S T S A R C E S

Standards for Safety Related Complex Electronic Systems

Annex 7

Methods for fault detection

Final Report of WP2.2

Jan Jacobson & Jacques Hérard

SP

European Project STSARCES

Contract SMT 4CT97-2191

Contents

Preface

The European Research project STSARCES Standards for Safety Related Complex Electronic Systems aims to harmonise validation methods for safety-related parts of machine control systems.

This is a draft issued to be subject of further discussions. Especially the requirements and the connection of requirements at specific categories (see chapter 4) have to be harmonised between the STSARCES partners.

Revision	Date	Changes made	Distribution
0	26 August 1998	First issue	BIA, INERIS
1	4 December 1998	Chapters on coverage and validation methods added	BIA
2	23 December 1998	Changes to chapter 1.1 and requirements. Chapter 6 integrated into chapter 2. Coverage chapter is now chapter 3. Some other minor changes.	BIA, INERIS
3	24 February 1999	Chapter 1.3 added. Several additions to chapter 3. Requirements of chapter 4 updated Check lists in chapter 5 modified. Some other minor changes.	All STSARCES partners
4	10 September 1999	Addition to chapter 1.1 Addition of figures to chapter 3	BIA
5	24 September 1999	Changes to chapter 1.1 : DC_N definition revised Changes to chapter 1.3 : new text added, rectification of the reference in last paragraph. Figures on applicable architecture configurations added. Changes to chapter 3 : Tables listing proprieties of methods for diagnostic coverage are slightly modified. Changes to chapter 4	All STSARCES partners
5	29 February 2000	Summary and conclusion added. Page numbers included in the table of contents	INERIS

Scope

Work Package 2.2 ”Methods for Fault detection” of the European research project STSARCES has the objective to establish which techniques and measures that can be regarded as basic or well-tried safety principles for detection of hardware faults in machine control systems. It is part of the STSARCES WP 2 ”Hardware Safety”, which also contains WP2.1 ”Quantitative analysis of complex electronic systems using fault tree analysis and Markov modelling”.

This document contains an introduction (chapter 1), a background description of the techniques (chapter 2), calculation and definition of diagnostic coverage (chapter 3), listing of the requirements (chapter 4) and validation methods (chapter 5). The introduction and the background description is regarded to be needed to harmonise the opinions of the STSARCES partners. The overall hardware architecture is not discussed in this document.

Summary

The evaluation of hardware safety is an important factor when assessing the overall characteristics of a safety-related system according to the machinery directive. The increasing integration of programmable electronics in industrial equipment adds new aspects that need to be considered. The experience and knowledge gathered from existing systems and the modern approach in system development could contribute to the validation process. The present work is an attempt to lay down some reference marks on existing methods and principles considered as state of the art. It also clarifies the interdependency between architecture design, diagnostic coverage and category (i.e. the system behaviour at fault).

The basic concepts of diagnostic coverage, test interval and architecture are explained. Measures for checking of processing units, memory, I/O units, interface, data paths, power supply and program sequence are described. These measures are then group into requirements for control systems to be used with safety functions of different category.

Eventually, preliminary tables are presented in order to summarise the aspects related to the validation of safety-related systems.

1 Introduction

Programmable electronic systems (PES) have the ability to detect faults within themselves before a fault is manifested as a failure of the system. The techniques and measures used focus on different parts of the electronic hardware and may require different amount of system effort. It is regarded as state-of-the-art to implement techniques for fault detection in PES used in safety-critical applications.

It is easy to imagine some possible faults which can cause unexpected behaviour of the machine which is controlled. A bit in a memory cell may be stuck at ”0” or ”1”. The output circuits may be stuck at ”ON”. A software fault may cause a task to enter an ”eternal loop”. Perhaps interruptions in the supply power, or variations in the voltage level, may influence the execution of the software. Data transferred on serial communication lines may be distorted by interference. An internal CPU fault might cause incorrect execution. There are techniques and measures to detect such faults before the machine gets out of control.

All safety critical systems should undertake basic measures to detect the faults, and possibly control the failures which might occur. The standard EN 954-1 specifies 5 categories for system behaviour at fault. Basic safety principles should be implemented for categories B, 1, 2, 3 and 4. For categories 1, 2, 3 and 4 also well-tried safety principles have to be implemented. The presence and performance of the safety principles must be validated.

It does not yet exist a common understanding if a certain measure fulfils the requirements for safety principles for a specific category.

Category	**System behaviour**	**Principles to achieve safety**
B	The occurrence of a fault can lead to the loss of the safety function.	Mainly characterised by selection of components
1	As in category B, but the probability of occurrence is lower than in category B.	"
2	Faults are detected by periodic checks at suitable intervals. The occurrence of a fault can lead to the loss of the safety function between the checks. The loss of the safety function is detected by the check.	Mainly characterised by structure
3	When a single fault occurs the safety function is always performed. Some but not all faults will be detected. Accumulation of undetected faults can lead to the loss of the safety function.	"
4	When the fault (faults) occur the safety function is always performed. The fault (faults) will be detected in time to prevent the loss of the safety function.	"

Figure 1a. Summary of categories for system behaviour at fault according to standard EN954-1.

1.1 Diagnostic coverage

Simple tests will not detect all hardware faults. Elaborate tests will detect many hardware faults at the cost of much processing effort spent. The diagnostic coverage, DC, is defined as the fractional decrease in the probability of dangerous hardware failure resulting from the operation of the automatic diagnostic tests. [IEC 61508-4, clause 3.8.6] See formula 1. If the test detects all faults, the coverage is 100%. If no faults are detectable, the coverage is 0%.

$$\text{diagnostic coverage DC} = \frac{\textit{the probability of detected dangerous failures}}{\textit{the probabilty of total dangerous failures}} \qquad [1]$$

Another definition of diagnostic coverage, $DC_{N,}$ (see formula 2) relates to the fraction of total number of different failures that is detected during a particular test. There can be large differences between the two ways to define the diagnostic coverage. The probability based approach distinguishes between faults which occur with different probability, while the number based approach does not. A test technique which detects faults occurring with high probability is very well likely to have a high DC, but may have a low DC_N if there is a large number of low probability faults which are not detected.

$$\text{diagnostic coverage } DC_N = \frac{\textit{the number of dangerous failures detected}}{\textit{the total number of dangerous failures}} \qquad [2]$$

It may be hard to find numerical values for the probabilities of different faults. Sometimes the assumption is made, that all faults have the same probability. This is always an approximation of reality.

It is possible to make a numerical calculation of the coverage of some methods. While the coverage of some other methods may have to be expressed in qualitative ways such as ”high/medium/low”. An estimation of the diagnostic coverage will be needed to be able to compare two diagnostic test methods.

A translation from the qualitative definition “low/medium/high”, to a quantitative measure expressed as a percentage will be needed. This report has chosen to follow the definitions suggested by the IEC 61508 standard. (See figure 1.) High coverage is used for techniques and measures with a probability higher than 99% to detect a fault. Medium coverage means a probability less than 99%, but higher than 90%. Low coverage will correspond to a diagnostic coverage greater than 60%, but lower than 90%. Techniques and measures offering less than 60% probability to detect faults are to be avoided in safety-related parts of control systems.

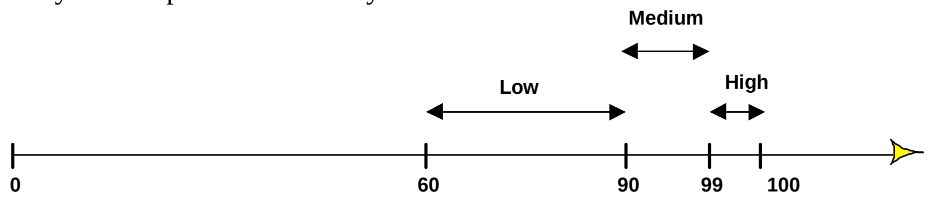


Figure 1. Diagnostic coverage defined as low, medium and high.

When numerical values are needed in calculations, 60% is used for “low” coverage, 90% is used for “medium” coverage and 99% is used for “high” coverage.

The quantification of the diagnostic coverage for different methods of fault detection in memory and I/O units (in chapter 3 of this report) is based on values extracted from the IEC 61508 standard. Those values are the results of theoretical studies based on a simplified probabilistic approach. The lack of data concerning the various types of memory chips, and the assumption that the potential faults are equally distributed introduce a number of uncertainties.

Similar uncertainties are introduced for other parts of the PES. The probability of different faults in the processing unit will depend on the type of processor, the manufacturer, the production process, the design etc. It is hardly possible to state a probability that will be valid in all cases. Assuming a normal fault distribution of possible faults, the previous expressions [1] and [2] for the diagnostic coverage become equivalent.

Faults in the programme sequence will have different probabilities depending on the programming language, the experience of the programmer, the testing effort etc. It will not be easy to state a probability-based diagnostic coverage for the programme sequence monitor.

The most valid estimation of diagnostic coverage for a fault detecting method should at this stage be limited to one of the three levels referred earlier in this section; low, medium or high. The level chosen may be different if probability, or numbers of errors, is used for the definition of diagnostic coverage. However, the level will be the same if all faults are equally probable.

1.2 Diagnostic test interval

Methods for fault detection can be used at power-up to establish if the electronic system is fit to start operating. Faults should be detected at start, and operation must not be allowed to start when faults are detected. Such a power-up test will not detect faults which occur during operation. Thus it will not be suitable for systems of high safety integrity with strict requirements for behaviour at fault. Power-up testing is sometimes used in systems with moderate safety requirements, and short operating time between power-ups.

Fault detection may also be performed at run-time to find faults occurring during operation. The diagnostic test interval will be of great importance to decide for which applications the system can be used. The test interval will be the maximum time during which an undetected fault may exist.

The diagnostic test interval is defined as the interval between on-line tests to detect faults in a safety-related system that have a specified diagnostic coverage [IEC61508-4, clause 3.8.7].

The application will decide the requirement for diagnostic test interval. It is not easy to define an exact requirement for a certain category. The failure rate being generally much greater than the diagnostic test interval, the diagnostic time interval should nevertheless not be longer than 1% of the mean time between operation demands.

1.3 Architecture

Different hardware architectures are possible for a PES intended to fulfil category 2, 3 or 4. Category 2 is often realised by a single channel structure, which is called a 1oo1 system (1 channel out of 1 provides the safety function). Category 3 may be realised by a single channel system in combination with auxiliary components such as an electromechanical module, or by a dual channel system. A PES intended to claim category 4 will probably be either dual-channel or triple-channel. Other more advanced architectures may certainly be used for any of the categories, but this is not state-of-the-art due to financial restrictions on the design.
The safety principles required will depend on the hardware architecture chosen. A dual-channel system employing safety principles of medium diagnostic coverage may provide the same probability for failure of the safety function, as a single channel system using safety principles with high diagnostic coverage. Faults in the single channel will have to be more accurately detected to provide a high probability of failure-free operation. A dual-channel system may tolerate an undetected fault in one of channels, while the other channel still provides the safety function.

In a similar way, a triple-channel system will need less efficient safety principles, than a dual-channel system. These differences will be reflected in the requirements on which safety principles to be employed for a specific category. (See the tables of chapter 4.)

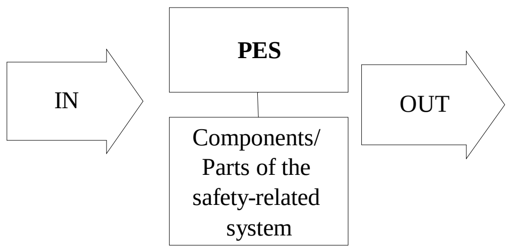


Figure 1.3.a Single channel

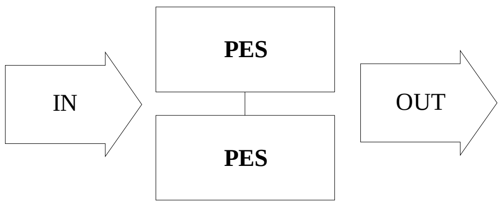


Figure 1.3.b Dual channel

Examples of block diagrams for cathegory 3 control systems

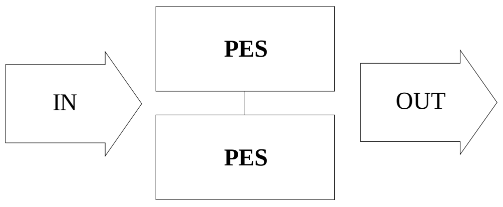


Figure 1.3.c Dual channel

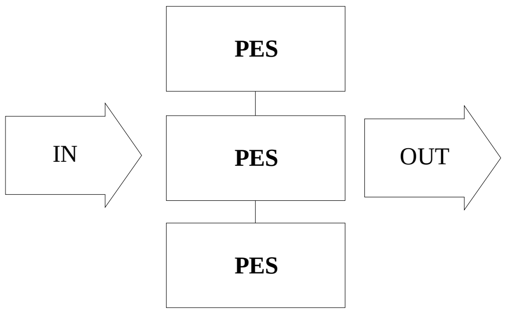


Figure 1.3.d Tripple channel

Examples of block diagrams for cathegory 4 control systems

2 Description of methods

There are several aspects in a programmable electronic system which can be automatically self-checked. The following chapter gives some background and explanations to the different techniques.

2.1 Processing units

The processing unit of the PES may suffer from faults which may cause malfunction. Examples of such faults could be bit errors in one of its internal registers, or malfunctioning of the instruction decoder.

Self-testing is employed by designing software routines which test the functionality of the processing unit. Certain operations are performed, and there will be only one correct result of such a test. The principle of letting a processor which might be defect check itself, relies on the assumption that a fault will corrupt the result of the self-check. The self-test has to be designed in such a way that the risk for a fault corrupting the test itself is negligible.

Some tests on the processing units are very hard to perform during run-time. A processing unit in full operation will make use of all registers, flags etc. It may be easier to perform exhaustive tests of the processing unit at power-up before the application software has been started.

Another approach is to let two processing units exchange data, and then compare the result. A fault in one of the processors is supposed to be detected by the other processor.

Following are examples of measures and techniques which are often employed to detect faults in processing units:
- self test of the execution of the instruction set.
- self test of registers by patterns or walking-bit [IEC61508-7, clause A.3.1, A.3.2]
- reciprocal comparison by software between two processing units [IEC61508-7, clause A.3.5]

2.2 Invariable memory ranges

Semiconductor memories may fail to work as intended. A fault in the invariable memory will corrupt the source code and constants stored there. The instructions to the processing unit may be distorted, and important constants and parameters will be incorrect. This will result in an unpredictable behaviour of the machine control system. It is important to try to detect such faults before the execution is disturbed.

The methods for checking of invariable memory all rely on reading the memory cells and compare the read values to what originally stored there. This may be done by direct comparison to a duplicate area, or by calculating a checksum (or signature). The checksum will then be compared to the sum which was originally calculated and stored in memory.

Memory tests can be quite time consuming. It may not be possible to cover the complete address range at one time. Run time checking is often performed by checking only a limited address range each time the memory test task is started. After a large number of calls to the memory test task the complete memory will have been tested.

Example: A 8 kbyte ROM memory is check summed by adding 8 bytes to the check sum every time the memory test task is called. It will take a total of 1024 calls (8*1024/8=1024) before the check sum is completed. If the memory test task is started every 50 ms, the diagnostic test interval will be 51 seconds (1024*50ms= 51.2 seconds).

It will not always be possible to use time-consuming memory tests at reset or power-up. The operator might find the time to wait for system start too long. In such cases, a simple power-up test may be combined with a more exhaustive run-time test.

Following are examples of measures and techniques which are often employed to detect faults in invariable memory ranges:
- checksum [IEC61508-7, clause A.4.2]
- 8-bit signature [IEC61508-7, clause A.4.3]
- 16-bit signature [IEC61508-7, clause A.4.4]
- replication [IEC61508-7, clause A.4.5]

2.3 Variable memory ranges

Semiconductor memories may fail to work as intended. A fault in the variable memory will corrupt all variables stored in memory. This will result in an unpredictable behaviour of the machine control system.

The methods for checking variable memory rely on different ways to stimulate the memory cells, and check that the function correctly. Test patterns of different complexity can be written and then read back. A complex test pattern will have a higher diagnostic coverage than a simple test pattern.

Example: A simple test of variable memory can be made by writing and reading bit patterns according to the following:
- save contents of memory cell to test.
- write bit pattern 055H.
- read and compare memory cell with 055H. Handle error if no correspondence.
- write bit pattern 0AAH.
- read and compare memory cell with 0AAH. Handle error if no correspondence.
- restore contents of memory cell.

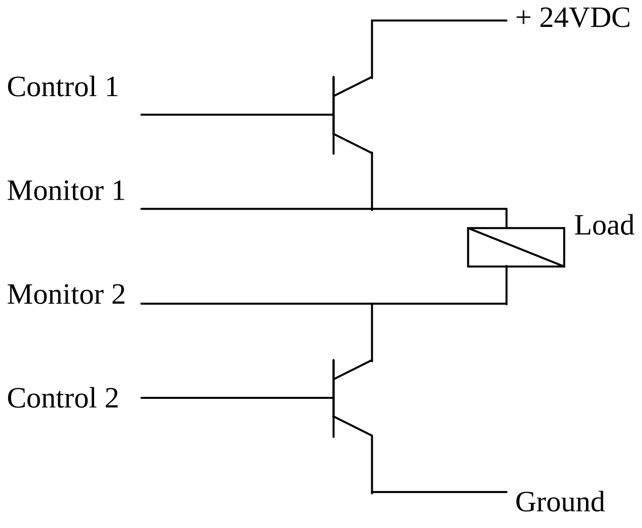
Also testing of variable memory can be time consuming, as described above for invariable memory.

Following are examples of measures and techniques which are often employed to detect faults in variable memory ranges:
- RAM test ”checkerboard” or ”march” [IEC61508-7, clause A.5.1]
- RAM test ”walkpath” [IEC61508-7, clause A.5.2]
- RAM test ”galpat” [IEC61508-7, clause A.5.3]

2.4 I/O Units and Interface

Interfaces to external units are present on many safety-related machine control systems. They can be analogue, digital, serial or parallel. The external communication through these interfaces can be of great importance for the safety. The status of the I/O units and interfaces may be critical, and should therefore be monitored.

Example: The semiconductor outputs of a safety-related PES are doubled and monitored. A fault in one output transistor will be detected, while the other transistor still is able to switch off the output load.



Following are examples of measures and techniques which are often employed to detect faults in I/O and interfaces:
- multi-channel parallel output [IEC61508-7, clause A.6.3]
- monitored outputs [IEC61508-7, clause A.6.4]
- input comparison/voting [IEC61508-7, clause A.6.5]

2.5 Data paths

Even a physically small PES may consist of several internal units which communicate. Examples of such data paths are electrical parallel buses, serial buses and optical fibres. The data paths between the internal units may fail, and should be checked.

Important signals, such as alarm signals, originating from one unit should be detected and processed by the appropriate receiving unit.

Example: An optical fibre is used as data path between two modules of a safety-related PES. All messages have an 16-bit checksum included, and all safety-related commands are transmitted twice before they are accepted and executed by the receiving module.

Following are examples of measures and techniques which are often employed to detect faults in data paths:
- inspection using test patterns [IEC61508-7, clause A.7.4]
- transmission redundancy [IEC61508-7, clause A.7.5]
- information redundancy [IEC61508-7, clause A.7.6]

2.6 Power supply

All safety-related programmable electronic systems should have some kind of circuitry to ensure that operation will not be started before an adequate supply voltage has been reached. The behaviour of the processor and other electronic circuits is specified only for a specific voltage range.

Monitoring of the supply voltage level will be important also during run-time. Interruptions of the supply power will cause the electronic hardware to enter an undefined range where the exact behaviour cannot be foreseen. The monitoring circuit must give an alarm in time before the voltage reaches a threshold value. The PES should have a ”graceful death” bringing the controlled machinery to a safe state.

The programmable electronic system should then have time to take proper action to enter a safe state. It may also be necessary to save machine status and calculated values in a non-volatile memory.

The output signal from the hardware circuit monitoring the supply power can be either static or dynamic. The output from a static monitor will indicate proper power supply with a constant high or low output signal. A change of the output will give an alarm to the processor. A monitor built on a dynamic principle will have a dynamic output signal. Power failure will be indicated by a constant signal.

The most common way to implement a supply power monitoring is to use a standard commercial supply power monitor circuit. Such ICs are available from several semiconductor manufactures. A disadvantage with this type of circuit is that it is normally not testable, i.e. a fault in the circuit will not be noticed before the PES is shut down in an uncontrolled way.

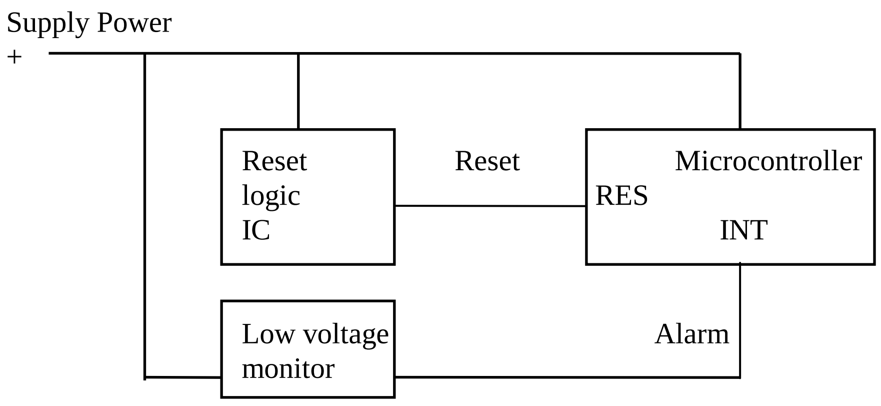


Figure 2.6.a Example of block diagram of external circuits for power-up reset and detection of low supply voltage

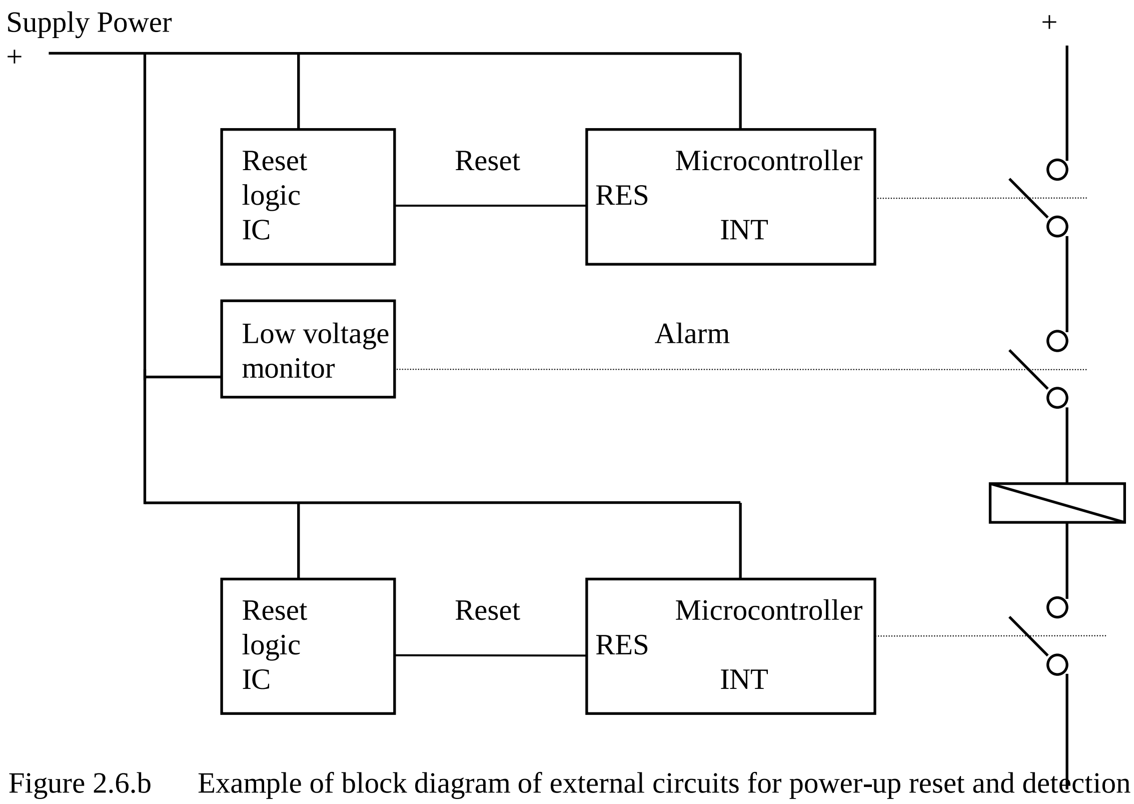


Figure 2.6.b Example of block diagram of external circuits for power-up reset and detection of low supply voltage in a dual-channel system.

Also over-voltage may create unwanted behaviour of the control system. Checking facilities can also be implemented to react on over-voltage before the specified operating voltage is exceeded, and the behaviour cannot be guaranteed. One example when over-voltage detection is needed may be a dual-channel system using a single power supply.

Following are examples of measures and techniques which are often employed to detect faults in the power supply:
- over-voltage protection with safety shut-off [IEC61508-7, clause A.8.1].
- monitoring of secondary voltages [IEC61508-7, clause A.8.2].
- power-down with safety shut-off [IEC61508-7, clause A.8.3].

2.7 Program sequence

The execution sequence of the software may be distorted by either software faults, hardware faults or environmental disturbances. This will with great probability lead to incorrect behaviour of the programmable electronic system. The consequences of such a fault are not possible to foresee since ”anything may happen”. However, there are well established techniques to monitor the programme sequence.

The monitoring of programme sequence may be realised both in hardware and in software. A combination of a hardware unit (”watchdog”) with a logical monitoring realised in software will be the most powerful. But also less sophisticated techniques built on simple hardware or software solutions will certainly be able to detect some errors in the programme sequence.

2.7.1 Monitoring by hardware

A watchdog is defined as a hardware design which monitors the operation of internal hardware functions, and/or application programme functions, and/or system software functions and will result in a safe condition if not periodically reset at a predetermined interval. [prEN 12978] In practice, this means a timer circuit which is triggered by the processor at a periodic interval. If the trigger does not reach the watchdog, a stop or reset signal is given to the processor.

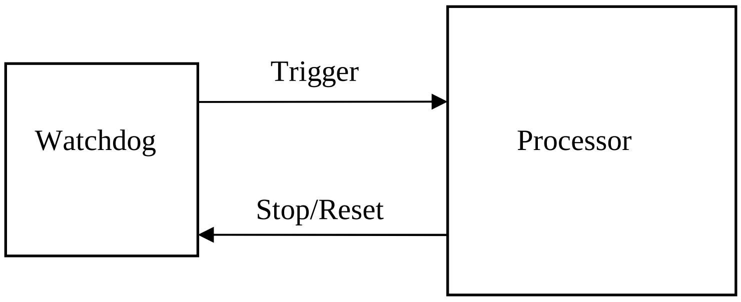


Figure 2.7.a. Watchdog, block diagram

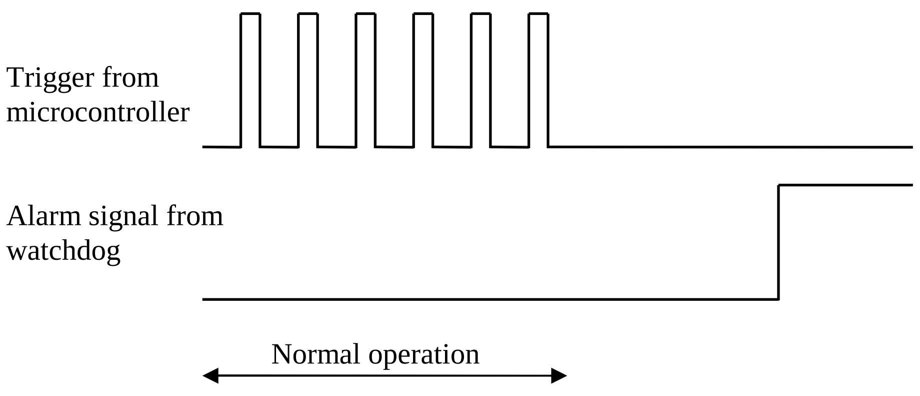


Figure 2.7.b. Watchdog triggering and alarm signals

There are several hardware solutions for watchdog functionality. Many microcontrollers offer a watchdog circuit on-chip. The watchdog is then programmed, activated and controlled through internal registers of the controller. There are also special circuits available to supervise microcontrollers. Such circuits often offer the facility of a watchdog. Another solution is to design a separate hardware circuitry based on a monostable flip-flop which has to be retriggered at a specified interval.

The watchdog circuit is preferred to be hardware independent from the processor itself. The same error which causes the fault in programme sequence should not cause also the watchdog to stop functioning. There is an increased risk for this if the watchdog is integrated on the same chip as the processor. Special caution must be paid if the time base used by the processor and the watch dog is the same. A clock fault might then affect both the microcontroller and the watchdog.

A good watchdog should be tested or fail-safe. Systems with high requirements for functional safety may let the processor fake a watchdog alarm at every power-up, or at a periodic interval. The processor will then detect if the watchdog is non-operational. Otherwise there is a risk for a fault in the watchdog to pass unnoticed, until it really will be needed. Another possibility is to design a watchdog circuitry where single hardware faults will cause an alarm.

2.7.2 Software aspects

Software must be used to generate the trigger to the watchdog hardware circuit. It may also be used to check the programme sequence even if no hardware has been implemented as watchdog.

The simplest way of triggering in a cyclical system is to check the program sequence once every cycle. It is not advisable to make the checking or watchdog triggering in a dedicated interrupt routine or isolated task. Such a triggering will only prove that one single routine or task is running. The main program may be ”locked up”, without this influencing the execution of a time triggered interrupt routine.

A more powerful way to check the program sequence by software, is to use trigger flags and key words to indicate that all the significant parts of the software are active and executed in the correct sequence.

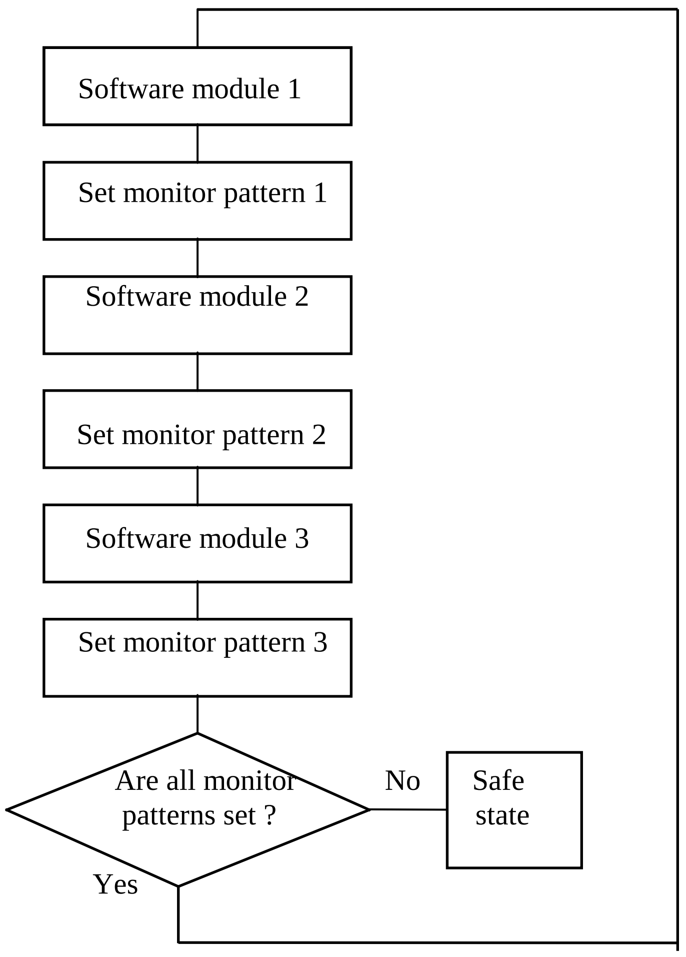


Figure 2.7.c. Example of program sequence monitoring by software

2.7.3 Actions at fault

A detected fault in the program sequence will require different kinds of actions depending of the type of system. It is important that the correct action is taken. It will not be enough to design a watchdog into the system, and not specify properly which action shall be taken at fault.

Most machines have a safe state which shall be entered when a fault is detected. A watchdog circuit may force the processor and the outputs to the safe state where signals are inactive. Another possibility is that a watchdog alarm will cut the power to the outputs and leave the machine in safe state. (See figure 2.7.d.) This safe state must not be possible to exit without a dedicated action of reset.

For applications with requirements for high availability, the watchdog alarm may be used to isolate one of the processors and let a redundant unit take over without affecting the normal operation of the system.

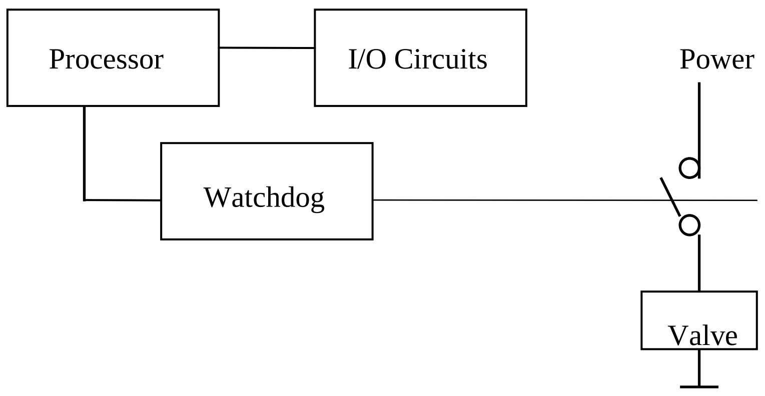


Figure 2.7.d. The watchdog may be used to disconnect the outputs.

A single-channel system which is intended to run in continuous mode should be reset by a watchdog alarm, and the operation should continue as soon as possible. No safe state exists in this case, and there will be risks associated with an inactive processor.

If the control system contains any values measured or calculated during run-time, it must be specified how these values shall be handled after a watchdog alarm. Can they still be trusted and used, or must all such values be reset to default?

2.7.4 Examples

Following are examples of measures and techniques which are often employed to detect faults in program sequence:
- an on-chip watchdog with separate time base without time-window, e.g. Motorola microcontroller 68HC11 [IEC61508-7, clause A.9.1]
- a watchdog with separate time base without time-window, e.g. a Maxim microprocessor supervisor IC [IEC61508-7, clause A.9.1]
- logical monitoring of programme sequence implemented in software[IEC61508-7, clause A.9.3]
- combination of temporal and logical monitoring of programme sequence [IEC61508-7, clause A.9.4]

3 Diagnostic coverage

3.1 Processing unit

The central processing unit (CPU) is responsible for executing all software instructions in their programmed sequence.
A correct function of the CPU relies upon the use of adequate features for control and avoidance of faults or dysfunction in any of its components. The arithmetic/logic unit, the control unit, the main memory, stacks and registers are integral parts of the CPU.

The arithmetic/logic unit

The arithmetic/logic unit (ALU) operates as the processor’s calculator, executing operations and returning the results to the main memory.
The operation of the arithmetic/logic unit and control unit depends on the contents of associated registers, accumulators and stacks. These accumulators, stacks, and registers hold the operands necessary to the execution of the arithmetic instructions as well as the results of arithmetic calculations or data manipulations.

The control unit

The task of the control unit is to interpret machine instructions and to issue commands to all other units such as storage devices, input and output devices, and main memory.
The current value of the control register indicates at which address the next instruction shall be fetched. The function of the instruction control unit is to generate the microprogram which results from the decoding operations performed by the instruction decoder.
The types of faults which may affect the instruction control unit may depend on register and main memory faults or on faults in the instruction decoder. The register faults are similar to memory faults. Faults in the instruction decoder may be a combination of faults in main memory and registers and faults in the decoder logic.

The main memory registers and stacks

The main memory is used for short term storage, holding data which is currently needed to carry out processing instructions.
Normally, program instructions and data are stored in different places in the main memory. The area of main memory where program instructions are held while awaiting execution is the instruction stack.

The registers
. Accumulator where results of computations are hold
. Storage register where data about to be used is stored
. Addressed register
. General purpose register
are used to hold current data and sometimes sections of the program to be processed.

The types of faults or dysfunction which may affect the main memory, stack and registers are similar to those listed in the following section. (3.2 Memory ranges). Consequently certain methods used to detect or avoid faults in the CPU may be common those used to detect faults in memory units.

There are several methods to achieve fault diagnostic coverage of the processing unit of which the following can be used to reach a specified rate of coverage.

This work shall focus upon the methods listed in the IEC 61508 standard (part 2, A.4).

Comparator
Majority voter
Self-test by software: limited number of patterns (one channel)
Self-test by software : walking bit (one channel)
Self-test supported by hardware (one channel)
Reciprocal comparison by software

3.1.1 Comparator (IEC 61508-7, A.1.3)

This method is applicable for the detection of errors in instruction decoding and execution.
In dual or multi-channel systems a special device provides for the detection of fault/error by means of comparison of data from the different channels.
The periodicity of the test execution shall comply with the specific application in order to reach an acceptable level of diagnostic coverage.
If a fail-safe comparator device is used, a continuous execution of the test procedure or an adequate periodic repetition frequency of the test should provide for a high diagnostic coverage. In the following example, the comparator is a hardware device.

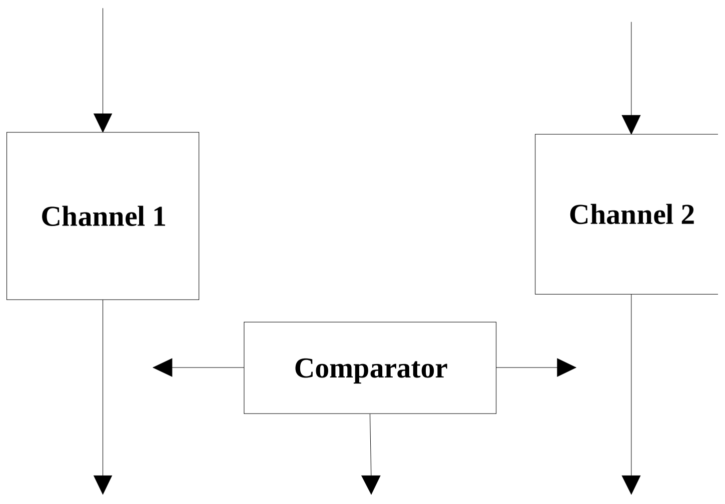


Figure 3.1a **Comparator in a dual-channel system**

In the following example the comparator is implemented by software. The multiple processing is realised by two independent sets of data and application programs. Since there is only one processing unit, the double processing is performed in sequence as shown in the figure. The independent output results are checked by the software comparator.
A high fault diagnostic coverage is achieved if the implementation of the two application programs is diverse.

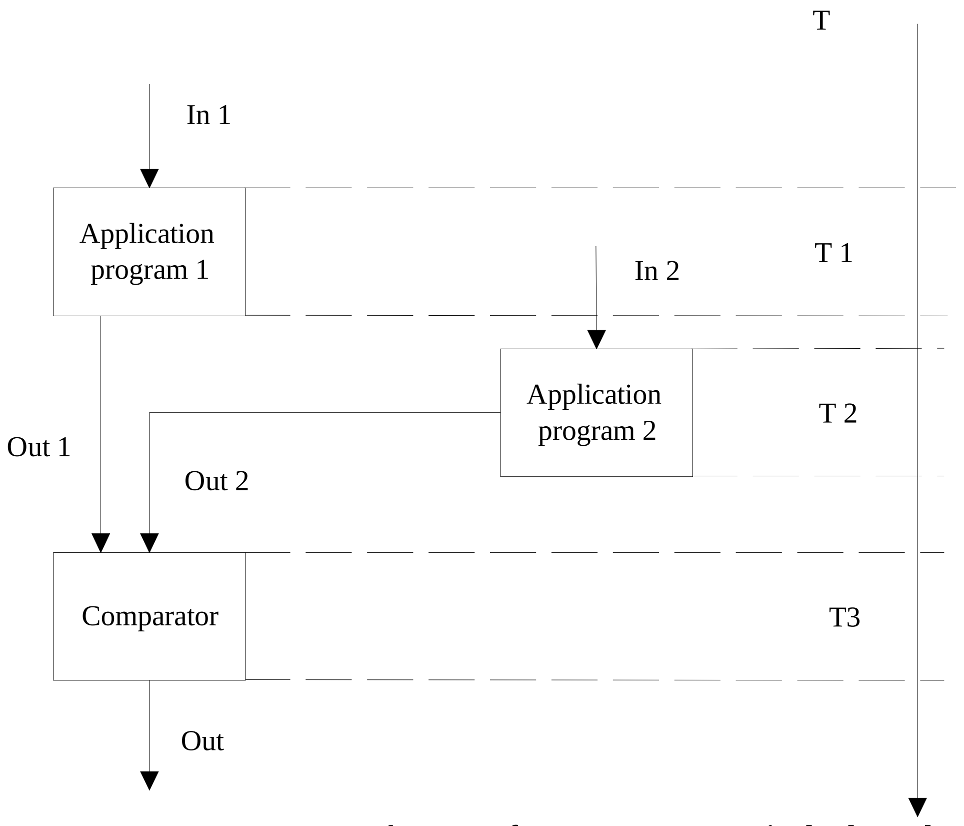


Figure 3.1 b **Software comparator single channel**

3.1.2 Majority voter (IEC 61508-7, A.1.4)

This method is applicable for the detection of errors in instruction decoding and execution in multi-channel systems. The method shall detect and mask failures in one of at least three hardware channels.

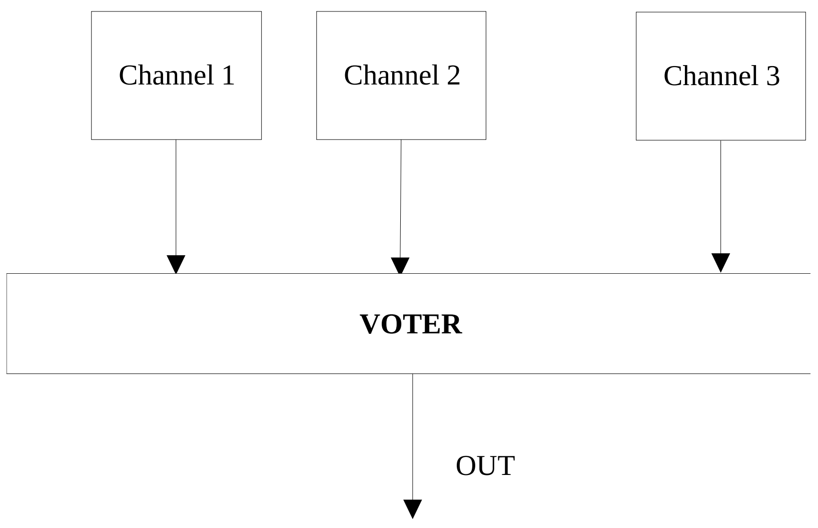


Figure 3.1 c Voter in a three channel system

3.1.3 Self-test by software: limited number of patterns [one channel] (IEC 61508-7, A.3.1)

Standard techniques are used for the hardware construction. The failure detection is realised by means of specific software functions which provide for self-tests using at least two data patterns.
For example a periodic testing can be carried out by introducing the test patterns 77hex and 88hex.
A low fault diagnostic coverage is achieved by this method.

3.1.4 Self-test by software : walking bit [one channel] (IEC 61508-7, A.3.2)

This test method is focused on faults which can affect the memory parts of the processing unit. Assuming that the hardware consists of standard memory type without any parity bit, software test functions are implemented. Self-tests are performed by using a data pattern to check the physical storage medium (stacks, registers, accumulators).
Given a specific memory unit to check, the different cells are sequentially addressed with a one bit value. The content of all the cells in the register are then read. The same procedure is repeated for the next cell until all cells are addressed.
A medium fault diagnostic coverage is achieved by this method.
Example : test of a four-bit register

Content of register when starting the test walking-bit sequence

0	0	0	0

Testing sequence starts

0	0	0	1

After checking the content of the register, the content of the adressed cell is reset to its previous value.

0	0	0	0

This procedure is then repeated untill all the cells in the register are set.

0	0	1	0
0	1	0	0
1	0	0	0

Figure 3.1 d Walking-bit (one-channel)

3.1.5 Self-test supported by hardware [one channel] (IEC 61508-7, A.3.3)

Hardware supported test method for failure detection. A periodic hardware facility may be used for monitoring a certain bit pattern according to the watchdog principle.

A medium fault diagnostic coverage is achieved by this method.

3.1.6 Reciprocal comparison by software (IEC 61508-7, A.3.5)

The method is built on the comparison result of data exchanged between the processing units of a multi-channel architecture. The types of data include results, intermediate results and test data.
The detection of discrepancies results into the generation of an error/failure message.

A high fault diagnostic coverage is achieved by this method.

3.1.7 Summary of examples

The table in figure 3.1 lists methods for checking the processing units. The list is not exhaustive, and other acceptable methods exist,

Test method for diagnostic coverage of processing unit	Maximum achievable fault diagnosis coverage	Reference in IEC 61508- 7
Comparator	High	A.1.3
Majority voter	High	A.1.4
Self-test by software : limited number of patterns (one channel)	Low	A.3.1
Self-test supported by software : walking bit	Medium	A.3.2
Self-test supported by hardware (one channel)	Medium	A.3.3
Reciprocal comparison by software	High	A.3.5

Figure 3.1 Examples of test methods of processing unit

3.2 Memory ranges

One aspect of memory design involves ensuring the integrity of data stored in memory. Memory devices can sometimes develop permanent faults which cause the memory to function incorrectly. Faults in a memory device can affect three different functions:
- The memory cell array,
- The decoder logic
- The Read/Write logic.

Memory Cell Array

A memory unit is built of memory cells. Physical cellular alterations such as metallization short-circuits and capacitive coupling may lead to the following types of faults:

1. One or more cells are stuck at 0 or 1
2. The occurence of a coupling between one or several pairs of cells. In such a case, the transition from x to y in one cell of the pair induces a state transition in the other cell.

Decoder

The function of the decoder is to select a unique memory cell for a specific address. The occurence of any failure in the decoder logic gives rise to one of the following behaviours :

1. The decoder does not access the addressed cell and may even access non addressed cells.
2. The decoder accesses multiple cells, including the addressed cell.

Depending on the logic used, the case of no access to an addressed cell is equivalent to that cell stuck at 0 or stuck at 1.
The case of multiple accesses is equivalent to coupling between cells in the memory cell array.

Consequently, the faults encountered in the decoder do not need to be treated specifically since they are in every aspect similar to the set of faults observable in a memory cell array.

Read/Write Logic

Independently of the type of memory (variable, invariable), the device can be considered to have *K* inputs and *N* outputs. The occurence of any failure in the Read/Write logic gives rise to one of the following behaviours :

Data input lines or data output lines may interact with each other via short-circuits or capacitive coupling. These errors are equivalent to coupling between memory cells.

Output lines of the sense amplifier logic or write driver logic may be stuck at 0 or stuck at 1. In either case this fault is equivalent to stuck-at-0 or stuck-at-1 memory cells that correspond to the stuck output lines.

The purpose of the following sections is to point out adequate methods and techniques to achieve specified levels of diagnostic coverage. These techniques, adapted to the current type of memory device, shall concentrate on faults in the memory cell only.

3.2.1 Invariable memory ranges

There are several methods to achieve fault diagnostic coverage of memory units of which the following can be used to reach a specified rate of coverage.

This work shall focus upon the methods listed in the IEC 61508 standard (part 2, A.5 - A.6).

Word saving multi-bit redundancy
Modified checksum
Signature of one word (8 bit)
Signature of a double word (16 bit)
Block replication

3.2.1.1 Word saving multi-bit redundancy (IEC 61508-7, A.4.1)

The multi-bit redundancy technique is used to detect single bit errors, 2-bit errors, 3-bit errors, and all-bit errors in a 16-bit word. A common approach to error bit detection, is the simple parity technique where one parity bit is added to each character.

The word saving multi-bit redundancy is a variation of simple parity and is achieved by the addition of several redundant bits, a so called check code, to each word. By doing so the probability of multi-bit error detection is increased.

Usually data rarely consist of long continuous streams of bits. Data access or recovering is broken into blocks of data.
A block code converts a fixed length of ***K*** data bits to a fixed length ***N*** code word, where ***N*** > ***K***. The rate of the code is the ratio ***K*** / ***N***, and the redundancy of the code is ***1*** - (***K***/***N***).

The procedure can also be used to detect addressing errors, by calculating the redundant bits for the concatenation of the data word and its address.

Diagnostic coverage : medium

$$DC_N = \frac{\left[\sum \textit{all single bit errors} + \sum \textit{all } 2-\textit{bit errors} + \sum 3-\textit{bit} + \sum \textit{all}-\textit{bit errors}\right]}{\left[\sum \textit{all possible errors}\right]}$$

3.2.1.2 Modified checksum (IEC 61508-7, A.4.2)

Common to all checksum techniques is that one set of code is generated by a specific algorithm. This set of code, representing the contents of a memory area is then stored as the defined checksum value. At runtime, during self test, a new set of code is generated by the same algorithm and compared with the stored value. If a difference occurs an adequate measure is taken and/or a failure message is raised to bring the condition to the attention of the user/operator.

What characterises a checksum technique is the nature of the set of code generated and the type of algorithm used. The combination of those two factors permits to chose the appropriate method for a specified rate of diagnostic coverage.

In the case of the modified checksum a single word is generated and saved. This word represents the contents of all words in memory.

A modified checksum is a fault/error control technique in which a single word representing the contents of all words in memory is generated and saved. The algorithm applied to derive the checksum value shall use all the words in the current block of memory.
During self test, a checksum is produced from the current algorithm and compared with the previously saved checksum value. The detection of a difference shall result in a predefined invariable memory-error measure and/or message.

Diagnostic coverage :

$$DC_N = \frac{\left[\sum \textit{all odd}-\textit{bit errors} + \sum \textit{even}-\textit{bit errorss}\right]}{\left[\sum \textit{all possible errors}\right]} \approx 50\ \%$$

3.2.1.3 Signature of one word (8 bit) (IEC 61508-7, A.4.3)

This technique is used to detect 1-bit errors and multi-bit errors within a word.

The Cyclic redundancy check (CRC), is used in applications involving detection of small changes in blocks of data. Such applications include start-up verification of ROM code, program and data correctness validation.

The CRC algorithm operates on a block of data as a unit. The CRC algorithm divides this single large value by the CRC polynomial or generator polynomial. CRC polynomials are designed and constructed for test of data blocks of limited size. Larger amount of data invalidate some of the expected properties such as the guarantee of detecting any 2-bit errors.

The remainder of the division between the value of the data block and the CRC polynomial is the CRC result or signature. In the case of Signature of one word technique, the CRC result shall be at least one word in size.

The signature is sent or stored along with the original data. When the data is received or recovered from storage, the CRC algorithm is reapplied and the latest signature compared with the original signature. A failure message is raised if there is a difference.

Diagnostic coverage : DC_N =

$$\frac{\left[\sum all\, 1-bit\ errors + \sum all\ multi-bit\ errors\ within\ a\ word + \sum all\ possible\ bit\ errors\right]}{\left[\sum all\ possible\ errors\right]}$$

= 99.6 %

3.2.1.4 Signature of a double word (16 bit) (IEC 61508-7, A.4.4)

This technique is used to detect 1-bit errors and multi-bit errors within a word, as well as all possible bit errors.

The signature of a double word technique is a variation of the signature of one word technique where the CRC result value is at least two words in size.
For 16-bit polynomials, the maximum designed data length is generally 2^{15} - 1 bits, which is just less than 4K bytes. Consequently, a 16-bit polynomial is not the best choice to produce a single result representing an entire file, or even to verify a single EROM device, of size 8K or more.

Diagnostic coverage :

DC_N =

$$\frac{\left[\sum all\, 1-bit\ errors + \sum all\ multi-bit\ errors\ within\ a\ word + \sum all\ possible\ bit\ errors\right]}{\left[\sum all\ possible\ errors\right]} \approx$$

99.998%

3.2.1.5 Block replication (IEC 61508-7, A.4.5)

This technique is used to detect all bit errors.

The block replication is a technique used for storage and loading of safety-related data and programs. The block of data is duplicated in different format and stored in separate memory areas. The contents of the two memory areas are compared and a failure message is raised if a difference is detected.
The detection of certain types of bit-errors requires that the data is stored inversely in one of the two memories and re-inverted when read.

Diagnostic coverage : high

$$DC_N = \frac{\left[\sum \textit{all bit errors} + \sum \textit{all word errors}\right]}{\left[\sum \textit{all possible errors}\right]}$$

3.2.1.6 Summary of examples

The table in figure 3.1 lists methods for checking the invariable memory ranges. The list is not exhaustive, and other acceptable methods exist.

Method for fault diagnostic coverage of invariable memory ranges	Maximum achievable fault diagnosis coverage	Reference in IEC 61508- 7
Word saving multi-bit redundancy	High	A.4.1
Modified checksum	Low	A.4.2 This technique detects all the odd errors and some of the even errors
Signature of one word (8 bit)	High	A.4.3 This technique detects all one bit and a high percentage of multibit errors
Signature of a double word (16 bits)	High	A.4.4
Block repetition	High	A.4.5

Figure 3.2.1 Examples of test methods for invariable memory

3.2.2 Variable memory ranges

3.2.2.1 RAM test ”checkerboard” or ”march” (IEC 61508-7, A.5.1)

The RAM checkerboard test is used to detect static bit errors.

The checkerboard test is based on a checker-board type pattern of 0’s and 1’s which is written into a bit-oriented memory area. The cells of the memory area under test are inspected in pairs to ensure that the contents are the same and correct. The address of the first cell in such a pair is variable and the address of the second cell is derived by bit inversion of the first address.
In the first phase, the variable address is incremented until the end of the address space of the memory area under test.
In the second phase, the address is decremented to its original value.

The test is repeated with the checker-board pattern inverted
A failure message is raised if any difference is occurs.

Diagnostic coverage : low

$$DC_N = \frac{\sum static\ \ failures}{\sum all\ \ possible\ \ failures}$$

3.2.2.2 RAM test ”walk-path” (IEC 61508-7, A.5.2)

The RAM walk-path test is used to detect static and dynamic bit errors, as well cross-talk between memory cells.

The first step in the walk-pat test consists of initialising the chosen memory area to be tested. This is realised by writing a standard data pattern in that memory area. A bit inversion is performed on the first cell and the remaining memory area is inspected.

The second step consists of re-inverting the first cell followed by another inspection of the remaining memory area.

These two steps are repeated for all memory cells in the memory area under test.

A symmetric test is carried out by performing a bit inversion of all cells in memory under test and proceeding as described above.

Diagnostic coverage : medium

$$DC_N = \frac{\left[\sum static\ \ failures + \sum dynamic\ \ failures + \sum cross-talk\ \ failures\right]}{\sum all\ \ possible\ \ failures}$$

3.2.2.3 RAM test ”galpat” or ”transparent galpat” (IEC 61508-7, A.5.3)

The Galpat and the transparent Galpat techniques are used to detect static bit errors and dynamic couplings between cells.

Description and performance of the method

The first step in the Galpat test consists of initialising a chosen memory area. This is realised by setting all cells to the value 0 or 1.
The second step consists of inverting one cell at the time;
For each inverted cell, the value of the remaining cells are inspected sequentially by single read access. The inverted cell is checked after each read access.

The same procedure is repeated with the opposite initialisation of the memory range to be tested.
In the transparent Galpat method no initialisation takes place. When a cell to be tested is chosen, the inspection of the remaining cells is realised by sequential read access followed by the generation of a signature e.g. S1, which is stored.

The next step is to invert the cell to be tested and to repeat the same inspection process of the remaining cells followed by the generation of a second signature e.g. S2, which is also stored. The signatures S2 and S1 are then compared. Any difference gives rise to an error message.
The last step in the method is the re-inversion of the cell to be tested followed by the generation of the signature e.g. S3 of all the remaining cells. A comparison of S3 and S1 is performed and any discrepancy gives rise to an error message.

The inspection of all the cells in the memory range are done in the same manner.

Diagnostic coverage : high

$$DC_N = \frac{\left[\sum static\ \ failures + \sum dynamic\ couplings\right]}{\sum all\ \ possible\ \ failures}$$

3.2.2.4 RAM test ”Abraham” (IEC 61508-7, A.5.4)

This technique is used to detect all stuck-at and coupling errors between memory cells.

Description and performance of the method

The Abraham test is a form of variable memory pattern test which identifies all stuck-at faults and all coupling faults between memory cells. The proportion of faults detected exceeds that of the RAM test ”galpat”. A number of 30n operations is necessary to perform the test of n cells in memory.

Diagnostic coverage : high

$$DC_N = \frac{\left[\sum all\ stuck-at\ \ failures + \sum\ \ coupling\ \ failures\right]}{\sum all\ \ possible\ \ failures}$$

3.2.2.5 Parity-bit for RAM (IEC 61508-7, A.5.5 One-bit redundancy - for example RAM monitoring with a parity bit)

This technique is used to detect all possible bit errors in the memory range tested.

A very common approach to error detection is the single parity check code. This code appends to each ***K*** data bits an additional bit whose value is taken to make the ***K+1*** word even (or odd).
Such a choice is said to have even (odd) parity. With even (odd) parity, a single bit error will make the received word odd (even).
The implementation of this technique for an N*8-bit memory simply requires an extra block of memory to realise an N*9-bit memory. If any single bit of the 8 stored bits is corrupted, the parity bit will be incorrect. This is also the case for 3, 5, or 7 bits errors. However 2 or any even number of errors will not be noticed.

Diagnostic coverage : low

$$DC_N = \frac{\sum all\ \ possible\ bit\ errors}{\sum all\ \ possible\ errors}$$

3.2.2.6 RAM monitoring with a modified hamming code (IEC 61508-7, A.5.6)

This technique is used to detect all odd-bit errors, all 2-bit errors, 3-bit and multi-bit errors.

Error-Correcting Codes - Hamming Code

Data is rarely transmitted in long continuous streams of bits. Transmission of data is usually broken into blocks or message. Each block or message is a separate unit of transmission.
A message of length ***n*** (***n*** = ***m*** + ***r***) consists of :

m message bits (data bits)
r redundant bits (check bits)

The check bits are part of the error protection and recovery mechanism built in the message.
The message of ***n*** bits size, is called an n-bit codeword.
The number of bits which differ in two codewords is called the Hamming Distance.

The significance of the Hamming distance is that if two codewords are a Hamming distance d apart, it will require d single bit errors to convert one into the other. In other words, errors that involve less bits than the Hamming distance can be detected.

In most data transmission applications, all 2^m possible data messages are legal. But, depending on how check bits are computed, not all codewords are used.

Knowing how codewords are generated, gives information on how to construct a list of legal codewords and from that knowledge find the minimum Hamming distance.
This distance is the Hamming distance of the complete code.

The error-detecting and correcting properties of a code depend on its Hamming distance.
To *detect* d errors requires a distance d +1 code.
With such a code it is not possible that d single-bit errors can change a valid codeword into another valid codeword.
To *correct* d errors requires a distance 2d+1 code, so that legal codewords are so far apart that even with d changes, the original codeword is still closer than any other codeword and consequently can be uniquely determined.

Diagnostic coverage : high

$$DC_N = \frac{\left[\sum all\ odd - bit\ errors + \sum all\ 2 - bit\ errors + \sum 3 - bit\ errors + \sum multi - bit\ errors\right]}{\sum all\ \ possible\ errors}$$

3.2.2.7 Double RAM with hardware or software comparison and read/write test (IEC 61508-7, A.5.7)

This technique is used to detect all bit errors.

The double RAM with hardware or software comparison and read/write test technique is based on duplication of the safety-related contents of a chosen memory area.
The contents of memory is duplicated in different format and stored in separate memory areas.
The contents of the two memory areas are compared and a failure message is raised if a difference is detected.

Diagnostic coverage : high

$$DC_N = \frac{\sum all\ bit\ errors}{\sum all\ \ possible\ errors}$$

3.2.2.8 Summary of examples

The table in figure 3.2 lists methods for checking the variable memory ranges. The list is not exhaustive, and other acceptable methods exist.

Method for fault diagnostic coverage of variable memory ranges	Maximum achievable fault diagnosis coverage	Reference in IEC 61508- 7
RAM test ”checkerboard”	Low	A.5.1
RAM test ”walk-path”	Medium	A.5.2
RAM test ”galpat or ”transparent”	High	A.5.3 Transparency during the operating cycle can be achieved by partitioning the memory. The testing of each partition is later realised in different time segments.
RAM test ”Abraham”	High	A.5.4
Parity-bit for RAM	Low	A.5.5
RAM monitoring with a modified hamming code	High	A.5.6
Double RAM with hardware or software comparison and read/write test	High	A.5.5

Figure 3.2.2 Examples of test methods for variable memory

3.3 I/O Units and Interfaces

There are several methods to achieve fault diagnostic coverage of I/O units and interface of which the following can be used to reach a specified rate of coverage.
This work shall focus upon the methods listed in the IEC 61508 standard (part 2, A.7).

- Test pattern
- Code protection
- Multi-channelled parallel output
- Monitored outputs
- Input comparison/voting

3.3.1 Test pattern (IEC 61508-7, A.6.1)

The aim of this test method is to detect the faulty states which may affect any input or output lines. A defined flow of data is fed via the input ports of the control system which after processing gives rise to the corresponding signals in the respective output lines. The output signal pattern is then compared to the expected signals.
The repetition frequency of this test procedure depends of the current application.
Diagnostic coverage : high

Example :

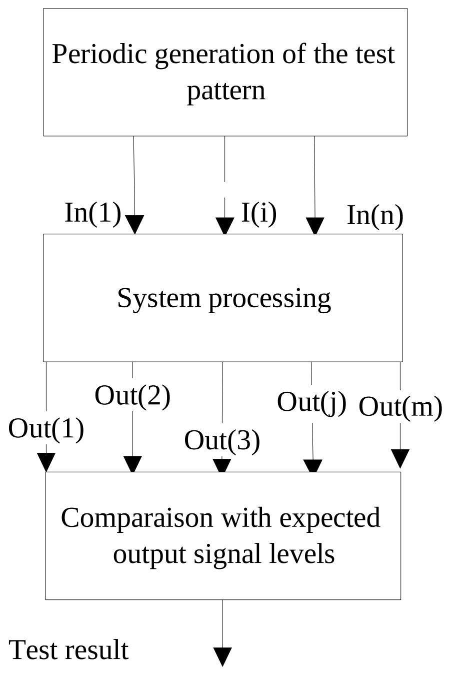

Periodic generation of the test pattern
In(1)
I(i)
In(n)
System processing
Out(1)
Out(2)
Out(3)
Out(j)
Out(m)
Comparaison with expected output signal levels
Test result

Figure 3.3 a Test pattern

3.3.2 Code protection (IEC 61508-7, A.6.2)

The aim of this test method is to detect random hardware and systematic failures in the input/output data flow. A coding procedure or a table system is used to limit the number of input or output signal combinations for which the system operates normally. By means of a decoding routine the status of the input and output signals are determined in such a way that any systematic error or random failure is detected. This information can be further used to add more protection under run-time.
Diagnostic coverage : high
Example:
A two channelled safety-related system is based on independent input signals In1 and In2.
By coding In1 and In2, information redundancy is added at the input stage. The parallel processing of these signals in respective computer results in two output signals. If special conditions are set to generate a common status signal any discrepancies are detected and result in appropriate safety measures.

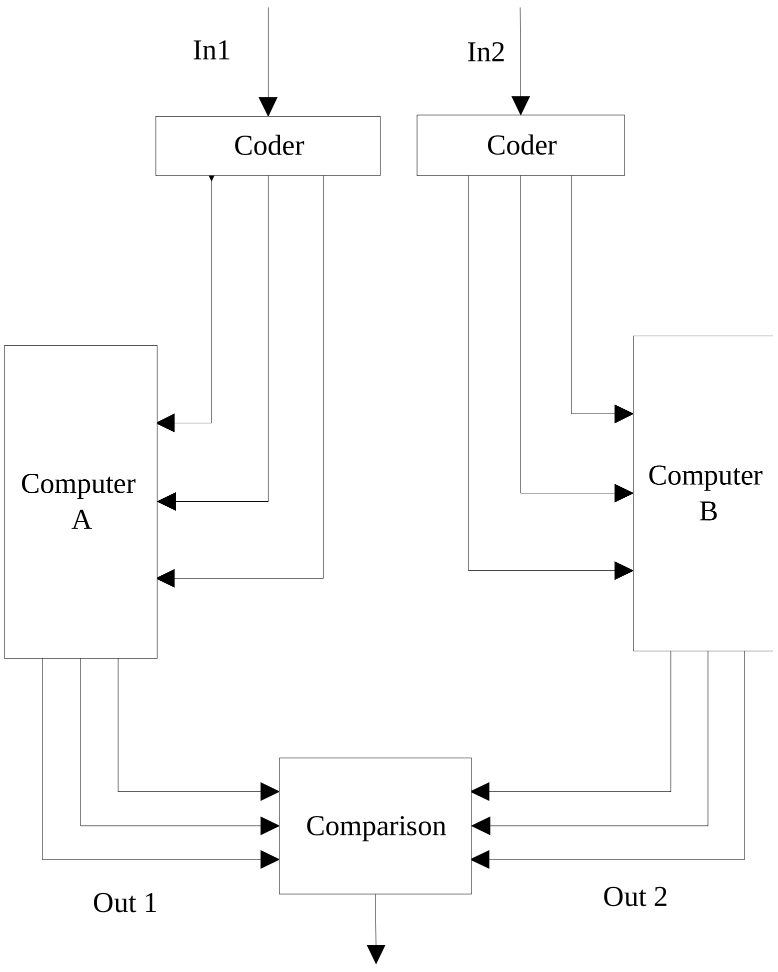


Figure 3.3.b Code protection

3.3.3 Multi-channel parallel output (IEC 61508-7, A.6.3)

The multi-channel parallel output feature enables the detection of random hardware faults such as stuck-at faults as well as faults resulting from external devices. Errors/failures such as timing errors, addressing errors, drift failures and transcient failures may also be detected with such a method. This control technique implies that the system is built with independent outputs. The error detection procedure is carried out under the diagnostic interval by external comparators which can switch off the current equipment. Diagnostic coverage : high

3.3.4 Monitored outputs (IEC 61508-7, A.6.4)

Individual failures can be detected by using the monitored output method which is data flow-dependent. This method enables detection of faults/errors and failures originating from external equipment, timing errors, addressing errors as well as drift failures and transcient failures
The error detection procedure is effective only if the data flow changes appear under the diagnostic interval. Depending on the robustness of this control mechanism, detected failures may sometimes not be track to a specific output line. Diagnostic coverage : high

3.3.5 Input comparison / voting (IEC 61508-7, A.6.5)

Input comparison voting is an error / fault method used to detect individual failures from external equipment or units. Errors/failures such as timing errors, addressing errors, drift failures and transcient failures may also be detected with such a method. This control technique implies that the system is built with independent input units. The error detection procedure is carried out under the diagnostic interval by an external comparator which can switch off the signal processing in respective processor unit.

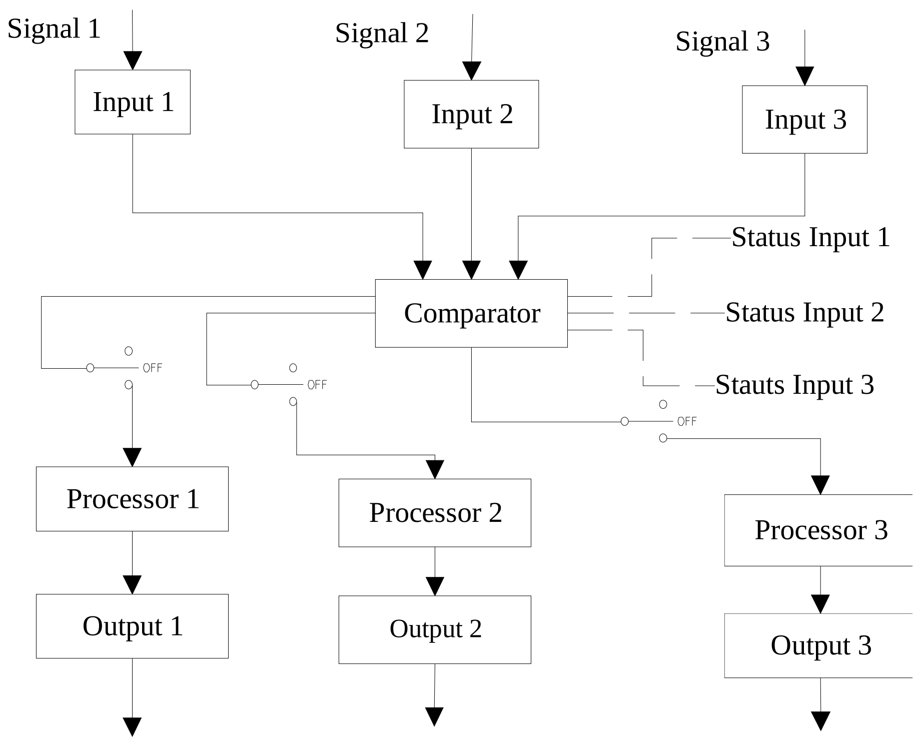


Figure 3.3.d Input comparison

Diagnostic coverage : high

3.3.6 Summary of examples

The table in figure 3.3 lists methods for checking the I/O units and interfaces. The list is not exhaustive, and other acceptable methods exist.

Method for fault diagnostic coverage of I/O units and interface	Maximum achievable fault diagnosis coverage	Reference in IEC 61508- 7
Test pattern	**High**	**A.6.1**
Code protection		**A.6.2**
Multi-channel parallel output	**High**	**A.6.3**
Monitored outputs	**High**	**A.6.4**
Input comparison / voting	**High**	**A.6.5**

Figure 3.3 Examples of test methods for I/O units and interfaces

3.4 Data paths

There are several methods to achieve fault diagnostic coverage of Data paths of which the following can be used to reach a specified rate of coverage.

This work shall focus upon the methods listed in the IEC 61508 standard (part 2, A.8).

Multi-bit hardware redundancy
Complete hardware redundancy
Inspection using test patterns
Transmission redundancy
Information redundancy

3.4.1 Multi-bit hardware redundancy (IEC 61508-7, A.7.2)

The aim of this method is to detect errors during transmission on the bus and in serial transmission links. By extending the bus with two or more lines, error detection is achieved by using hamming code techniques.

Diagnostic coverage : medium

3.4.2 Complete hardware redundancy (IEC 61508-7, A.7.3)

The aim of this method is to detect errors during the communication process between units within a system. The communication bus is doubled and the additional lines are used to detect errors.

Diagnostic coverage : high

3.4.3 Inspection using test patterns (IEC 61508-7, A.7.4)

The aim of this method is to detect static failures (stuck-at failure) and cross-talk. By using a data flow-independent cyclical test of data path, a comparison is done to compare current observation with the expected values. This method is effective only if the pattern information, the test pattern reception, and the pattern evaluation are independent of each other.

Diagnostic coverage : high

3.4.4 Transmission redundancy (IEC 61508-7, A.7.5)

The aim of this method is to detect and avoid transcient failures. The transmission of information is repeated several time in sequence.

Diagnostic coverage : high

3.4.5 Information redundancy (IEC 61508-7, A.7.6)

The aim of this method is to detect and avoid errors in bus communication. Usually the information transfer is carried out in blocks, followed by a checksum calculation.

Diagnostic coverage : high

3.4.6 Summary of examples

The table in figure 3.4 lists methods for checking the data paths. The list is not exhaustive, and other acceptable methods exist.

Method for fault diagnostic coverage of data paths	Maximum achievable fault diagnosis coverage	Reference in IEC 61508- 7
Multi-bit hardware redundancy	Medium	A.7.2
Complete hardware redundancy	High	A.7.3
Inspection using test patterns	High	A.7.4
Transmission redundancy	High	A.7.5
Information redundancy	High	A.7.6

Figure 3.4 Examples of test methods for data paths

3.5 Power supply

The possible faults cannot be defined in the an exact way, as for example memory faults can be. Faults in the supply power may be characterised as:
- supply voltage lower than specified minimum limit.
- supply voltage higher than specified maximum limit.
- supply voltage drifting within the specified range.
- oscillations.

The voltage variations can further be characterised as fast or slow, and intermittent or singular.

Neither is it possible to exactly define the faults which are detected by a certain technique to monitor the supply power. The diagnostic coverage of the power supply monitoring cannot be calculated. It will have to be defined in terms such as low, medium or high.

Continuous monitoring without checking for faults in the circuitry can be defined to have low diagnostic coverage. A hardware or software fault in the monitoring mechanism may easily disable the supply voltage monitoring without this being observed by the processing unit.

Continuous monitoring using dynamic signals can be defined to have high diagnostic coverage. A fault in the monitoring function must then be regarded as likely to corrupt the dynamic signals. The processing unit will observe the change in the dynamic signal, and detect the fault.

Machine control systems do very seldom employ advanced or redundant monitoring of the supply voltage in a single channel. Dual-channel designs usually have two independent voltage monitoring circuits. Most circuits for supply voltage monitoring may be considered to have low diagnostic coverage.

The table in figure 3.5 lists methods for checking the power supply. The list is not exhaustive, and other acceptable methods exist.

Method for fault diagnostic coverage of power supply	Maximum achievable fault diagnosis coverage	Reference in IEC 61508- 7
Continuous monitoring (no checking in the circuitry)	Low	
Continuous monitoring using dynamic signals	High	

Figure 3.5 Examples of test methods for power supply

3.6 Program sequence

Faults in the programme sequence may be caused by hardware faults, software faults or environmental disturbances. It is not possible to define the fault modes in a precise way. Neither is it possible to calculate the number of faults which will be detected. The diagnostic coverage cannot be calculated, but has to be agreed upon by definition, e.g. as low, medium or high.

Techniques to detect faults in programme sequence are based on software, hardware or combinations of both.

A frequently use technique is a hardware circuit (‘watch dog’) triggered by the software of the processing unit within a periodic interval. This technique may be further enhanced by logical monitoring of the execution. The different parts of the software will check for a correct sequence between them. Software checking may also be used without hardware circuitry.

Depending on combinations of techniques, different diagnostic coverage may be claimed.

There are some hardware aspects which will influence the diagnostic coverage:
- monitoring time base shared with CPU, or independent of CPU time base
- on-chip hardware or hardware separated from the processing unit.
- fail-safe monitoring hardware, i.e. a fault in the hardware used for monitoring the execution will cause an alarm.

There are also software aspects which will affect the diagnostic coverage:
- triggering from a non safety-related part of the software.
- testing of the program sequence monitoring function at reset.
- software monitoring of the correct sequence of the individual program sections by passing ”keys” between software modules, or by ”counting”.

Execution monitoring without hardware support is very sensitive to hardware faults in the processing unit. Faults in the processing unit may disable both the execution of the application software, and the monitoring software. Execution monitoring based only on software must be defined to have low diagnostic coverage.

A monitoring device integrated on the same chip as the processing unit, and triggered at a periodic interval, may be regarded to have low diagnostic coverage.

An external hardware circuit dedicated to execution monitoring will be less subject to common cause failures with the processing unit. It can be regarded to have medium diagnostic coverage.

An external fail-safe circuit, or an external periodically tested circuit, can be regarded to have high diagnostic coverage.

Machine control systems do seldom employ advanced or redundant monitoring of the programme sequence in a single channel. Dual-channel designs usually have two independent 'watchdogs', and the processing units may synchronise and thereby monitor the programme sequence of the other channel.

The table in figure 3.6 lists methods for checking the invariable memory ranges. The list is not exhaustive, and other acceptable methods exist.

Method for fault diagnostic coverage of program sequence execution	Maximum achievable fault diagnosis coverage	Reference in IEC 61508- 7
Monitoring device integrated on the same chip as the processing unit	**Low**	
External hardware circuit or external periodically tested circuit	**Medium**	
External fail-safe circuit	**High**	

Figure 3.6 Examples of test methods for program sequence

4 Requirements

In the work package 2.1 it is shown that, for system architectures typical for complex electronic systems (CES) a correspondence between the categories of EN 954-1 and the Safety Integrity Levels of IEC 61508 is possible if a complete safety function is executed by the CES. To get an hierarchical risk reduction between categories B, 2, 3 and 4 the following tables can be used. If the designer can justify that the assumptions for the architectures are the same than made in chapter 1.3 the following tables could be used without quantification. Using the result of chapter xxx a link can be made between the categories of EN 954-1 and the Safety Integrity Levels of IEC 61508 for these designated architectures for the machinery sector for CES.

For complex electronic systems according to chapter 6.2.2 of EN 954-1 cannot be used to realize Category 1. For this reason, the following tables do not contain any checking principles for category 1.

The requirements listed in this report are combined to form adequate safety principles for a certain category (B, 1, 2, 3 or 4). Different safety principles are listed for the different architectures in category 3 and 4. This is commented in chapter 1.3.

Some of the checking principles may be applied either at power-up, or continuously during run-time. This will be much depending on the application and no specific requirements are given for when (or how often) the checking must take place.

4.1 Processing unit

4.1.a Requirement: The CPU shall be checked for stuck-at failures of registers and internal RAM.

4.1.b Requirement: The decoding and execution of instructions shall be checked.

4.1.c Requirement: All registers must be checked.

4.1.d Requirement: Faults in the processing unit shall be indicated by the PES.

Requirement	Category B	2	3 Single	3 Dual	4 Dual	4 Triple
4.1.a Registers&RAM			x	x	x	x
4.1.b Instruction decoding			x		x	
4.1.c All registers			x		x	
4.1.d Indication			x	x	x	x
Minimum diagnostic coverage	-	-	High	Medium	High	Medium

Table 4.1 Safety principles for monitoring of the processing unit.

4.2 Invariable memory ranges

4.2.a Basic requirement: The PES shall be able to detect faults in the invariable memory.

4.2.b Additional requirement: The complete address range must be checked.

4.2.c Basic requirement: Memory failures shall be indicated by the PES.

Requirement	Category B	2	3 Single	3 Dual	4 Dual	4 Triple
4.2.a Memory check	x	x	x	x	x	x
4.2.b Complete address range			x		x	
4.2.c Indication		x	x	x	x	x
Minimum diagnostic coverage	Low	Low	High	Medium	High	Medium

Table 4.2 Safety principles for monitoring of invariable memory.

4.3 Variable memory ranges

4.3.a Basic requirement: The PES shall be able to detect faults in the variable memory.

4.3.b Additional requirement: The complete address range must be covered.

4.3.c Basic requirement: Memory failures shall be indicated by the PES.

Requirement	Category B	2	3 Single	3 Dual	4 Dual	4 Triple
4.3.a Memory check	x	x	x	x	x	x
4.3.b Complete address range			x		x	x
4.3.c Indication		x	x	x	x	x
Minimum diagnostic coverage	Low	Low	High	Medium	High	Medium

Table 4.3 Safety principles for monitoring of variable memory.

4.4 I/O Units and Interface

4.4.a Requirement: The PES shall automatically check the input and output units (digital, analogue, serial or parallel).

4.4.b Requirement: Faults detected in the internal communication shall be indicated.

Requirement	**Category**					
	B	**2**	**3**		**4**	
			Single	**Dual**	**Dual**	**Triple**
4.4.a I/O check			x	x	x	x
4.4.b Indication			x	x	x	x
Minimum diagnostic coverage	-	-	High	Medium	High	Medium

Table 4.4 Safety principles for monitoring of I/O units and interface.

4.5 Data paths

4.5.a Requirement: The PES shall automatically check the internal communication.

4.5.b Requirement: Faults detected in the internal communication shall be indicated.

Requirement	**Category**					
	B	**2**	**3**		**4**	
			Single	**Dual**	**Dual**	**Triple**
4.5.a Data path check			x	x	x	x
4.5.b Indication			x	x	x	x
Minimum diagnostic coverage	-	-	High	Medium	High	Medium

Table 4.5 Safety principles for monitoring of data paths.

4.6 Power supply

4.6.a Basic requirement: The PES shall be able to detect decreases in the supply voltage, and the execution of the processor must be halted in a controlled way.

4.6.b Additional requirement: The supply voltage monitoring circuit must be dynamic, i.e. correct supply voltage is indicated by a dynamic signal.

4.6.c Additional requirement: The supply voltage monitor circuit must be fail-safe, i.e. a fault in the circuitry shall lead to a power fail alarm.

4.6.d Additional requirement: Power supply failures shall be indicated by the PES.

Requirement	**Category**					
	B	**2**	**3**		**4**	
			Single	**Dual**	**Dual**	**Triple**
4.6.a Supply power monitoring	x	x	x	x	x	x
4.6.b Dynamic			x or 46c		x or 46c	
4.6.c Fail-safe			x or46b		x or46b	
4.6.d Indication			x	x	x	x
Minimum diagnostic coverage	Low	Low	High	Medium	High	Medium

Table 4.6 Safety principles for monitoring of supply power.

NOTE: It is difficult to find a good definition of diagnostic coverage for the power supply checking.

4.7 Program sequence

4.7.a Basic requirement: The PES shall have a watchdog implemented in hardware to monitor the program sequence.

4.7.b Additional requirement: The hardware (especially the time base) used for the watchdog shall be independent of the processor it is supposed to supervise.

4.7.c Additional requirement: There shall be software means of monitoring the program sequence.

4.7.d Additional requirement: The watchdog must be automatically tested by the software at power-up or a periodic intervals.

4.7.e Additional requirement: The watchdog must be fail-safe, i.e. a fault in the watchdog circuitry will lead to a watchdog alarm.

4.7.f Additional requirement: A watchdog alarm shall be indicated by the PES.

Requirement	**Category**					
	B	**2**	**3**		**4**	
			Single	**Dual**	**Dual**	**Triple**
4.7.a Hardware watchdog	x	x	x	x	x	x
4.7.b Independent hardware			x		x	
4.7.c Software monitoring			x	x	x	x
4.7.d Tested			x or 47e		x or 47e	
4.7.e Fail-safe			x or 47d		x or 47d	
4.7.f Indication			x	x	x	x
Minimum diagnostic coverage	Low	Low	High	Medium	High	Medium

Table 4.7 Safety principles for monitoring of program execution.

NOTE: The principle of requiring more functionality for a higher category corresponds to requiring a higher diagnostic coverage.

5 Validation methods

5.1 Processing units

Aim: To validate the checking of processing units, and to determine its diagnostic coverage.

Description: The processing unit in a safety-related PES used for machine control is most often a microprocessor or a microcontroller. A hardware reference manual of the semiconductor manufacturer will be needed to study the processing unit. Software routines for checking of the processing unit shall be analysed. The analysis can be summarised in following check list:

	Processing Units				
		Ap.*)	Yes	No	Comment
A	Are the processing units checked for faults?				
B	Have techniques and measures according to IEC 61508-2, table A.4 been used?				
C	Have the software routines used for checking of processing units been identified?				
D	Have the software routines for handling of faults been identified?				
E	Is the instruction decoding covered by self-checking ?				
F	Are the internal registers covered by self-checking ?				
G	Is self-checking performed at power-up ?				
H	Is periodic self-checking performed at run-time ?				
I	Has the diagnostic coverage been determined?				
J	Has the software implementing the self-checking been checked without finding logical faults?				
K	Are the techniques and measures used adequate as safety principles for the intended category (see chapter 4)?				

*) Ap. = Applicable question (Some questions may not be applicable for all control systems.)

5.2 Invariable memory ranges

Aim: To validate the memory checking, and to calculate its diagnostic coverage.

Description: Programme code and constants will be stored in invariable memory (ROM, EPROM etc.). A memory map will give an overview of the memory ranges. The hardware circuit diagram or the microprocessor manual may have to be studied. Software routines for memory checking and handling of memory faults shall be analysed. The analysis can be summarised in following check list:

	Invariable Memory Ranges	Ap.*)	Yes	No	Comment
A	Is automatic and periodic checking of the invariable memory range made at a periodic interval?				
B	Is automatic checking of the invariable memory range made at power-up?				
C	Have techniques and measures according to IEC 61508-2, table A.5 been used?				
D	Have the software routines used for memory checking been identified?				
E	Have the software routines for handling of faults been identified?				
F	Is the complete address range of the invariable memory covered by the test?				
G	Has the software for memory checking been analysed without finding faults?				
H	Has the software for handling of faults in memory been checked without finding faults?				
I	Has the diagnostic coverage been calculated?				
J	Are the techniques and measures used adequate as safety principles for the intended category (see chapter 4)?				

*) Ap. = Applicable question (Some questions may not be applicable for all control systems.)

5.3 Variable memory ranges

Aim: To validate the memory checking, and to calculate its diagnostic coverage.

Description: Variables and parameters will be stored in variable memory (RAM, EEPROM etc). A memory map will give an overview of the memory ranges. The hardware circuit diagram or the microprocessor manual may have to be studied. Software routines for memory checking and handling of memory faults shall be analysed. The analysis can be summarised in following check list:

	Variable Memory Ranges				
		Ap.*)	Yes	No	Comment
A	Is automatic and periodic checking of the variable memory range made at a periodic interval?				
B	Is automatic checking of the variable memory range made at power-up?				
C	Have techniques and measures according to IEC 61508-2, table A.6 been used?				
D	Have the software routines used for memory checking been identified?				
E	Have the software routines for handling of faults been identified?				
F	Is the complete address range of the variable memory covered by the test?				
G	Has the software for memory checking been analysed without finding faults?				
H	Has the software for handling of faults in memory been checked without finding faults?				
I	Has the diagnostic coverage been calculated?				
J	Are the techniques and measures used adequate as safety principles for the intended category (see chapter 4)?				

*) Ap. = Applicable question (Some questions may not be applicable for all control systems.)

5.4 I/O Units and Interface

Aim: To validate the checking of I/O units and interfaces (external communication), and to determine its diagnostic coverage.

Description: The control system may be equipped with I/O units and interfaces for communication with external units. A hardware circuit diagram, and possibly also the reference manual of the semiconductor I/O circuits, will be needed for the analysis. Software routines for checking of external communication shall be analysed. The analysis can be summarised in following check list:

	I/O Units and Interface				
		Ap.*)	Yes	No	Comment
A	Have all I/O units and interfaces been identified?				
B	Are the I/O units and interfaces checked for faults at power-up?				
C	Are the I/O units and interfaces checked for faults at run-time?				
D	Have techniques and measures according to IEC 61508-2, table A.7 been used?				
E	Have the software routines used for checking of I/O units and interfaces been identified?				
F	Have the software routines for handling of faults been identified?				
G	Is self-checking performed at power-up ?				
H	Is periodic self-checking performed at run-time ?				
I	Has the diagnostic coverage been determined?				
J	Has the software implementing the self-checking been checked without finding logical faults?				
K	Are the techniques and measures used adequate as safety principles for the intended category (see chapter 4)?				

*) Ap. = Applicable question (Some questions may not be applicable for all control systems.)

5.5 Data paths

Aim: To validate the data path (internal communication) checking, and to determine its diagnostic coverage.

Description: Communication between modules of the PES is made through different data paths. A block diagram of the hardware has to be studied in combination with the circuit diagrams. Software routines for checking of internal communication shall be analysed. The analysis can be summarised in following check list:

	Data Paths				
		Ap.*)	Yes	No	Comment
A	Have all data paths for communication been identified?				
B	Are the data paths checked for faults at power-up?				
C	Are the data paths checked for faults at run-time?				
D	Have techniques and measures according to IEC 61508-2, table A.8 been used?				
E	Have the software routines used for checking of data paths been identified?				
F	Have the software routines for handling of faults been identified?				
G	Has the diagnostic coverage been determined?				
H	Has the software implementing the self-checking been checked without finding logical faults?				
I	Are the techniques and measures used adequate as safety principles for the intended category (see chapter 4)?				

*) Ap. = Applicable question (Some questions may not be applicable for all control systems.)

5.6 Power supply

Aim: To validate the power supply monitoring, and to determine its diagnostic coverage.

Description: The monitoring of supply voltage requires a combination of hardware and software measures. An analysis of the hardware circuit diagrams is often the best way to commence. The sensing of the power alarm signal, and the power down routines should be analysed in the source code. The analysis can be summarised in a check list:

	Power Supply Monitoring				
		Ap.*)	Yes	No	Comment
A	Is the supply power monitored to give alarm at low voltage ("brown-out") ?				
B	Is the supply power monitored to give alarm at high voltage?				
C	Have techniques and measures according to IEC 61508-2, table A.8 been used?				
D	Have the hardware circuit diagrams been analysed without finding faults?				
E	Have the software routines used for power supply monitoring been identified?				
F	Have the software routines for handling of faults been identified?				
G	Has the interface between hardware and software been analysed without finding faults?				
H	Has the control flow at low voltage alarm been analysed without finding faults ?				
I	Has the time from low voltage alarm to entered 'safe state' been analysed without finding faults ?				
J	Has the action taken at high voltage been analysed without finding faults?				
K	Has the diagnostic coverage been determined?				
L	Has the software implementing the self-checking been checked without finding logical faults?				
M	Are the techniques and measures used adequate as safety principles for the intended category (see chapter 4)?				

*) Ap. = Applicable question (Some questions may not be applicable for all control systems.)

5.7 Program sequence

Aim: To validate the programme sequence monitoring (watchdog), and to determine its diagnostic coverage.

Description: The programme sequence monitoring may be implemented in combination of hardware and software, or just in software. The hardware circuit diagrams should be analysed for the measures in hardware. The source code should be analysed for triggering of hardware, monitoring mechanisms in software and handling of watchdog program sequence alarm.

	Programme Sequence Monitoring				
		Ap.*)	Yes	No	Comment
A	Is there a hardware device (watchdog) to monitor the execution of the software?				
B	Is there a software monitoring of the execution of the software ?				
C	Have techniques and measures according to IEC 61508-2, table A.10 been used?				
D	Has the hardware circuit diagram been analysed without finding faults?				
E	Will a watchdog alarm result in a reset of the microprocessor?				
F	Will a watchdog alarm result in the loss of the calculated values, and machine state?				
G	Is the watchdog triggered at a period less than 5 seconds?				
H	Is the time base of the hardware watchdog independent from the processor time base?				
I	Is the watchdog a separate hardware circuit (not integrated into the microcontroller) ?				
J	Is an unintended change in the program flow likely to lead to a watchdog alarm ?				
K	Is the watchdog triggered during the execution of the main program ?				
L	Has the diagnostic coverage been determined?				
M	Has the software implementing the self-checking been checked without finding logical faults?				
N	Are the techniques and measures used adequate as safety principles for the intended category (see chapter 4)?				

*) Ap. = Applicable question (Some questions may not be applicable for all control systems.)

6 Conclusions

Summing up the results of this work, it appears without any doubt that when safety-related functions are at stake, safety-principles shall be implemented in the design of the control system. The level of safety or the category shall be the starting point for the choice of architecture to be implemented. This architecture shall be supported by adequate techniques and measures to detect and control faults, to minimise their repercussion on the whole system and eventually to provide a soft transition to a safe-state.

Checking of processing units, memory ranges, I/O units, interfaces, data paths, power supply and program sequence are regarded as state-of the-art. The selection of checking techniques and the diagnostic coverage needed, will be depending on the risks associated with the machine to be controlled. A complex electronic control system of a high-risk machine shall be able to find most faults in the control system before hazardous situations can be caused. That calls for elaborate checking facilities with high coverage. A low-risk machine must also have safety principles implemented, but lower coverage can be accepted.

It has been possible to use the techniques and measures suggested in the IEC 61508 safety standard, to prescribe safety principles required by the EN 954-1 standard. The IEC 61508 has established a frame of reference which is well possible to use also with the EN 954-1. However, the recommendations for the safety integrity levels of IEC 61508 will have to be modified for the categories of EN 954-1.

The safety measures to be implemented are described and associated to the current architecture for a specific category. The checklists presented in this report will be of help both to the developer and to the assessor.