Paper 2:

Microprocessors in Fail-Safe Systems

*By Mr. C. G. Shook**

INTRODUCTION

In the creation of fail-safe railway control systems, the designer is confronted with a myriad of choices. This is perhaps especially true when applying a new technology, such as microprocessors, where standards of acceptance are as yet unestablished.

The current point of debate is the choice of hardware and software configurations involved in the implementation of fait-safe railroad control functions using microprocessors. We have chosen to pursue the single processor approach. We are aware, of course, that ether companies (mainly European) follow the multiple processor approach to achieve safety. These appear at first sight to be distinctly different approaches. However, we note that the major design techniques used in the two cases to achieve safety are not that different.

We hold the belief, shared by many, that it is possible to implement safe systems with either single or multiple processor hardware configurations. We further believe that in many circumstances there are distinct advantages to the single processor approach.

The terminology, single processor, as used herein means the employment of a single microprocessor to achieve fail-safe design. While it is possible that such a system may include more than one processor, the reason for inclusion of ether processors is not to create a fail-safe system by having two or more processors perform the same tasks and then checking them against one another. Rather the reason for using multiple processors in our systems will be to allow partitioning of functions, to control the overall system response time, or to improve system availability. This concept is illustrated in Fig. 1.

For contrast, the general multiple processor approach is illustrated in Fig. 2. Here, two or more processors are involved with executing the control logic. Some form of cross-checking is included to determine whether the processing is free from errors. If more than two processors are included, increased availability can also be achieved.

This paper will outline the technical and marketing environment that led to the choices made as well as enumerate the design challenges that had to be met in the creation of fail-safe systems employing microprocessors. It will also give some insight into our choices of solutions, although details of the design techniques used are not included here. These can be found in other papers presented to this Institution and elsewhere.

ENVIRONMENT

There has, for many years, been a fundamental difference between European and North American approaches to safety in railway signalling. North American approaches have, for the most part, been based on devices with intrinsic fail-safe characteristics such as the K and B relays of GRS and the Type P relay of US&S. European approaches by contrast have tended to use non-vita! devices such as metal-to-metal contact relays.

*G.R.S. Rochester U.S.A.

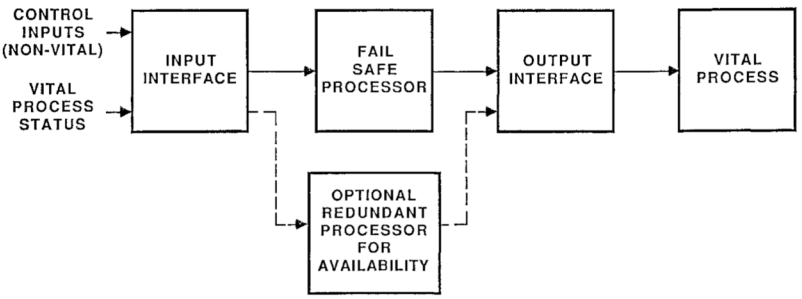

Fig. 1. Single Processor.

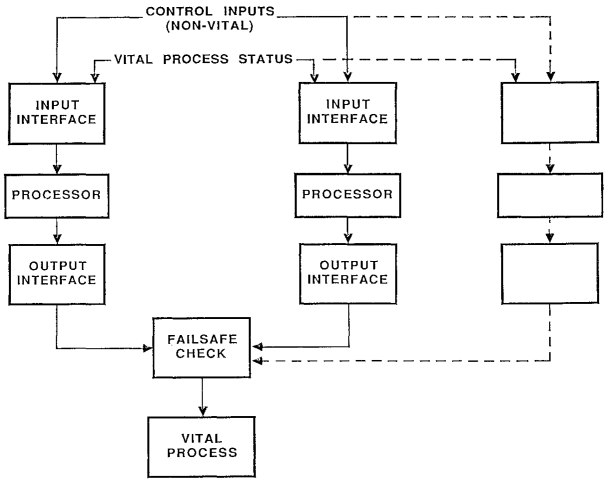

Fig. 2. Multiple Processor.

Safety is then assured by design techniques such as redundancy, back checking, cross checking etc.

Given these fundamental differences that have existed for decades, it is not too surprising that European suppliers would approach microprocessor safety system design using checked-redundancy, cross checking and ether techniques borrowed from the predecessor relay systems. By the same token, it is hardly surprising that North American suppliers would choose the techniques familiar to them, such as diversity, cycle checking, etc., to build intrinsic safety into a single processor system.

With the recent exceptions of Conrail and Amtrak (passenger service), U.S. railroads have been privately rather than state owned. There have always been no less than two, and frequently more, major signal industry suppliers in the U.S. Thus there is inherent competition in the U.S. marketplace with the resulting drive to produce a product which provides the lowest cost/benefit ratio and simultaneously satisfies industry, the individual railroad's, and the supplier's own standards.

There are estimated to be more than 2700 multiple crossover interlockings in North America as well as a substantially larger number of single end-of-siding locations. Large complex interlockings, however, are relatively few in number, being located in and around the larger cities. This track configuration is of course directly related to the large distances spanned by the North American railroads and to the fact that the traffic is and has for many years, been dominated by ft eight rather than passenger service.

This track configuration has further led to the North American practice of vital interlocking logic being housed in wayside locations close to the devices being controlled. Remote control (ctc) is provided from central locations by communication of non-vital information over multiplexed communication facilities.

Much of this is in contrast to the European situation where distances are shorter, traffic density is greater, and passenger traffic represents a much greater proportion of the total. Interlockings are larger with the resulting larger concentrations of controlled devices. Prudent system design in these circumstances has led to larger, centrally located interlocking logic plants, with transmission of vita! signals to and from the trackside.

HISTORY

GRS has been involved with and thinking about the use of processors for executing vital logic for more than twenty years. In 1965-66 we carried out a mainline railroad, radio cab signal experiment called Zone Control. At the time we were not satisfied with the degree of safety provided by the purpose-built digital processor. However, we felt the time was not too far distant when we would be able to satisfactorily use processors for such purposes.

In 1971, we committed to the design and supply of a control system for the Rohr Monocab, a people mover to be demonstrated at Transpo '72, the transportation exhibition held at Dulles Airport near Washington D.C. This control system provided all the features of automatic train supervision (ATS), automatic train operation (ATO) and automatic train protection (ATP). The vital train protection function (ATP) was implemented in dual minicomputers, each executing the same program. A fail-safe comparafor allowed the vehicles to proceed only if the two computers produced identical results. At the time we were not totally satisfied with this approach but had no better solution to offer.

During the exposition in May and June of 1972, the Monocab PRT system under the control of this multiple processor safety system carried more than 10,000 people. Two separate vehicles, operating simultaneously on the same guideway, and three guideway switches represented the potential hazards against which the safety system provided the needed protection.

In 1974, we began work as a subcontracfor on a control system for the Advanced Group Rapid Transit (AGRT) program of the U.S. Department of Transportation. This control system, based on the Monocab demonstration system, also used dual minicomputers for achieving safety.

In November 1977, at an AGRT meeting, much concern was expressed over how to assure the correctness of two complex, identical software programs such as were being designed for the AGRT project. No conclusion was drawn, leaving the uneasy feeling that this would always be an area of risk.

When the primary contractor of the AGRT program decided in 1978, to drop the line of business, our only active connection with

multiple-processor safety applications was terminated. From that day on, GRS has never seen fit to resume vital multiple-processor activity, believing that single-processor methods are preferable.

The use of processors in vital signalling applications is, of course, not necessarily limited to interlocking or train protection logic. There are a number of safety devices in use in conventional signalling systems which are expensive and difficult to manufacture. These, and other devices, can be vastly improved in performance by digital techniques, if the logic can be made fail-sate and cost competitive.

One such device is the motor-driven, vital timer. In 1979 we designed a single microprocessor based vita! timer which was introduced to the market in 1981.

Another such device is the mechanical rate code generafor used for the generation of coded signals in track circuits and cab signal systems. We undertook the development of a single microprocessor based device for this purpose in 1977.

In July of 1978, a single microprocessor based vital speed enforcement governor was field tested on the Airtrans people mover system at the Dallas-Fort Worth Airport.

Trakode II, our vital microprocessor based track communication system was introduced in 1981, to replace the older all relay version. None of these small, microprocessor based safety devices could be made cost effective and competitive with their predecessors if multiple processors were required to be used in their implementation.

DESIGN CONSIDERATIONS

The use of microprocessors, in any configuration, in safety applications, involves generating task logic that is primordially correct and a hardware/software system that will either faithfully execute the task logic or will revert to a known safe state if hardware failures result in execution faults. An additional requirement is the ability to check the hardware 1/0 ports for proper functioning. A further very desirable feature would be the ability to predict, with confidence, the rate at which wrong side failures could be expected to occur.

The fundamental design approach taken today does not differ significantly from that in use for over a century. Namely, one must determine what can go wrong, design in such a way that when the possible faults occur the result is no less safe than if the system were functioning properly.

There are these who argue that microprocessors cannot be adequately analyzed in safety appfications. This is said to be due to the fact that the component level failures within the microprocessor can neither be accurately predicted nor simulated. While this statement may be true, it is also true that such an approach is not necessary. For any failure that may occur internal to the processor hardware, it is possible to identify the complete set of undesirable outcomes. The task then is to design the system in such a way as to protect against these undesirable results of failure.

Approaching the problem from this perspective, one determines that the design must provide adequate assurance that:

- the inputs to the processor are correct
- the program has executed correctly
- the program has not changed in memory
  data tables have not changed
  inputs and variable data are current
  no program segments have been skipped
  the outputs are correct
  the outputs have not been changed by device failures.

A major application of microprocessor safety systems is interlocking control. Fig. 3 illustrates the genera! control process for an interlocking. In this application we conclude that three things are necessary in order to have a safe processor-based system. First, we must have a set of logic expressions that completely describe what we wish the interlocking to do. Second, we must have a means to accurately translate the interlocking logic expressions into software to be run by the processor. Finally, we must have a logic processor that can be relied upon to either process the program exactly as intended or revert to a known safe state if the processing is in any way flawed.

Fig. 4 illustrates the fulfilment of these necessities. For the first part, the primordially safe interlocking logic, we must still rely on the signa! engineer. He must do what he has done since Interlockings first carne into being, that is describe how the interlocking is intended to operate under all foreseeable circumstances. His output is a logic expression set.

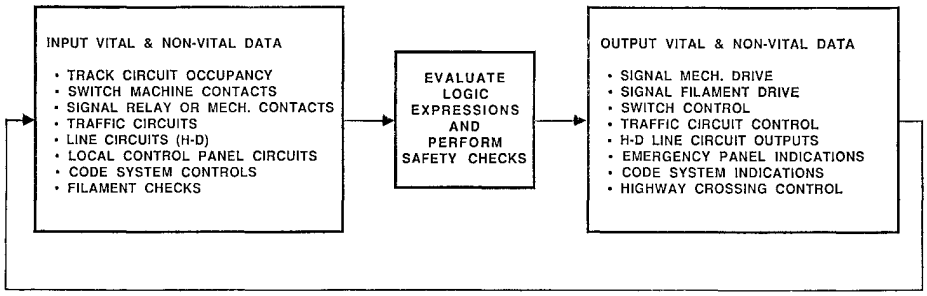

Fig. 3. Interlocking Control Process.

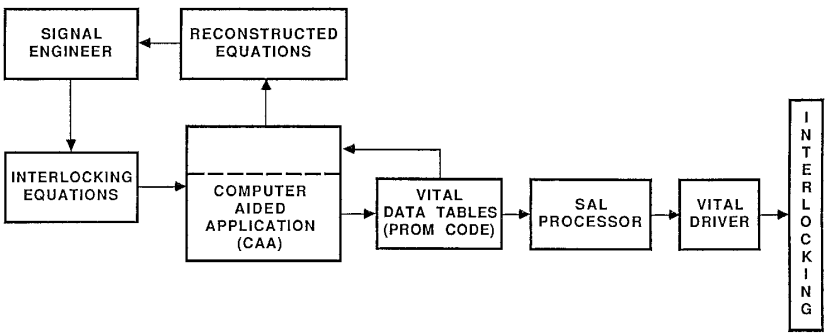

Fig. 4. Interlocking Processor Application.

For the second part, a computer-aided assembly (CAA) package has been designed to assist the application engineer in configuring the system. The CAA program constructs, in two independent diverse channels, the vital data base from the interlocking logic expression set. It encodes testword values, memory location assignments, and expression product term data definitions into PROM code.

To ensure integrity of the vital data base, the CAA check program reconstructs two logic expression sets, one from each of the two independent channels, using only the PROM code as source. These reconstructed expression sets are then checked against each ether and against the original by the application engineer to ensure that the CAA's interpretation of the logic expression set is correct.

For the third requirement, we have developed Safety Assurance Logic (SAL) which proves that the primary logic has been processed correctly, or it does not allow an output to be delivered.

The Safety Assurance Logic verifies the performance of the primary logic by making prescribed tests. These tests are designed to reveal any failures that could result in an incorrect output. The mechanism employed is the generation of checkwords. Each checkword certifies the accurate completion of a group of steps. Correct checkwords are not permanently stored in processor memory. All tests must be completed on every processor cycle and new checkwords must be generated. Therefore, a complete and correct set of checkwords is assurance that all vital tests were made and passed.

The checkwords generated during the processing cycle are passed to an independent arbiter, known as a vital driver. This device is designed sa that the output is allowed to be delivered only if a prescribed dynamic output is maintained. Only a full and correct complement of checkwords, periodically delivered and destroyed after each cycle will allow the correct dynamic output to be maintained.

In essence then, the processor contains two logic systems: the primary logic to perform the interlocking logic, and the Safety Assurance Logic to vitally assure that the primary logic is accurately executed.

A recent variation on the SAL concept is referred to as NISAL or Numerically Integrated Safety Assurance Logic. NISAL is similar to basic Safety Assurance logic in that it uses checkwords to prove the processes have all been performed, the data used is current, the outputs have not been corrupted, etc. NISAL differs from SAL in that the checkwords and data are integrated. Each parameter is represented by a multi-bit word which identities not only the state of the parameter, but also its unique identity. Therefore, for a Boolean parameter there are two applicable words, one for its true state and the other for its false state.

The system is arranged so that the parameter representations are affected by every operation that must be checked. A set of correct values of the parameter representations therefore is proof of correct system performance and that permissive outputs may be allowed to exist.

The probability of erroneously allowing the existence of a permissive output is related to the lengths of the words that represent the parameters. The langer they are, the less likely it is that they will, by chance, turn out to be one of the applicable words when a failure has occurred.

With basic SAL the checkwords can be thought of as generated separately from the data representations, whereas with NISAL the checks are carried within the data representations.

While basic design techniques used in dealing with fail-safe microprocessor systems remain very similar to these which have been used for decades, there is now one more design factor which must be dealt with. Digital and microprocessor systems lend themselves readily to statistical! failure prediction. Formal failure analysis techniques which were developed in conjunction with military and aerospace programs are now being applied to railway signa! systems. One of the advantages of NISAL over SAL is that the probability of wrong side failure is more easily predicted and controlled. The ability to predict numerical rates of wrong side failure now adds this new dimension to safety design, that of deciding, in quantitative terms, what rate of wrong side failure is acceptable.

In the past, judgement was made as to the likelihood of each failure possibility and those deemed to be sufficiently unlikely to occur were dismissed from further consideration. These deemed unacceptable were designed away. The judgement was made on the basis of intuition and experience. This more subjective judgement of the past was easier in many respects to deal with.

CONCLUSION

1 hope I have conveyed the message that the GRS preference for the single microprocessor method of implementing vital logic functions did not develop without some understanding of and experience with the multiple processor alternative.

1 also hope to have conveyed the message that the final choice of hardware and software configuration can be, and aften is, significantly influenced by factors other than just the technical ones.

Finally, 1 reiterate my opening statement that we believe safety can be achieved in either single or multiple processor configurations.

REFERENCES

(1) "Numerically Integrated Safety Assurance Logic", J. H. Auer, Jr., Internal Document 14, May 1983.

(2) "GRS Philosophy on Fail-Safe Design with Microprocessors - Its Development - Its Rationale", J. H. Auer, Jr., J. P. Huffman, C. G. Shook, Internal Document, March 1985.

(3) "Applying the Microprocessor to Grade Crossing Warning Systems", F. Ballinger, AAR, Chicago, October 1981.

(4) "A Review of Jointless Track Circuits", C. Brown, I.R.S.E., Londen, 7th February, 1985.

(5) Letter to the Editor, J. D. Corrie, Railway Gazette International, December 1985.

(6) "The Application of Microprocessors to Interlocking Logic", J. R. Egnot & D. R. Disk, AAR C&S Division, Committee Reports & Technical Papers, 1985.

(7) "Safety Demands Formal Software Engineering", Andrew Goltz, Railway Gazette International, July 1986.

(8) "Microprocessor-Based Interlocking Control - Concept to Application", J. R. Hoelscher, J. B. Balliet, APTA, Miami, 4th June, 1986.

(9) "Methods for Applying Single Computers in a Fail-Safe Manner", European Workshop on Industrial Computer Systems, Position Paper TCSS Number300, Chapter 4.2.

{10) "Microprocessors for Coded Track Circuits", H. C. Nagel, L. R. Hay, AAR Northern Regional Meeting, Montreal, 22nd April, 1982.

(11) "Safety by Redundancy", D. J. Norton, I.R.S.E., London, 7th March, 1979.

(12) "An Analysis of Probability of an Unsafe Failure - and the Mean Time Between Failures in Systems That Uses Numerically Integrated Safety Assurance Logic", Dr. H. E. Rhody, Internal Document, February 1985.

(13) "Fail-Safe Microprocessor Interlocking - An Application of Numerically Integrated SAL (NISAL), D. B. Rutherford, Jr., I.R.S.E., Londen, September 1984.

(14) "The Design of Fail-Safe Processor Systems", R. C. Short, I.R.S.E., Londen, 16th January, 1980.

(15) "Safety Assurance Logic - Using Microprocessors Vitally", H. C. Sibley, AAR, Chicago, 1981.

(16) "Vital Microprocessor Interlocking Demonstration", B. L. Smith, AAR, Toronto, October 1983.

(17) "Cash Savings Force the Pace in Solid-State Signalling", 0. Stadler, Railway Gazette International, September 1985.