scheduled and have resources allocated to them. In essence the function of resource allocation is to assign resources to the tasks in order to make efficient use of available resources in task accomplishment.

The AM storage is divided into two tables, the Task Table and the Resource Table. The Task Table contains information about each task's status and scheduling information. Certain tasks are periodically scheduled according to a parameter in the table which indicates how often the task should be scheduled. The Task Table contains all tasks that ever get scheduled, not just the tasks that are currently scheduled for operation. The Task Table can accommodate up to 96 tasks.

The Resource Table is rather novel in design, although this is because it is located in an associative memory rather than a location address memory. A columnar array of 66-bits is used to specify the resource requirements for a specific task. Certain groups of bits are used for specifying the number of a specific class of resource required, while

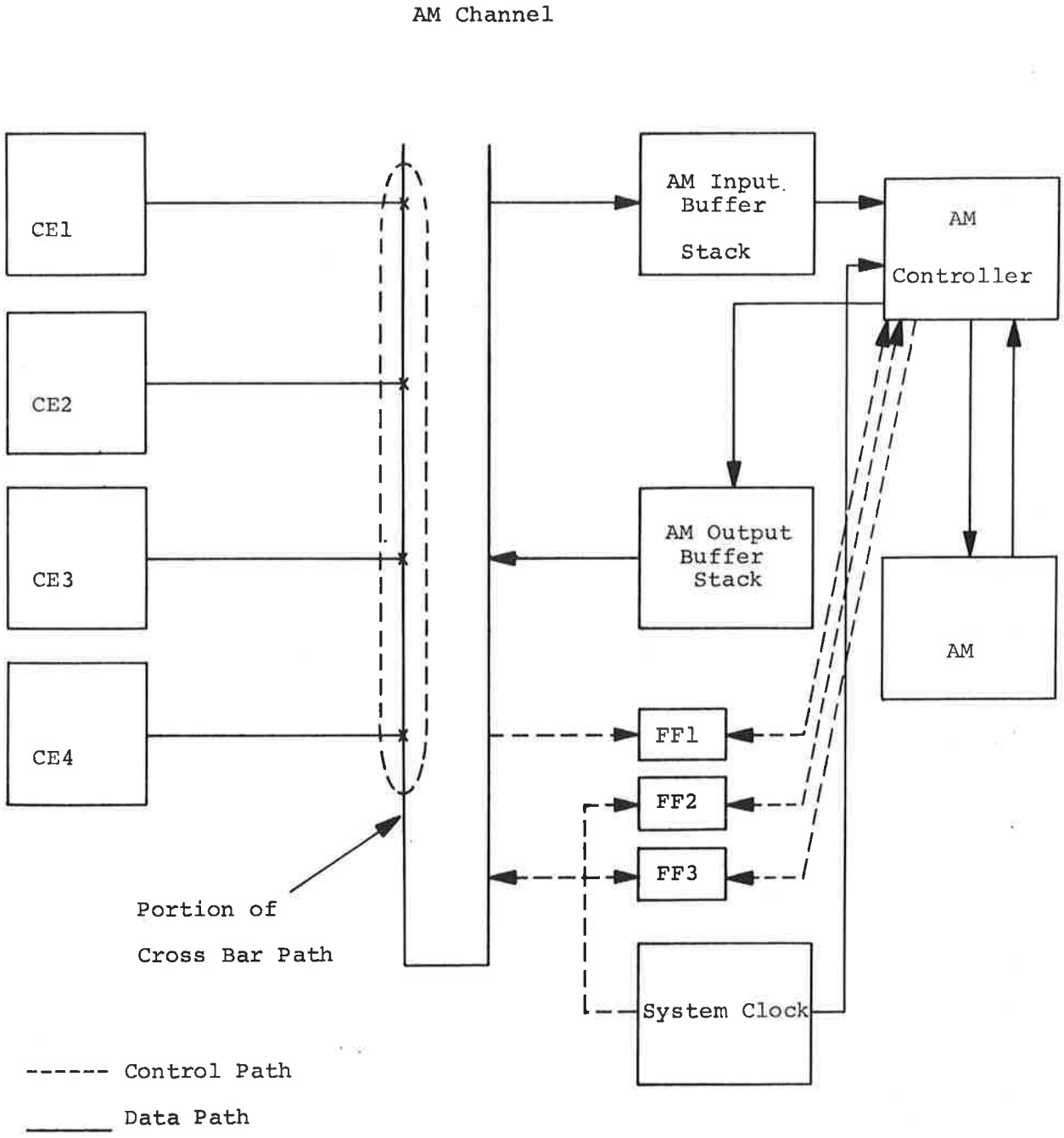


FIGURE 8. Interface Between the Associative Memory and the Supervisor

certain bits are used for specifying that one unique resource is required. For each word in the Resource Table, fifteen bits are available for information about the specific resource it corresponds to: what type of resource it is, whether it is failed, whether it is allocated to a task currently operating.

The algorithm for resource allocation, presented in a report by IBM for NASA, titled "Study of Automatic Recovery of Aerospace Multiprocessor Systems," has a speed of T_{RA}, where

$$T_{RA} = 30.8a + 14.1b + 0.6 \text{ microseconds}$$

where a is the number of tasks in the queue assigned and b is the number in the queue not assigned. The basic cycle time for the associative memory is 400 nanoseconds. This equation makes certain assumptions about the resources and tasks that may or may not be true in another application involving resource application. If the algorithm is used as an incremental allocation algorithm (because of task completion or task addition), then a = 1 and b = 9 (if there is a total of 10 tasks) gives 158.3 usec. If the algorithm is used for total resource allocation, the time required for a = 4 and b = 6 is 208.4 usec. Considering the amount of data manipulation that would be necessary to perform this processing in a nonassociative manner, it does appear that the associative memory is being well-used for the task of resource allocation.

By performing the resource allocation function, the associative memory provides EXAM with a graceful degradation capability (failsoft) in case of failure because it computes an efficient reallocation of nonfailed resources to enable the higher priority tasks to pre-empt resources from less important ones.