**FAIL-SAFE MICROPROCESSOR INTERLOCKING**
**- AN APPLICATION OF NUMERICALLY INTEGRATED SAFETY ASSURANCE LOGIC -**

**David B. Rutherford, Jr.**

**General Railway Signal Co.; U.S.A.**

**Summary**

**A new generation of vital wayside interlocking logic is being implemented electronically with fail-safe microprocessor-based systems. The General Railway Signal Vital Processor Interlocking (VPI) system is based on the principles of Safety Assurance Logic , a philosophy developed in 1975 and applied to several commercially available products in which vital processes are microprocessor-controlled. The fail-safety of the VPI system is based on an extension of these principles. The VPI system is essentially a vital Boolean logic processor, enabling the functional logic of the interlocking to be expressed as a closed set of primordially safe Boolean expressions. The expression set may be constructed as a direct representation of conventional relay logic, using existing nomenclature and design principles. Numerically Integrated Safety Assurance Logic allows for the fail-safe evaluation of the set of Boolean expressions designed to represent the logical requirements of an individual interlocking in a "single processor" environment The classical techniques of cycle-checking and diversity are used in this single processor system to ensure fail-safe operation.**

**Basic VPI System Configuration**

**The VPI system replaces conventional fail-safe relay interlocking logic, situated at the wayside, with a microprocessor-based system using vital solid-state input and output circuits to interface with the wayside appliances: switch machines, signal heads, track circuit inputs, and line circuits. It may receive commands via a non-vital communication system from the central office and deliver indications of system status and interlocking operation over the same non- vital link, or it may receive operational commands locally (Figure 1).**

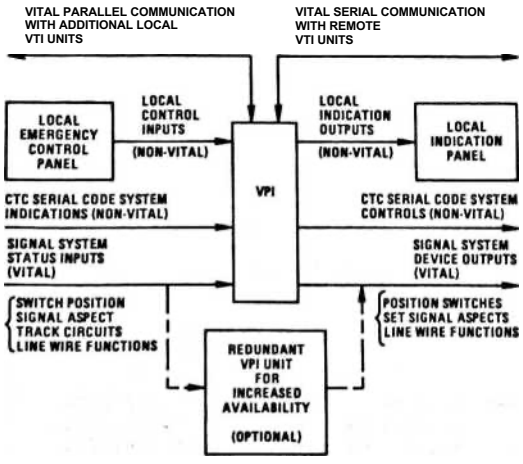


**Figure 1. VPI system configuration.**

**The core of the VPI system is a vital Boolean logic processor. Since the logical functions inherent in a railroad interlocking can be reduced to a closed set of Boolean expressions, a general purpose Boolean processor that evaluates those expressions vitally is an efficient vehicle for interlocking control. It also offers the advantage of being able to implement the interlocking logic using expressions that are a direct representation of existing relay logic designs, designs that have become standards for individual railroads by proving themselves safe and reliable over the past 50 years.**

**The states of inputs from the field (switch position, track occupancy, signal status, and line circuit inputs) are vitally sensed via solid-state input circuits.**

**Vital solid-state output circuits are used to position switches, set signal aspects and energize relays and line circuits in accordance with the interlocking logic rules. The solid-state output circuits are designed to provide vital assurance that any system failure that attempts to issue an improper permissive output will be immediately detected, and power to the driven output device will be disabled before its effect is felt by any wayside appliance.**

**The states of the vital outputs are determined by evaluating the set of Boolean expressions describing proper interlocking operation. The expression set should not only provide for efficient operation of the interlocking, but must be "primordially safe." The term "primordially safe" means that if the Boolean expression set is evaluated correctly and without failure, safe operation of the interlocking will result. The expression set is designed by the signal engineer or application engineer to satisfy the individual railroad's safety standards and operating procedures.**

**The VPI system can generate any number of vital time delays. The time delays are implemented by a set of expressions with special properties; however, their evaluation follows as a natural extension of the way in which all other logical expressions are evaluated. These time delays are identical to those performed by conventional motor-driven timers or vital solid-state timers. The time delays may be either switch-settable or permanently programmed or both. The range of each vital time delay is from zero to 59 minutes, 59 seconds in one second increments.**

**The VPI system also can emulate any non-vital code system. Centralized traffic control commands sent via the code lines are inserted directly into the system as expression parameters. Indication messages, which can represent the states of any vital input, output or expression parameter, are internally constructed and then delivered to the non-vital code system in the appropriate code format.**

**Non-vital parallel inputs can be accepted as expression parameter values (from local emergency control panels, for example). Likewise, non-vital parallel outputs may be used to illuminate local emergency control panel lamps, maintenance procedure indications, etc.**

**Vital emergency-working functions can be designed directly into the interlocking logic expression set to enable maintenance personnel to block switches, lock routes, etc., via specified vital local inputs.**

Communication between several remote VPI systems distributed throughout one large interlocking or situated within several interlockings is possible via vital serial communication links. This feature allows large interlocking plants to be divided into sections so that the VPI system controlling the section is in proximity to its wayside hardware. Vital data are passed between VPI systems over a standard data link, such as a simple twisted-pair (Figure 2).

The VPI system is modular. It is expanded by adding functions to the expression set and inserting additional vital input and output groups.

## Principles of Operation

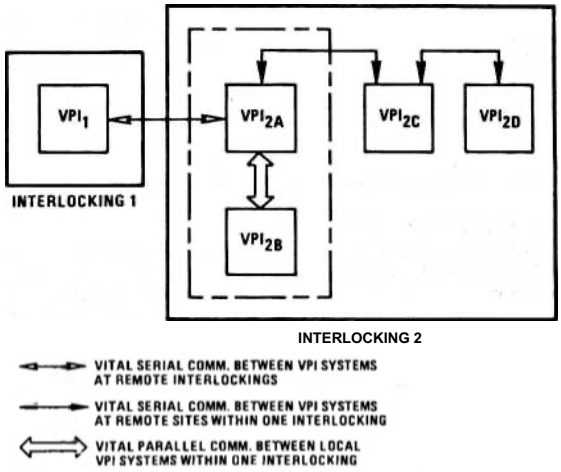


**Figure 2. Vital communication between local and remote VPI systems.**

The microprocessor-based vital Boolean logic processor core is a single-processor system that processes vital data in two diverse, independent channels. The two diverse independent channels are software channels, each of which contains data representing the state of each parameter, but encoded in diverse forms. These two forms of data representation must correspond exactly for permissive outputs to be allowed. A single-processor system does not need more than one processor for safety assurance. Although there are a number of separate processors in the VPI system, they are required to divide the work load rather than provide safety checks on each other. System safety is ultimately assured by a directly acceptable device.

## The VPI System Cycle

Figures 3 and 4 illustrate the vital processes occurring during the system cycle. One VPI system cycle consists of sensing and encoding vital and non-vital input parameters, determining the states of the vital outputs by evaluating the entire Boolean expression set using the current values of the input parameters, and driving the vital output ports to their proper states. The system cycle is exactly one second. During the cycle, the states of the vital output ports are checked every 50 milliseconds (ms).

## Determining the Vital Input States

The state of each vital input (TRUE or FALSE) is determined at the beginning of each system cycle. Each input state is represented as a parameter value in each of the two independent channels. These parameters constitute "vital data" within the system. Vital data are data representing vital parameters that have been encoded to include unique parameter identification within a separate polynomial code assigned to each channel. Unique codewords are assigned to the TRUE and FALSE states of each input parameter for each channel.

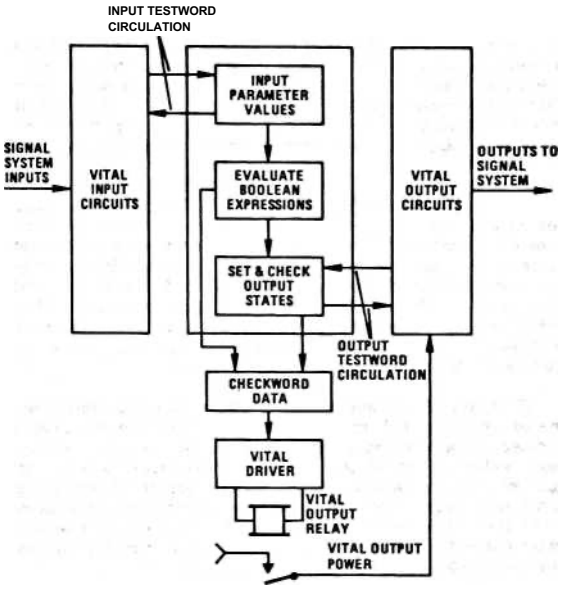


**Figure 3. VPI vital processes.**

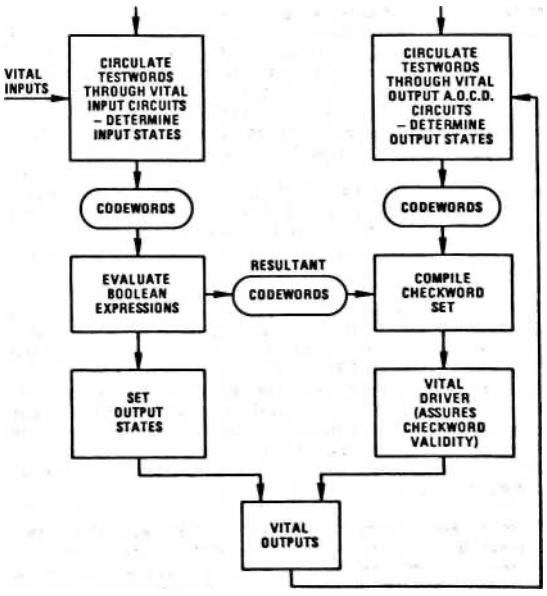


**Figure 4. VPI vital processes.**

The states of the vital inputs are then sensed by circulating a testword through each vital input circuit for each of the two diverse independent channels. The testword has properties peculiar to the TRUE codeword associated with the input parameter. The vital input circuit is designed so that the testword cannot be circulated through the circuit unless the input is energized. The testword is circulated serially through the input port circuit represented by that parameter.

If the input is energized, the testword is transformed and returned as the TRUE codeword. It is stored in an identity-sensitive location in system data memory and represents the state of the input parameter as TRUE. An identical operation on a diverse testword is performed for the second channel. In the event that the circulated testword has corrupted or was unable to be circulated because the input was not active, the assigned FALSE codeword value is inserted in the memory location.

Thus, the TRUE state representation of each input parameter is protected in three ways. First the codewords themselves are encoded with unique identification. Second, each value is stored in an "identity-sensitive" location in system memory, and third, different representations of the same parameter exist in each of two independent channels. Note that TRUE parameter codeword values appear nowhere in permanent machine-readable memory, but must be built by the input process during each system cycle.

Note that non-vital input parameter values are also determined at this point in the system cycle; however, they are encoded by direct assignment of the appropriate TRUE or FALSE values, not by testword circulation through a vital input circuit.

Expression Evaluation

After all parameter states have been encoded into the two channels, the expression set is evaluated using the encoded data. The Boolean expression set that describes the logical operation of the interlocking contains expressions of the form:

X = (A*B*C) + (D*/E*F*G) + .....

Where "X" is the result, (A*B*C) is the first product term, consisting of parameters A, B, &. C, and (D*/E*F*G) is the second product term, consisting of parameters D, E, F, 4 G. Expressions written using this format are said to be in "sum of products" form. The first product term is said to be TRUE if all of its parameters *A,* B, 4 C exist in their TRUE states (here the "*" symbol indicates "logical and"). The second product term is said to be TRUE if parameters D, F, 4 G exist in their TRUE states and parameter E exists in its FALSE state. Here the "/" symbol denotes the false state of the parameter which follows (E in this case). The expression result "X" is said to be TRUE if any of the product terms is TRUE. (Here the "+" symbol indicates "logical or".) In the event that no product term is TRUE, the expression result is FALSE.

Expressions are evaluated using a mathematical process that operates on the encoded values of the parameters within a selected product term, and each product term is evaluated sequentially until a TRUE term is found. The result is the unique TRUE codeword representing this expression result if, and only if, the parameter values used are the correct parameter representations indicating the correct parameter states, and are operated on in the correct order. The correct TRUE codeword value is not known to the system; however, the FALSE value is readily accessible.

Again, each expression is evaluated in each of the two channels using that channel's parameter values, and the results in each channel are codewords in that channel's unique polynomial code. If the expression is determined to be false by a cursory look at the parameter states within each product term, the FALSE codeword value is assigned to the expression result.

There is no practical limit to the number of product terms contained in an expression or to the number of parameters contained in any product term. The parameters in an expression may represent the states of any of the following:

- vital inputs
- non-vital inputs
- results of previous expressions
  (evaluated during the current system cycle)
- results of subsequent expressions
  (evaluated during the previous system cycle)
- the result of the current expression
  (evaluated during the previous system cycle)

Note that the use of expression results evaluated on the previous system cycle allows the expressions to emulate "self-latching" relay configurations.

The entire expression set exists in each of the two diverse independent channels. Parameter values used in evaluating channel 1 expressions are members of the channel 1 codeword set and are located in identity-sensitive locations in channel 1 data memory, and likewise for channel 2.

Driving the Vital Output States

After the expression set has been completely evaluated in both channels, those expression results that directly determine the states of the system's vital outputs are examined, and the output circuits are driven to the corresponding states.

After the vital output states have been established, the system constructs the non-vital indication messages that are then transmitted serially on the code system communication lines in the proper code format or transmitted in parallel on non-vital output ports.

The VPI system cycle is now complete. The duration of the cycle is one second. However a "vital recheck" cycle occurs every 50ms during the one second VPI system cycle as explained below.

The Assurance of Safety

Safety Assurance Logic

Safety Assurance logic is a set of design principles used in the design of fail-safe processor products or systems using a single processor with two logic systems incorporated into the program. The primary logic performs the necessary task (executes interlocking logic, for example) and the Safety Assurance Logic proves that the primary task is performed correctly, or does not allow an output to be delivered. The most simple processor-based system comprises a central logic unit and the necessary devices to send inputs to it and deliver outputs from it. The vital processor-based system has, in addition, Safety Assurance Logic and design attention directed to the safety implications of its connections to other safety systems or appliances.

The primary logic is designed to be safe, but it, in itself, cannot be used as a fail-safe system. Safety Assurance Logic must be added to verify that:

- the inputs to the processor are correct
- the program is executed correctly
  - the program has not changed
  - data tables have not changed
  - inputs and variable data are current
- the outputs are correct
- the outputs have not been changed by device failures

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. Checkwords are generated, with each word conveying vital certification of one feature of vital performance. Checkwords do not exist in permanent processor memory; therefore, a complete complement of checkwords is assurance that all vital tests were made and passed. All tests must be repeated on every processor cycle and new checkwords must be generated.

The "watchdog" of the system is the "vital driver." This is a processor-based device that assimilates the complement of checkwords each cycle, and generates a dynamic signal of an exact form and frequency. The dynamic signal is then passed through a vital analog frequency decoder that assures that its form and frequency content are correct and generates a DC voltage, which in turn picks a vital relay. Closure of the vital relay's front contacts provides the only source of power to the vital system outputs. Thus fail-safety is ultimately assured by a directly acceptable device whose nature is completely different from the processor-based system it is designed to protect.

A full complement of checkwords enables the vital driver logic to produce its dynamic output for only a vitally-limited time. The production of the dynamic output not only verifies that every checkword is correct, but also that the checkwords are vitally destroyed as a necessary condition for the dynamic output to exist.

In the case of the VPI system, a full complement of checkwords is delivered every 50ms. The checkword set enables the vital driver to produce only 50ms of dynamic output. Unless constantly refreshed

with new checkword data each 50ms, the dynamic signal ceases and the vital relay opens its front contacts, removing power from the vital outputs.

Numerically Integrated Safety Assurance Logic

Systems, such as VPI, differ from small single-purpose, processor-based products in that the process of accommodating the unique variations associated with each application is more involved. Normally, the instructions and data in the permanent processor memory of a single-purpose device are not changed from application to application. With VPI, however, each application requires its own unique set of expressions and parameters, which leads to a unique data pattern in the associated microprocessor memory.

The implied need to redesign the Safety Assurance Logic for each new application has been eliminated by the development of a variation of the Safety Assurance Logic concept in which the vital checkword structure is woven throughout the data. Instead of checkwords being built in parallel with the processing of the data to prove that the processing has occurred, the checkword information is built into the data structure. The bits in each checkword are made to depend upon every parameter and process upon which the permissive output depends. Any checkword built from incorrect data or incorrect processing will be affected.

This variation allows all of the program instructions (routines) to be designed as universally applicable to all interlockings. The application-dependent information is contained in a data base. In this way, all of the program may be written at the outset and thoroughly debugged. Once this is done, it is only necessary to create a new data base for each new application. The Safety Assurance Logic, which assures that the system performs correctly or not at all, is imbedded in the unique data base. This variation is known as "Numerically Integrated Safety Assurance Logic."

The Vital Output Recheck Cycle

An important application of Numerically Integrated Safety Assurance Logic in the VPI system is the assurance that vital outputs are in their non-permissive states unless specifically allowed to be permissive by TRUE expression result values in both channels. This is accomplished by executing a "vital recheck" program every 50ms. The recheck program circulates unique testwords through the vital "absence-of-current detector" (a.o.c.d.) circuit associated with each output. This vital circuit allows a testword to be circulated only if the current flowing to the output is zero (below a vitally established threshold that is sufficiently low so that the driven appliance will not interpret it as permissive). If testwords that have been transformed into TRUE codewords are returned through the a.o.c.d., they are proof that the ouptut is in its non-permissive state. If TRUE codewords are not returned intact, the output is assumed to be in its permissive state.

A complete set of checkwords validating the states of all system vital outputs is thus generated. If the set is correct, it provides vital assurance that only those outputs which correspond to expressions evaluated as TRUE are in their permissive states. A correct set of checkwords permits the vital driver to generate 50ms of vital dynamic output.

Another important application of Numerically Integrated Safety Assurance Logic in the VPI system is the vital erasure of variable data buffers each system cycle (one second). On completion of each system cycle, the data buffers that contain input parameters and expression results are vitally erased. This is necessary to assure the use of current data every cycle. In the case of expression results used as "self-latching" parameters, vital erasure assures that these results have been evaluated on the previous cycle. The vital erasure process produces a checkword set that is delivered to the vital driver at the beginning of each one-second system cycle. This checkword set must be correct to enable the vital driver to produce its dynamic output for each of the recheck cycles in the subsequent system cycle.

Applying VPI to an Individual Interlocking

Application of the VPI system begins with the definition of the input and output interfaces (vital and non-vital). Vital inputs are of one type only, while there are a variety of vital output types. They include single- and double-break, low- current outputs and high-current, single-break outputs for driving signal aspects. Vital inputs and outputs are available in modular groups and are inserted into the system as needed. Non-vital I/O definition includes the type of serial code system to be emulated, as well as parallel configurations.

The application engineer then produces a "primordially safe" set of Boolean expressions which logically describes the proper operation of the interlocking. These expressions may describe the interlocking operation in a manner identical to that currently used in designing vital wayside relay logic - using the same nomenclature, the same functional logic principles, and the same logical configuration.

A "computer-aided assembly" (CAA) package has been designed to assist the application engineer in configuring the system. It includes a "logic simulation" facility that enables the Boolean expression set to be exercised, simulating functional interlocking operation. Syntax-checking features in the CAA crosscheck parameter and I/O names for consistency and report discrepancies. An easy-to-use editor is available for syntax correction and expression changes. The CAA documents the physical configuration of the VPI system modules and produces wiring lists to facilitate wiring to external interlocking circuits and appliances.

Safety Considerations within the CAA

The most important responsibility of the CAA program is the construction of the vital data base from the final primordially-safe Boolean expression set. Encoded testword values (which are to be transformed into TRUE codeword parameter values by circulation through vital input and output circuits residing in their appropriate states in actual operation), directly readable FALSE parameter codeword values, identity-sensitive memory location assignments, and expression product term data definitions must be assembled into PROM-based code in each of the two diverse independent channels in exactly the correct form. To insure the integrity of the vital data base, a CAA CHECK program is run off-line.

The output of the CAA CHECK program is two expression sets reconstructed using only the PROM-based data and the assigned parameter nomenclature list, one expression set from each of the two channels. These reconstructed expression sets may be then checked against the original to assure the engineer that the VPI system's interpretation of the expression set is correct. The primordial safety of the original expression set is checked in much the same way as relay logic circuits.

The integrity of the vital data base is protected against hardware failure once the system is in operation by the coded nature of the data themselves, in accordance with the principles of Numerically Integrated Safety Assurance Logic. Once the vital data base is originally verified as correct, any value change in either the data base or the vital routines themselves is immediately revealed during operation as an aberration in the checkword values, and power to the system's outputs is terminated as described above.

The "vital routine software" resident in the VPI system remains unchanged from application to application.

The VPI system has been designed to enable operation in a fully redundant mode. This optional mode allows two identical VPI systems to be connected in parallel. In the absence of any failure, both systems are fully operational However, only one system is driving the system outputs. In the event of a failure in one system, the outputs are automatically driven by the other system, which also reports the failure to the central office via the code system.

In the redundant mode, no mechanical or electrical switching of individual outputs is necessary under failure conditions. The vital outputs of both systems are simply tied together. Vital isolation of the output ports of each system prevent one system's failure from unsafely affecting the other.

User-oriented maintenance tools enable maintenance personnel to isolate failures in the VPI system quickly. Since only the main CPU board contains application-unique hardware (in the form of the memory data base), only a small number of universal spare board types is required to put a failed system back in operation.