PERIODICA POLYTECHNICA SER. TRANSP. ENG. VOL. 26, NO. 1-2, PP. 89-100 (1998)

REMOTE CONTROLLED TRAFFIC FOR SMALL RAILWAY STATIONS*

Károly GYENES

Technical University of Budapest
H-1521 Budapest, Hungary
Tel: (36-1) 463-1993, Fax: (36-1) 463-3087
e-mail: gyenes@kaut.kka.bme.hu

Received: July 1, 1998

Abstract

This paper presents the most important aspects of up-to-date computerised solution of the remote controlling for traffic especially for the railway (railroad) application. There will be discussed the hardware and software components and the questions of data transmission. The paper deals with the theoretical and practical methods of fail-safe realization of the instrument and communication.

Keywords: interlocking system, remote control, computers, fail-safe, data transmission.

1. Introduction

On the lines of the MÁV (Hungarian Railway Company) there is a great demand for a high safety computer controlled traffic system operating without traffic personnel at the small railway stations. Today all the technical conditions are given for this. According to our idea, traffic control would be ensured by the train service of the nearest station with the help of an up-to-date, simple and reliable computerised remote control system. This article deals first with the theoretical aspects of the hardware and software elements needed for the development of this system. Second, we investigate the requirements of safety regarding the computerized system.

2. Goal of Development

Our aim is to develop the hardware and software elements required for the realization of the above goal on the basis of fail-safe principle. In addition, the following are to be developed: a safe data transfer system, a simple but

*This subject has been elaborated in detail in connection with the MÁV-OMFB project by the author in 1996.

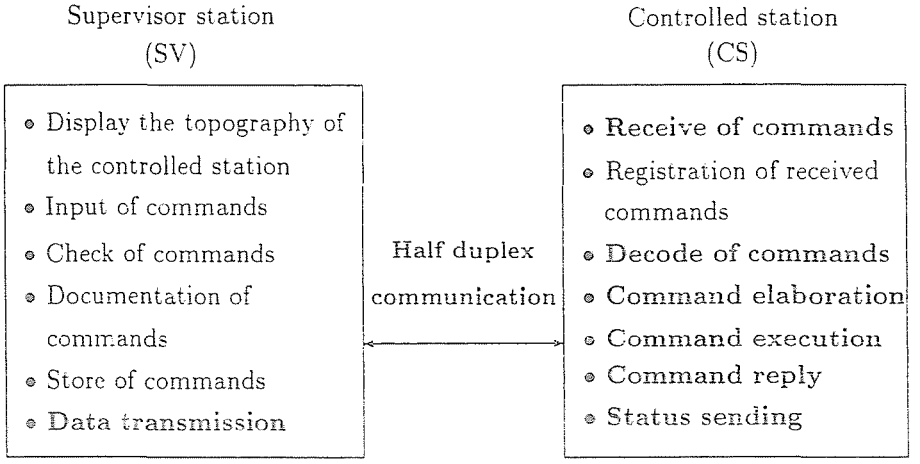

Fig. 1. The logical scheme of the remote controller

clear and handy operating surface, an emergency executive computer system, and last but not least, a program system complying with the principle of diversity (independent programming) .

Those solutions which are independent of the actual station interlocking system should be examined.

The technical, testing and operation procedures for the safe operation should be elaborated, too.

3. Brief Description of the Original Principle

The operation of the system is based on the fail-safe principle of the multi computer system. This solution is encouraged today by the relatively low price of the high capability industrial microcomputers. Today a 2-out-of-3 or a 2-out-of-2 redundant fail-safe system can be implemented at a technically reasonable price. Earlier, in the era of traditional mainframes, these solutions represented a very expensive answer to the problem, so they could not have been afforded as devices for overcontrol.

Today, in contrast, it is rather the salaries of the traffic personnel that incur high costs, therefore it became urgent to find a computerized solution to the problem.

4. Hardware Elements

For the realization of research on a laboratory level, as well as for the execution of safety and reliability examinations with the minimal configuration, the building of a system is required which includes four Advantech or Opto 22 industrial computers. Among them, one of the computers functions as a remote control station which uses a color, high resolution, 20" screen. Another computer performs the function of communication, while the other two serve as logical and safety channels.

The task of the system is to control the traffic of the controlled station without staff on the spot. The commands come from the nearest staffed station. The job of the two sides is explained in *Fig. 1.* The bold text indicates the fail-safe tasks.

In the course of development a number of problems raised can be answered. First of all, the joint reliability and safety investigations into the complex system consisting of traditional relayed devices and the most up-to-date microprocessor-equipped computers can provide valuable results. The operation of such mixed equipment will still be required for long decades in the field of the guided land transport.

These specialists, in turn, can make good use of their knowledge acquired during academic years, for the solution to many transport automation problems.

The problem of documentation (handling registration or registration of performed operations) should be solved on the side of both the transmitter and receiver.

The graphic man–machine interface should be implemented in accordance with the MÁV. To this end, the layout of station topography with security display should be studied, which is already made available by different firms (for example Siemens, Alcatel, Sasib, MÁV, BKV, etc.).

The most compatible way of connecting the outputs of the computer to the relay equipment should be found with the use of high reliability miniature industrial relays.

One possible solution of the hardware is shown in *Fig. 2.*

5. Software Elements

There is a requirement that the software should be modular, and these modules must be as small as possible. The development of the software begins with the system plan, in which the functional description of the individual modules and their interfaces to other modules should be clearly defined.

Applying independent programming is enabled by hardware redundancy. The fail-safe feature can be achieved by the application of these

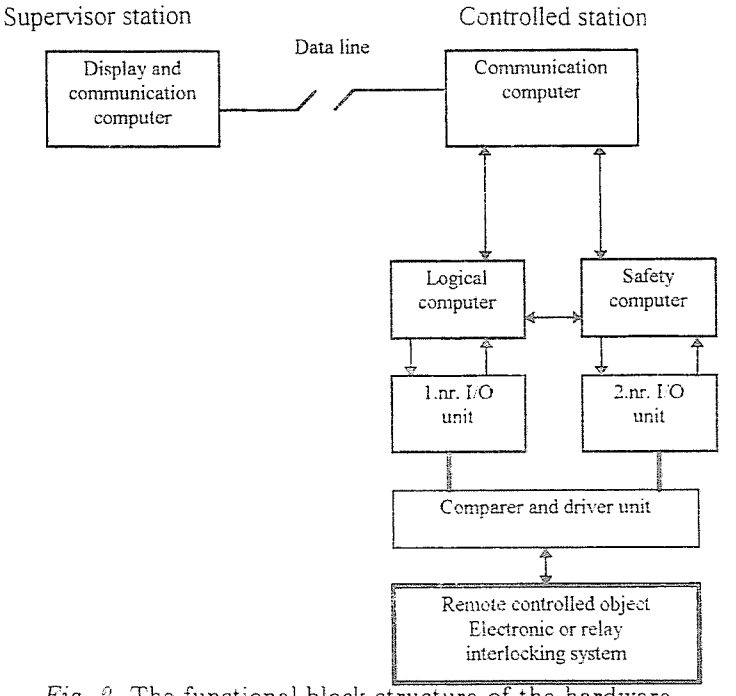

Fig. 2. The functional block structure of the hardware

two factors together. Safety considerations raise the following requirements against the software:

- The first step of software design is the modular system plan. This includes the detailed analysis of the requirement book and the division of the task into smaller functional modules.
- It is necessary to elaborate the mutual acts of the individual modules. The modular interfaces should also be precisely defined.
- The communication between the individual modules can be realized by the separated memory. The best way of intertask communication is by pointer passing. The use of global variables should be avoided.
- The working of the software shall be demonstrated by flowcharts that depict the interfaces of each module to other modules. Also, detailed flowcharts shall be made of the internal workings of modules where description is necessary on the statement level.
- All modules should be testable individually. The interfaces shall be simulated.
- Individual modules, even in the worst case, must never stop the running of the kernel program. When a module depends on external events timers should be used to have time-outs that limit the waiting on the events.
- It is advisable to avoid the use of interrupts.

- Writing recursive segments should also be avoided.
- The use of multitasking operating systems is not recommended either, since it makes the proof of safety very difficult.
- The depth of segment calling should be limited. Return to the kernel program often gives higher safety than the application of highly nested segments.
- The commonly used data base must be handled as a file, but the individual modules can work on their own buffers, common data base can be modified only by the kernel program according to the contents of the individual buffers.
- To increase safety the common data base can be duplicated. When there is difference between the data base and the shadow, the kernel initiates a refreshing cycle.
- The state of data base must be refreshed cyclically depending on the state characteristics of the remote station.
- For storage of the data base *non-volatile memory* (EEPROM, FLASH RAM, etc...) should be used.

The *diversity* is given by two independent programs having the same input, and calculating the outputs. They also calculate the state tables (actual data base) independent of each other. The results should be compared by a third program.

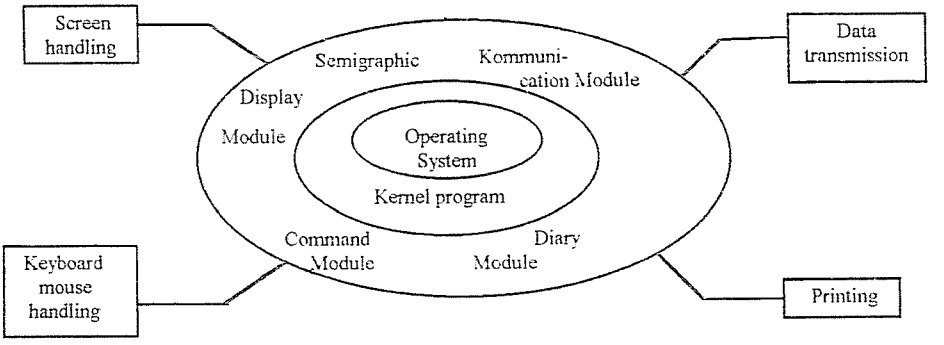

Fig. 3. The structure of the remote control software

Fig. 3 shows the functional structure of the remote control software.

6. Data Communication

The fail-safe data communication system can be realized by application of a hierarchical system. This is defined in the ISO international pattern seven layer model but we suggest the usage of the simplified model for railway purposes, as shown in *Fig. 4*. This can be found in the ORE recommendation 155/RP 10 (see references).

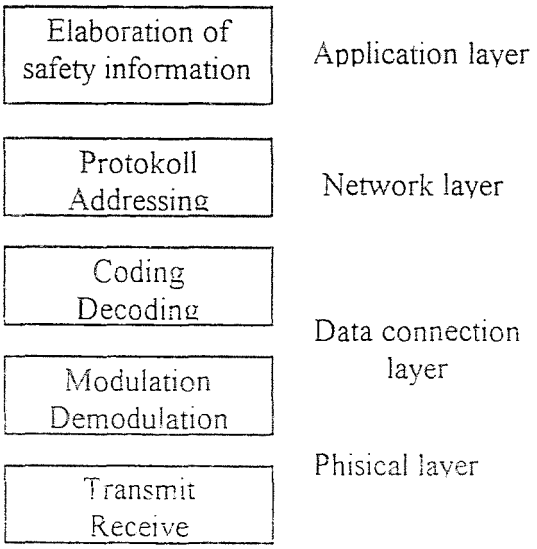

Fig. 4. The simplified four layer model for communication

Due to the necessary safety errors discovered at the lower layer have to be corrected at a higher layer. For example, errors occurred during decoding should not be corrected immediately by using complicated error correction codes which make the error disappear, because in this case the system has no information about the deterioration of the communication line. If there are multiple errors that even the error correction code cannot fix, the data transmission will collapse without any previous warning. If the error is registered by the protocol, and the correction takes place using data transmission repetition, then the system continuously has information about the quality of the data line and can do the necessary steps (call upon maintenance in time).

Safety is based on error discovering. That is why we must select codification, at which any data corruption interference will not be hidden, in other words, *will be discovered.*

This covering method is relatively complicated, the reader can find more details in other articles.

7. Protection Against Errors

In this section we discuss the questions of safety. The most important question is how the undesirable effects of errors can be avoided. This process includes many steps starting from the design, through the realization, to the operation and the maintenance. We briefly discuss the requirements connection with the hardware, software and the operation.

The theoretical and practical aspects of the safety can be described according to *Fig. 5* as follows.

The protection against errors is divided into two parts:

- The **design**, where it must be ensured that it is in accordance with the functional requirements on which it is based.
- The **operation**, where it must be ensured that the consequences of external actions to which the remote control system is exposed meet a specific requirement.

The proof of the fail-safe nature of the design is divided into two classes, which are in turn divided according to their nature:

- Correctness of production:

  For software the program correctness; For hardware, correctness of production is ensured by inspection of the hardware to verify equivalence with the product documentation.
- Functional correctness:

  For software functional completeness is defined as the sum of functional correctness, correct error response, completeness and freedom from side effects; For hardware the same definitions are used but without freedom from side effects.

Safety during operation is divided into three safety classes:

- Defective components, i.e. that a component has not functioned as intended from a given point of time on.
- Interferences, i.e. that for external reasons a component is assigned an incorrect value, although the component's function is not otherwise affected.
- Communication error, i.e. that during transmission between two computers data are corrupted or delayed so much that the requirements concerning correct reaction within a specific time interval cannot be met.

7.1. Safety Requirements

In the following we summarize the most important requirements concerning safety.

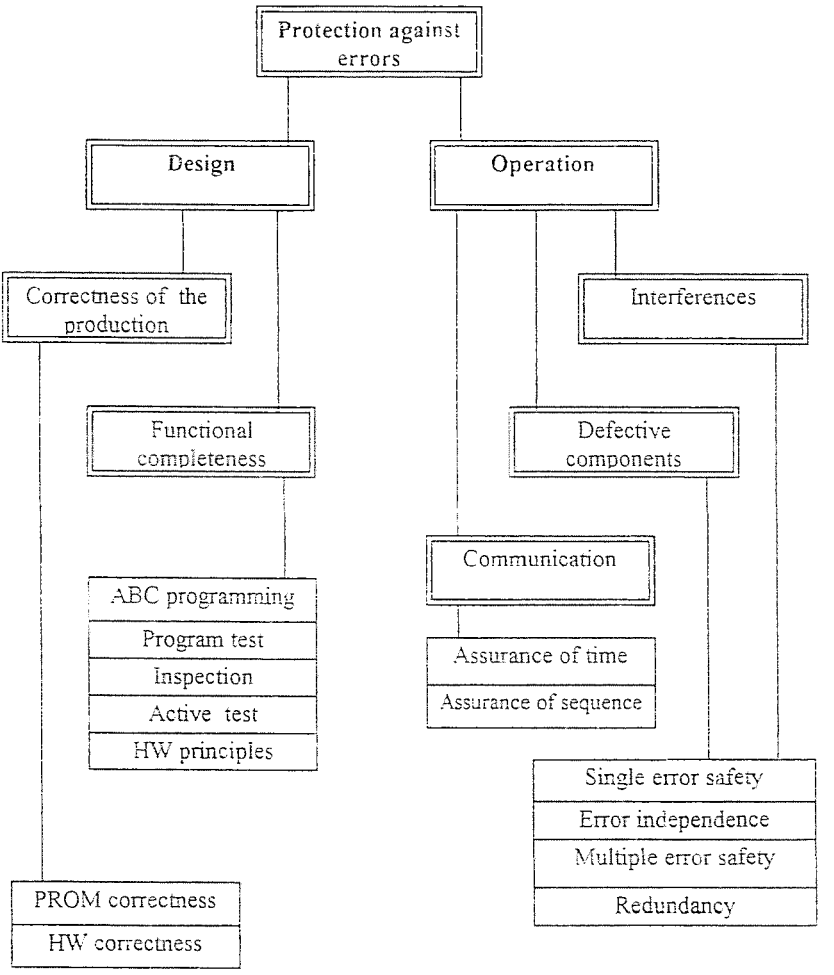

Fig. 5. The logical scheme of the protection against errors

7.2. Definitions and Assumptions

Definition: If a system consists of two or more hardware sections which have the property that a single error in one section, together with a random single error in the other section, cannot cause together a dangerous error, these two sections are called *error segregated.*

Definition: Extensive use is made of the safety principle that one function group monitors another or that the activities of two function groups which, when functioning correctly, must result in identical results, are compared and thus monitored by a third function group. In the following the two function groups whose activity is monitored by one

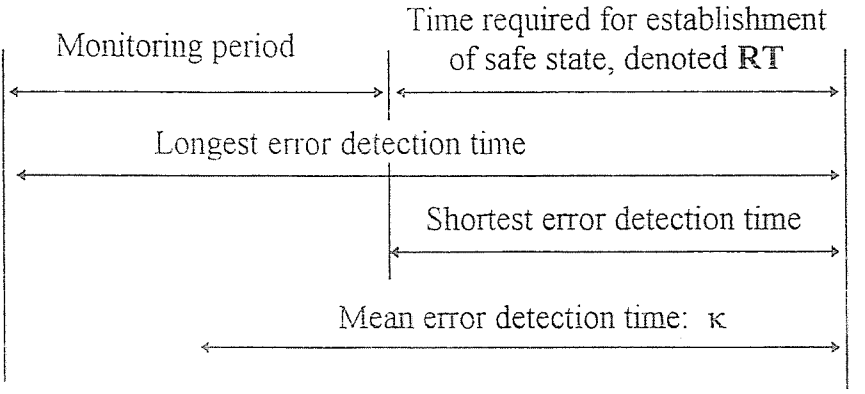

Fig. 6. Calculation of mean error detection time

of these closely related methods, are denoted by A and B.

The time that elapses between two consecutive monitorings is called *monitoring period*, denoted by MP.

Assumption: In the design proofs for the individual hardware sections it is demonstrated that an error E occurring in the function group A will be detected within the monitoring period provided an error has not occurred in the monitoring function group B, or another error in function group A, during the same period which prevents detection of the first error.

If such error E is detected, the system is brought into a *safe state*, so that the error cannot have any dangerous consequences.

Assumption: An error is assumed to occur at random within a monitoring period, which means that the *error detection time* is the average of the shortest and the longest time from occurrence of an error until the system has been brought into a safe state, as shown in *Fig. 6.*

The average is called the mean error detection time and is denoted by κ, which can thus be calculated according to the next expression:

$$\kappa = 0.5 * MP + RT \, .$$

Definition: The mean error detection time is explicitly given for each of the error segregated hardware sections. The mean error detection time for the i-th section is denoted by κ_i.

Assumption: Calculations of data and inverted data which are staggered in time are treated as two function groups. In other words, interference that affects both data and inverted data in the same error critical direction is regarded as two errors.

7.3. Protection Against Single Errors

The system has to fulfill the next requirements:

RS1. A single error and any consequential error must not be dangerous. Single error safety will be demonstrated by defining possible single errors in the design proof for the hardware sections in which there are safety critical components. The design proof contains the argumentation showing that these single errors cannot be dangerous.

RS2. In design it is necessary to apply the principle of error segregated hardware sections.

RS3. If two function groups, A and B monitor each other, the probability of an error in A or B within a given *time unit* (TU) must meet the following requirement:

$$P(\text{error in } A \text{ or } B)/TU < 10^{-8(\dagger)}/\kappa\,.$$

We mention, that in practice if three or four hardware sections are used for mutual monitoring, the requirement concerning the probability of a dangerous combination of errors in a monitoring period is the same as if only two sections are used for mutual monitoring.

7.4. Protection Against Multiple Errors

This part contains an explanation of the method used to calculate how multiple errors in the form of interference and defective components affect the safety of the planned system. The system has to fulfill the next requirements:

RM1. When the first error occurs, the system must move into a safe state. The time elapsing from occurrence of this first error until a safe state has been established is called the error detection time (see above).

RM2. When the first error has occurred, the probability of further errors occurring during the error detection time which, together with the first error, are dangerous, should be less than 10^{-8}.

RM3. The dangerous error frequency in the whole system, regarding all error segregated sections, must be less than once per 10^5 operating year.

RM4. The dangerous error frequency, supposing n error segregated hardware sections can be calculated according the next expression :

$$\sum_{i=1}^{n}\{(P(\text{error in } A \text{ or } B)/TU)^{2^{\ddagger}}\kappa_i\}\,.$$

---

†This value is established by the DSB (Danish State Bahn)

‡This value is established by the DSB (Danish State Bahn)

8. Software Independence

This section contains an interpretation of the requirements concerning independent programming, which should be used in the fail safe-system.

In the error critical parts of the software there are two types of protection against error where software independence is used:

- assurance that the program source code used for arguing functional correctness are correctly compiled;
- assurance that there is accordance between the definition of a function actually performed, i.e. equivalence between design definition and program source code.

8.1. ABC Programming Method

This section describes the scope and methods of independent programming, which is called ABC programming method. This method is able to fulfill the requirements relating to *PROM correctness* and *equivalence* (equivalence with the estimated content).

The PROM (Programmable Read Only Memory) holds the target code of the program. The source program compilation is performed error-free if the decompiled program is *congruent* with the original source program.

This applies provided an error in a compiling tool cannot be cancelled out by the same error in the decompiling tool. This is ensured by using ABC independent compiler/decompiler.

If the program proof includes active testing, both the test procedure and the testing tools must be ABC independent of the program.

The ABC programming is interpreted as follows:
Let APR be a program made by the person(s) A.
Let BPR be a verification program made by the person(s) B.
C is a person who co-ordinates the work of A and B.
D is the approving person(s).
The work of A, B and C is the following:

- A and C must prepare the specification together. When the specification has been completed D must certify that it does not contain implementation details. Only then B starts the actual implementation work.
- A and B must not use the same computers and tools.
- A must not prepare anything in BPR and vice versa.
- A must not speak or write with B on any matter concerning the work/solution of the task.

More requirements in connection with the tools:

- If the compilers used by A and B are not explicitly documented as ABC assured, A and B shall use compilers from different origin.
- Both A and B state which libraries they are using, C shall provide proof that these are of different origin.
- If A or B uses program parts that have not been made specially for this project, both A and B shall explain which libraries and collections of algorithms they are using. C shall establish that these are of different origin. When this principle is applied, A and B can use related languages, as for example language C and $C++$.
- The best solution is when the programming language of APR is different from the language of BPR. For example A uses the PASCAL language, while B uses the C language.

Requirements relating to the communication between the participants of the project.

- All exchange of information between A and B must take place via C in traceable form.
- APR shall be tested by C and A without B.
- BPR shall be tested by C and B without A.
- If the result B's work is test result, C must compare this result of the test with the specification for APR approved by D. If C discovers any divergence between test and specification, then:
  - if the divergence is due to an error in the specification, C shall correct this and pass it to A and B.
  - if there is an error that can be attributed to either A or B, then C shall pass information about the error on to both A and B without deciding where responsibility for the error lies.
- The above shall be checked by D.

In this description we do not deal with the alteration procedure relating to the hardware or software.

References

[1] GYENES, K.: Kis- és középállomási biztosítóberendezések távvezérlésének kialakítása MÁV-OMFB 1996. (Remote Control of the Interlocking Systems of Small and Middle Railway Stations).
[2] GYENES, K.: A biztonsági adatátvitel kérdései a vasútnál (The Fail-safe Data Communication at Railways) *Vezetékek Világa*, 96/4.
[3] GYENES, K.: A vasúti távvezérlés adatátviteli protokollja, (The Protocol of Data Transmission at the Railway Remote Control) *Vezetékek Világa*, 97/4.
[4] GYENES, K.: A CRC blokk kódolás hiba analízise számítógépes szimulációval (Error Analysing of CRC Block Code with Computer Simulation) *Vezetékek Világa*, 98/1.
[5] Transmission of Safety Information ORE red books 155/ RP A 2,8,10,13, 1987.
[6] The Elaboration of the Safety Informations (A biztonsági információk feldolgozása és átvitele) UIC 738 R Recommendation 1992.
[7] Application of Fail-safe Electronic Systems at the Railway, CENELEC Standard Plan 1995.