Fault Tolerant Fail Safe System for Railway Signalling[1]

Alapan Chakraborty[2]

Abstract: **Railway Signalling is an area which demands the use of ultra reliable fault tolerant system since it is directly related to the movement of passenger trains. In the present article, the first part deals with the general concept used in designing such ultra reliable fault tolerant system. The second part describes a fault tolerant fail-safe system developed for Indian Railway Signalling System.**

***Index Terms* — fail-safe, real time, redundant hardware**

I. General Concepts of Fault Tolerant Fail-Safe System

Real-time computing is one of the most demanding and challenging areas in computing. It is also of great importance, since real-time software is indispensable to all ultra reliable and safety critical applications.

The correct behavior of non-real time systems is founded on *logical correctness* of the results. By contrast, real-time systems require both the *logical correctness* of the result and its *timing correctness*. Logical correctness is expressed in terms of correct and desired functional outputs with respect to its inputs. Timing correctness may be viewed as an extension to logical correctness that includes a unique new dimension, to application requirement, the *real-time*.

Real-time systems may be divided into two broad categories; hard real-time systems and soft real-time systems. In hard real-time systems, timing correctness is critically important and may not be sacrificed for other gains. Typical example is the control system of Nuclear Power plants. In soft real-time systems, timing correctness is important but not so critical. Typical example may be computerized banking, ticket reservation etc.

The nature of computing underlying these systems may vary, ranging from complex numerical computation, as in radar, to relatively simple computations over vast amount of data, as in image processing. The time scales may also vary from application to application, from seconds to milliseconds to microseconds. The main characteristic feature of these systems is, however, not the operational time scales, but the emphasis is placed on real-time in defining the system behavior. Thus, real-time computing does not necessarily imply fast computing but rather, it guarantees the timely response of the system to external events.

The primary concern of conventional design is to make sure that the system or the product functions correctly, assuming that, everything supporting of the functioning of the system also works correctly. However, there are many applications where such confidence in ideal behavior could be badly misplaced. There can be failures in the components or infrastructure supporting the system and in the environment surrounding it. It could be a power failure or electromagnetic interference. It could also be an internal failure due to poor design or a design error. Even with the best endeavors, modern electronic systems are still susceptible to failures. Under such circumstances, as in all other branches of engineering, there must be some safeguards against such failures that ensure the ability of the system to continue to deliver its service, fully or partially, or, in the worst case, ensure the safety of any human life involved the environment and the system itself. Reliability and safety are the additional key attributes, above the logical correctness and timing correctness, which specify stringent design criteria applicable to ultra reliable real-time systems.

Ultra reliable real-time computing has become an important issue in today’s hi-tech world, particularly in the field of guidance, navigation and control systems. Real-time information processing is intrinsic to the operation of all these systems. Early systems emphasized on fault avoidance through rigorous quality control and component engineering to enhance reliability. This proved to be quite satisfactory, although, there was a cost penalty for engineering high reliability into devices by reducing the component failure rate.

Today, reliability and safety is realistically achieved through fault tolerance which is a collection of highly specialized measures such as redundancies, recovery and protection mechanism etc. to assure its attainment. These redundancies come in different forms. They may include redundancies in hardware, redundancies in software and employment of diverse tools and methodologies during the development process. With the advancement of digital technology and microprocessor it has become much more feasible to achieve ultra reliability in real-time systems at lower cost, weight, volume and power consumption associated with redundant hardware.

Redundancy alone cannot ensure reliability or safety of a system in operation. Correct management of redundancy is essential in making a redundant system fault tolerant and fail-safe. Fault propagation, synchronization of and consensus among redundant elements and other redundancy management considerations along with real-time computing are the key issues in designing a fault tolerant fail-safe real-time system.

[1] This work was fully supported & sponsored by Webfil Ltd., Kolkata, India.
[2] At the time of this work Alapan Chakraborty was employed as Development Engineer at Webfil Ltd. He is currently studying in The University of Adelaide, Australia.

II. Safety Requirements and Design Approach

The design of safety critical system begins with the acknowledgement that safety is a supreme design objective considered at the system design level. This may not be traded off in favour of other technical design objectives and cost savings. It also begins with an explicit statement of what safety is i.e. *the safety target*. This involves

a) Identification of hazards involved.

b) What safety measures to be taken in case of undesirable system behavior; *Safety requirement specification*.

c) How tightly the safety measures must be observed. These requirements are expressed through normative probabilistic criteria reflecting the risk levels; *Safety Targets*.

The objective of fault tolerance measures, in the context of safety critical systems, is to localize the effects of faults in such a way that the overall system performance is not affected unduly by any component failure. However, in case of a critical failure, the system fails safely. Such multilevel failure mechanism affected in proportion to the degree of "criticalness of failures" forms the basis of the overall performance and cost-effectiveness of any fail-safe design.

Fault tolerant computers are now used in diverse set of applications and the techniques for achieving fault tolerance vary as much as the application requirements. One way to define reliability requirements for these systems is to specify their acceptable probability of hazardous or catastrophic failure. The typical safety targets normally considered for various applications are

	Application	**Probability of failure**	**Remarks**
a.	Mission Critical Application	10^{-4} to 10^{-6}	Failure would abort the mission.
b.	Military Aircraft	10^{-7} to 10^{-9}	Presumably the crew can bail out.
	Vehicle Critical Application		Cost of failure is huge
	Nuclear Power Plant		Secondary protection is there.
c.	Commercial Aircraft, Airbus A320	10^{-10}	Safety of passengers involved.
	Railway signalling		Wrong signalling may lead to accident resulting in loss of property and human life.

Fault Tolerant principle is also applied in various non-critical applications, not for safety but to increase the availability of the system. Typical examples are

a) Dual computer in on-line transaction processing (OLTP).

b) Dual ring in information networks for protection against link failure.

c) Frequency/Space Diversity Microwave Radio with hitless switching for protection against frequent multi-path fading.

The real-time response requirements for the application under consideration are also very important. For example, aircraft may develop instability if control is not applied every 40 to 100 ms. Similarly, a nuclear power plant may blow up if certain controls are not applied every 20 to 50 ms. In contrast, OLTP applications can withstand a delay of seconds to process on-line transactions. In any event, the penalty for slow response is not a catastrophe.

Another important requirement for fail-safe system is the capability for validation. Validation is not only necessary for qualitative and quantitative assessment of safety but also elevates the confidence level of the system. All safety-critical applications undergo validation by respective authorities before actual commissioning for usage. However, it may be appreciated that failure rate of ultra-reliable fail-safe system is extremely low and lifetime testing for certification is logically not possible. The primary means of validation are a hierarchy of analytical model, simulation, mathematical deductions of reliability etc. Together with this, the empirical data collected from test articles in the lab and field reports constitute the basis of validation.

Safety Terminologies (CENELEC EN 50126, IEC 2)

Fault - A defect either in hardware, software or in the design. Fault is an identified or potential cause of an error.
Faults can be classified into two categories
a. Systematic Faults in hardware and software caused by human error.
b. Random hardware fault

Error - The product state or incorrect information in the system which is liable to lead a failure. In terms of consequences, error can be classified as
– Internal
– External
– Transient
– Intermittent
– Persistent
– Permanent

Failure - Effect of an error. It is the non-conformity of the external behaviour of a component, subsystem or system.
It may be noted that a fault leading to a fail-safe state is not considered as a failure in a safety critical system. For such a system, a hazardous failure is of main concern.

Critical - Single fatality or severe injury.

Catastrophic - Multiple fatality or severe injuries.

Hazard - A physical situation with a potential for human injury.

III. Design Philosophy

The basic principle of a fail-safe design is to identify the fault and mask its effect until recovery measures are taken. Redundancy alone does not guarantee fault tolerance. For a redundant system to function properly in presence of a fault, the redundancy must be managed properly. Redundancy management issues are deeply interrelated to ensure the reliability and safety issues. It also puts a lot of stringent conditions on various timing issues of the software. Redundancy management may even consume more than 50% of the processor time.

As a first step in addressing this issue, it is necessary to partition the redundant elements into individual fault containment regions (FCR). An FCR is a collection of components that operates correctly regardless of any arbitrary logical or electrical fault outside the region. Conversely, a fault in an FCR cannot cause hardware outside the region to fail. Further interfaces between FCRs, must not interfere each other in case of any arbitrary fault. A realistic FCR is normally a whole processor subsystem, called a *channel* in avionics. If the FCR is conceived properly, one can argue that random hardware component in the FCR constitute independent and uncorrelated events. This is an important criterion in predicting the probability of failure of these systems.

IV. Consensus through Voting

To mask errors, outputs of redundant channels must be compared and voted. Two distinct voter approaches have evolved to provide these functions. These methods affect everything from efficiency of fault tolerance and coverage of faults to validation of hardware and software.

The first is the exact bitwise consensus used in most fault tolerant systems. For Railway signaling application, where the information is binary in nature this is the obvious method of voting.

The second approach uses the approximate consensus. Most physical parameters are analog in nature and so the value measured by the redundant hardwares will differ depending on the precision and accuracy. Such parameters are validated by a threshold to arrive at a consensus. However, there is no mathematically precise way to define these thresholds or window of agreement and so most designers use heuristics, guided by opposing requirement. Making the threshold too small generates nuisance false alarms. Again making the window too wide to avoid false alarm may suppress some real faults and lower fault detection coverage.

In any case, to arrive at a stable consensus it is necessary that

- All redundant channels are in identical states.
- All redundant hardware reads the inputs at the same instant.
- Each channel execute the same sequence of operation on the same input / process within a bounded time skew i.e. the time of execution between the slowest channel and fastest channel is defined within a limit.

To satisfy the above requirements, it is necessary that the FCRs are synchronized among themselves by some means for real time concurrency. A popular technique is to exchange semaphores among the FCRs while exchanging the data frames among themselves. An alternative approach is to use a fault tolerant external hard clock to perform the synchronization task. This approach is transparent to application software.

One of the commonly encountered problems is to shield the transient faults arriving from the input or arising from the timing skews. These are handled by validating the data and allowing reasonable time for the data to settle. However, this puts a limit on the consensus time after which the system can be forced to a fail-safe state.

At this point it is worthwhile to mention about a particular type of failure, popularly known as Byzantine Failure, since this has a significant role in the architecture of the fault tolerant system. The analogy between fault tolerant systems and Byzantine generals originates in a famous paper by Lamport, Shostak and Pease [10]. Reliable computer system must handle errors that give conflicting information to different parts of the system.

The Byzantine generalizes problem is set in a hypothetical historical context and concerns an army preparing for a battle. The generals in charge of the battle cannot communicate with each other directly. The technology at their disposal is modest and is limited to messengers. The problem is whether the generals can work out a sensible battle plant in the presence of potential betrayals, either on the part of one or more generals or the messengers involved.

Consider, for example, a scenario involving three generals and a betrayal involving one of them. Suppose that the two loyal generals come to different conclusions. One of them concludes to “attack” and the other to ‘retreat’ and accordingly they both inform each other, as well as the third general of their independently reached decisions. The third, however, communicates conflicting information about his conclusion. He communicates to the first general ‘attack’ and to the second ‘retreat’. As a result, the two perfect ‘loyal’ generals see no reason to change their views, although, the three together could have arrived at a mutual consistent decision, had there not been a betrayal.

According to this theory, to achieve input source congruency in presence of $\boldsymbol{f}$ arbitrary or Byzantine faults, the following conditions must be satisfied.

- The system must consist of $3f$+1 FCRs
- The FCRs must be interconnected through $2f$+1 dis-joint paths.
- The inputs must be exchanged f+1 times between the participants.
- The FCRs must be synchronized with a finite skew.

Byzantine system is particularly important for fail-safe systems which handles analog or continuous variables and are distributed over space. A common Byzantine failure occurs when a marginal bus transmitter or a noisy communication channel causes two receivers to perceive different values for a transmission.

Conflicting information from communication can be virtually eliminated by adopting a rigorous protocol involving error detection codes, checksum, cyclic redundancy checks etc. For railway signaling, all the information is binary in nature (contact status, point status etc.) so that one can apply exact consensus for validating the data instead of approximate consensus required for analog values. Hence for railway signaling application, fault tolerant architecture with 2-out-of-3 voting, instead of 3-out-of-4 according to Byzantine fail-safe principle, is adequate desired degree of safety.

V.Redundancy Management

Redundancy is an additional resource supporting parallel computation of the same process. In fault tolerant design, redundancy helps to increase the level of reliability and safety through consensus. This is normally achieved through a logical mixture of hardware and software, keeping in mind, the contradictory requirements of cost and timing constraints.

Hardware redundancy is the replication of hardware components within a system. It is commonly used for addressing hardware and operational faults and for supporting various forms of software redundancy. Hardware is replicated in units with independent resources such as processing unit, peripheral devices, input/output interfaces, power supply and clock facilities. The objective of hardware redundancy in fault tolerant architecture is to partition the system into fault containment regions such that the non-faulty FCR can operate correctly in spite of a fault in some other FCR.

Hardware redundancy may be managed by two different types of operation - asynchronous and synchronous. In asynchronous operation the sampling of the sensor data, its processing and generation of control signals are done independently in the respective channels. By contrast, synchronized operation work on a common time clock for all the channels for all its inputs, processing and output operation, and of course within a limited time-skew. Asynchronous operation is relatively easy to implement but suffers from one drawback that the process data may be different in different channels due to variation in sampling time. However, for slow varying signals, this is not a problem.

Software redundancy aims at error detection, error recovery and other error handling measures. This may involve

- Multiple computations producing the same result or different results meeting the same objective.
- Supplementary computations for error detection, error handling, data validation, redundancy management, time management etc.

Software redundancy associated with multiple computations on redundant hardware channels may be achieved by means of

- Execution of identical copies of software on multiple hardwares.
- Execution of software which is diverse in design and executed on units of same or different hardware.

The former is a special case of the latter, which is commonly known, is N-version programming. All N-version softwares, obviously, are developed based on same input/output functionality, timing constraints and error handling procedures. N-version programming relies on the hope that, when executed independently or in parallel, alternative algorithms solving the same problem may in consort exposes and trap most of the system failures.

Architecture of fault tolerant fail-safe design depends heavily on the application requirements although it is built around some basic design concepts and techniques. Partitioning of software and hardware and development of the FCRs are still an art which leads to varied cost and performance results, for the same targeted reliability and safety levels.

VI. Development of Fault Tolerant Fail-Safe Block Interface Equipment for railway signalling

Movement of train between two stations is controlled by a pair of equipment called "**Block Instrument**" in railways. Conventional Block Instruments are operated by a simple loop current either positive or negative, sent through copper wires. This current in turn operates some interlocking relays for activating the **Green** signal. The principle of operation of the system is very simple (Ohm's Law), but what is more important is the inherent fail-safety built into it. In case the current loop fails, the interlocking relay logic trips, which turns the signal to **Red**.

Today, the use of copper cable for communication and signalling has a lot of limitations. First, the cost of copper is sharply increasing day by day. Second, its information bandwidth is limited which cannot support the increasing demand of communication of Railway. Last, but not the least, is its maintenance, which is very cumbersome.

Considering the above, Railway is quickly switching over to digital communication through Radio and Optical equipment. This has prompted the development of a fail-safe interface for the block instrument to exchange the information and status between the two adjacent stations for signaling. The inmformation are all digital in nature which makes the interface quite simple and straight forward. However, since it will be carrying signaling information which is directly related to the safety of the passengers, **fail-safety** is of prime importance for this interfacing equipment. The probability of any hazardous failure for such system is targeted at 10^{-10} worldwide.

One can aim at designing the interfacing equipment using highly reliable components like, mil grade electronic devices, connectors etc. but without any redundancy. With this approach, one can achieve a mean time between failure (MTBF) figure of typically 100 years. This is, no doubt, a significant improvement in the reliability figure compared to conventional design using commercial grade components where the typical MTBF is 5-10 years. Even with such improved reliability figure, if one thinks of deployment of 1000 such systems in railway network, then one can expect a probability of 10 failures every year, out of which, quite a few of the failures may be hazardous. It may be noted that it is not the failure which is important. Hazardous failures are the ones which are of prime importance in railway signaling from the point of view of safety.

However, using the fault tolerant architecture and concepts mentioned in the previous section, the probability of hazardous failure of such signaling equipment can be upgraded dramatically to something better than 10^{-10}.

The desired degree of fail safety can very well be achieved by using dual hardware redundancy at all the levels, i.e. from I/O to the processor stage. This is based on the principle that probability of simultaneous failure of a component or subsystem in the same functional area of the dual redundant hardware is typically much has than 10^{-10}. Adopting a simple 2-out-of-2 voting logic, as soon as an anomaly or mismatch is obtained, the system may be forced to a fail safe state. Mishaps are thus avoided, but the movement of the train is also restricted. To increase the availability of the system, meeting the safety criteria mentioned above, a triple hardware redundant system has been proposed.

In railway, there are quite a few types of Block Instruments in operation and the system under consideration was planned to interface all the types. The specification of the Universal Fail Safe Block Instrument (UFSBI) Equipment was derived accordingly by Railways.

VII. Salient point of the system requirement

- Inputs will be from potential free contacts of NO/NC type. 16 nos. maximum.
- Outputs should be above to drive fail-safe signaling relays. 16 nos. maximum.
- The design should have triple hardware redundancy to ensure fail-safety and availability.
- The mean time between wrong side failure (MTBWSF) should be more than 10^9 hrs.
- The inter-block communication protocol should have a Hamming distance of at least 5 to ensure a high integrity of data.
- The end-to-end response should be better than 500 ms.
- The system should have its power supply totally isolated from external power supplies and earth.
- It should be able to withstand the harsh railway environment of temperature, humidity, vibration, dust, surge & transient, EMI etc.
- In case of a failure, the faulty subsystem should be isolated and the system will then run in 2-out-of-2 mode. In case the failure is in two or more subsystem, the whole system must be shut off which will force the signal to RED.
- The system should meet CENELEC fail safety standard EN50128 and EN50129 for software and hardware respectively.

The architecture of the UFSBI system that has been developed is as shown in *figure-1* & *figure-2*.

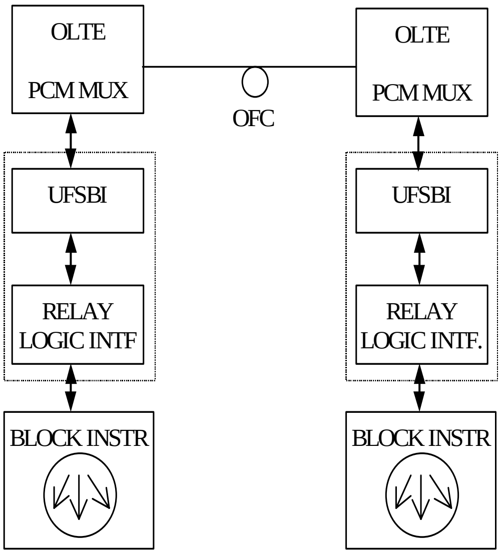

Fig.1 SYSTEM ARCHITECTURE

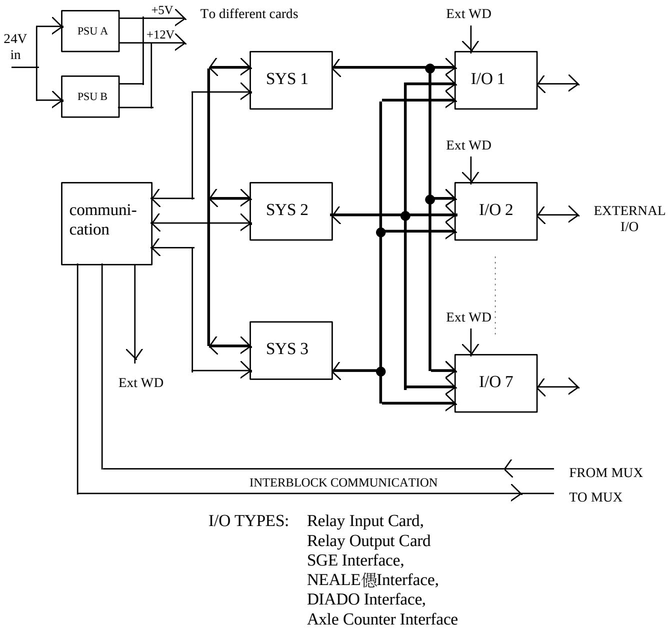

FIG. 2 MODULE INTER-CONNECTION DIAGRAM

VIII. Equipment Operation

It may be noted that the UFSBI system which has been developed basically consists of three fault containment region (FCR), each consisting of Input hardware, processor with its associated peripherals and output hardware. The three FCRs are connected with three dedicated communication links for inter-processor exchange of data for voting.

The status of the Block Instrument is fed through three independent set of interface hardware to three different processors. This information is cross checked among the three processors and depending on at least 2/3 conformity, it is arranged to send to the other end through a serial link. Feedback from the serial output port ensures reliability of transmission.

At the other end, the information of the block instrument is received through the serial link which is fed to the three processors through three different set of hardwares. The information is exchanged among the processors for conformity and each of the processors initiates the output. These activation output signals pass through a 2/3 hardware voter logic and the final output energises a set of Q-series fail safe relay to operate the Block Instrument. Feedback from various levels ensures very high degree of safety for the control outputs.

The transmission between the two block interface equipment is carried out through a built-in data modem operating at 1200/2400 baud. This can be interfaced directly to a voice channel of a Digital Multiplexer connected either to a Microwave or Optical transmission equipment. It can also be interfaced to a data channel of the Digital Multiplexer in RS-232C electrical specification by using external adaptor for necessary ground isolation. Thus the system can be used virtually with all types of communication system and is independent of communication medium.

To ensure better availability of the system, dual redundant power supply has been used. On the transmission side the serial communication interface also utilises triple redundant logic to ensure the high degree of availability.

The system is designed for indoor operation and can withstand strong electromagnetic interface and high humidity. The compact and rugged design of the system ensures its reliable operation even in the cabin room where all the Block instrument inmformation are available at nearby locations. Communication between UFSBI and Microwave / Optical Transmission Equipment is then established through two balanced screened twisted pair cables with 1200/2400 baud modulated signal to remove the interference from noise and common mode induced voltages.

IX. Hardware Description

The equipment consists of some basic modules and Input/Output modules for interfacing various types of Block Instrument.

The basic modules are -

a. Processor Module

b. Communication Driver Interface

c. Power Supply Module

d. 16 digit Alphanumeric Display Unit

The various personality modules are -

a. Input Module (8 change over inputs)

b. Output Module (8 relay outputs)

c. Output Feed back Module

d. Personality module to interface block specific signal

Auxiliary Modules

a. Power Filter Module

b. I/O Filter Module

X.Software Considerations

Redundancy alone cannot ensure reliability or safety of a system in operation. Correct management of redundancy is essential in making a redundant system fault tolerant and fail safe. This is realised through the use of properly designed software with a well-defined architecture and tasks. The basic objective is that the processor, apart from carrying out its normal job of inputting & outputting, should also comprehensively identify a fault within its domain through the various on-line diagnostics, feedback from various stages and comparing its data with others. Once a fault is identified, the particular subsystem (FCR) is forced to a restrictive state (shutdown state) so that it is virtually isolated from the system. Further the fault is highlighted on an alphanumeric display panel to assist the maintenance personnel to quickly rectify the failure.

In this design, the periodicity of jobs have been fixed taking into consideration the response time of the input/output relays. The response time of the fail-safe relays used is typically 100 ms, with a bounce settling time of another 100 ms typical. The inputs and outputs are accordingly serviced every 50 ms. The three FCRs are synchronised by exchanging semaphores through the inter-processor communication packet every 100ms. Using the exact consensus method, under normal situation, the three FCRs arrive at the consensus on a data within 200msec maximum. The system works in certain modes depending on the fault status of the system (see *figure 3*).

The software has also adapted various safety considerations meeting the CENELEC guidelines to improve the reliability.

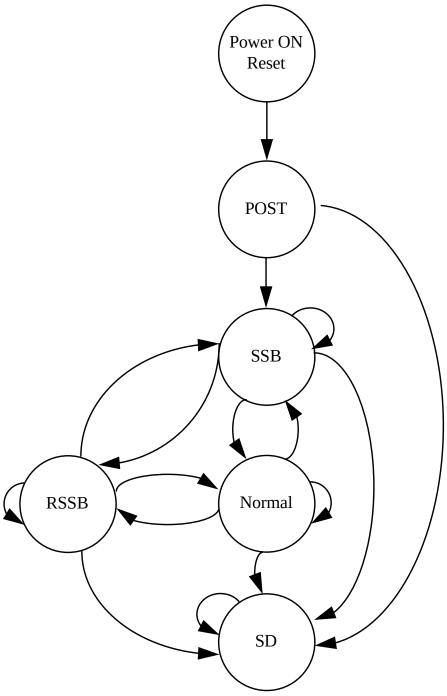

Fig. 3: System Mode

POST : Power On Self Test

SSB : Safe Standby during link failure between the two UFSBI Equipments. Under this condition all the output relays remain in OFF condition which in turn forces the RED signal.

RSSB : Safe Standby due to SSB in the remote station. The output relays in this condition also remains OFF ensuring RED signal.

Normal : This is the normal mode of operation of the equipment when no alarms are there or at least two sub-systems are in operation without a failure.

SD : The system assumes this mode when there is a common mode failure or more than one failure in the system. In this mode all the output relays are switched off to ensure RED signal. The system can come out of this state only through power on reset.

XI. System Statistics

Processor used	:	Intel 80C196KC, 8 Mhz clock
Software size	:	40 Kbytes
Inputs	:	16 Nos.
Outputs	:	16 Nos.

Interprocessor Communication speed	:	9600 baud
Interblock communication speed	:	1200 baud
Process updation period	:	50 ms
Synchronisation period	:	100 ms
Synchronisation skew	:	20 ms maximum

Software partition		
Voting logic	:	25%
Process management	:	10%
Interprocessor comm.	:	10%
Interblock comm.	:	15%
Alarm management display	:	10%
Power on reset and online diagnostics	:	30%

Processor Occupancy		
Redundancy Management	:	15%
Process Management	:	10%
Interblock communication	:	4%
Alarm management display	:	1%
Online diagnostics	:	20%

Mean Time Between Failure (MTBF)	:	10 years
Mean Time Between Wrong Side Failure (calculated by Fault Tree Analysis method)	:	Better than 10^{11} hrs

XII. Conclusion

The system has already been installed between Rourkela and Birmitrapur stations of S.E. Railway for field trial and is under operation since January, 2005. This is the first fault tolerant fail-safe signalling system with triple hardware redundancy which is in operation in Indian Railway.

References

[1] Nissanke, Nimal., *Real Time Systems*, Prentice Hall.

[2] Goldsmith, Sylvia. *A practical guide to realtime systems development*, Prentice Hall.

[3] Chandra, V. and Kumar, K.V., *Reliability and Safety Analysis of Fault Tolerant and Fail-safe node for use in Railway Signalling System*, Reliability Engineering and System 57 (1997) 177-183.

[4] Lala, J.H., Harper, R.E. Alger, L.S., *A design approach for Ultrareliable Real-Time Systems*, Fault Tolerant Computing, IEEE 1992.

[5] Safety Related Electronic Systems for Signalling EN50129, European Committee for Electrotechnical Standardisation for Railway Application (CENELEC).

[6] Software for Railway Control and Protecton Systems EN50128, European Committee for Electrotechnical Standardisation for Railway Application (CENELEC).

[7] The Specification and Demonstration of Dependability - Reliability, Availability, Maintaintability and Safety (RAMS) EN50126, European Committee for Electrotechnical Standardisation for Railway Application (CENELEC).

[8] Safety System validation, Institution of Railway Siganlling Engineers.

[9] Universal Fail-safe Block Interface Equipment, specification No. RDSO/SPN/147/97, Research, Design and Standard Organisation, Lucknow.

[10] L. Lamport, R. Shostak and M. Pease, "*The Byzantine General's Problem*," ACM Trans. Prog. Languages and System, vol. 4, no. 3, pp. 382-401, Apr. 1982.