The Knowledge Network

ELECTROMAGNETIC COMPATIBILITY & FUNCTIONAL SAFETY

Annex H - Software

A Factfile provided by The Institution of Engineering and Technology
www.theiet.org/factfiles

Annex H

EMC and functional safety for software

Abstract

The design of software should be part of the design features for tolerance against EM disturbances, as required by IEC 61508 part 2. This annex reviews the recommendations of the automotive industry guidance (the MISRA guidelines) for dealing with EMC issues in embedded software.

These guidelines may also be of great value in improving the robustness of embedded software in other application areas. They consider the possible effects of EM disturbances on the following:

- Digital and analogue inputs, including the special case of external interrupts
- Communications lines
- Corruption of various types of memory
- Loss of control of the processor
- Misinterpretation/disruption of micro-code and corrupted clock pulses
- Corruption of data, including that on address buses, and stack-pointer corruption
- Floating point co-processing
- Open and closed-loop control systems

The MISRA guidelines also recommend software solutions for mitigating EMI, including:

- Digital filtering of data
- Comparing of data with constant, or inferred values to identify errors
- Performing error detection and correction on data in memory
- Detecting and correcting errors in communications data
- Dynamically adjusting scaling to optimise signal-to-noise ratios
- Managing failures and safe-state transitions

Contents

H1. Introduction 2

H2. The automotive industry guidance 3

H3. Summary of the industry guidance on software development in relationship to EMC 4

- H3.1 Emphasis on safety-related systems 4
- H3.2 Effects on embedded microprocessors 4
- H3.3 General recommendations 4

H4. Specific recommendations 6

- H4.1 Input processing and external interrupts 6
- H4.2 Queue-based systems 6
- H4.3 Digital communications and multiplexing 6
- H4.4 Concerns with floating point and EMI 7
- H4.5 On-board diagnostics and fault management 7
- H4.6 The imposition of safe states 7
- H4.7 Fault masking versus false alarms 7
- H4.8 Run-time errors associated with a high-level language 7
- H4.9 Dynamic heap memory allocation 8
- H4.10 Using auto-code generating tools and standard operating systems 8

H5. Conclusions 9

H6. References 10

H1. Introduction

IEC 61508 [1] part 2 specifies EMC requirements as part of the overall safety integrity requirements for safety-related systems. For example, paragraph 7.2.3.2 states that “The E/E/PES safety integrity requirements specification shall contain …e) the electromagnetic immunity limits (see IEC 61000-1-1) which are required to achieve electromagnetic compatibility.”

Furthermore, in section 7.4.5 “Requirements for control of systematic faults” the standard states, “For controlling systematic faults, the E/E/PES design shall possess design features that make the E/E/PE safety-related systems tolerant against … b) environmental stresses, including electromagnetic disturbances”. It should be noted that the design of software can be part of these design features for tolerance against electromagnetic disturbances. The remainder of this annex reviews the recommendations of the automotive industry guidance (the MISRA guidelines [2]) on embedded software for dealing with EMC issues in software. It should be noted that the MISRA Guidelines themselves were compiled with reference to an earlier draft of the documents that have become IEC 61508.

H2. The automotive industry guidance

It is necessary to consider EMC in both the hardware and software design of embedded automotive systems. This is particularly important as the electromagnetic environment of vehicles is varied and unpredictable. A road-going vehicle will be driven to many different locations during the course of its operational life and exposed to varying levels of electromagnetic disturbance. It is therefore necessary to engineer vehicles to an appropriate degree of robustness to their electromagnetic environment, considering the "worst case" fields likely to be encountered but at the same time ensuring the risk is reduced to an appropriate but realistic level.

The automotive industry guidance on embedded software [2] recognises that, while the primary defence against electromagnetic interference (EMI) is always in hardware, software can play an important part in mitigating the effects of EMI. Several of the recommendations in the MISRA guidelines [2] and in the subsequent guidance on the C programming language [3] relate to the robustness of real-time embedded systems in difficult environments.

Although EMI will occur at random, the same EMI will produce identical effects if the system is in the same state each time it occurs; the same situation that holds for software when it responds to a given stimulus. However, the corollary of this is that in the "real world" of operation it will be very unlikely that a particular combination of stimuli and operating conditions will be repeated. In addition the normal complexity of electronic circuits, and of software, when combined with the very large number of possible stimuli, means that it will never be possible to fully test either the EMC or the software of the system. Testing can therefore only be used to identify faults, not to prove correctness.

The DRIVE II project EMCATT [4] followed the philosophy used by MISRA to develop a corresponding set of recommendations for achieving Safety Integrity Levels (SILs) for EMC. The main features of this philosophy are:

- EMC should not be added to a system as an afterthought, it must be designed in from the beginning.
- Testing can signal the presence of errors but not their absence, i.e. it cannot guarantee the EMC of a system. Additional forms of verification and validation, performed under a Quality Assurance regime (e.g. ISO 9000), are therefore necessary to gain the confidence that the EMC of the system has been developed correctly.
- Quality Assurance techniques are necessary to reduce the possibility of manufacturing faults compromising the EMC of the system.

In order to assess the EMC of a product a number of criteria are identified for each SIL in a "degree of assessment" table. This table shows that as the SIL increases greater knowledge is required of the system and the environment in which it is to operate. This knowledge is used to justify the actions that are undertaken to ensure that the necessary level of confidence is obtained. The degree of assessment table for EMC created in the EMCATT project was defined to correspond to the degree of assessment needed for software as defined in the MISRA Guidelines. Although the EMCATT work requires further development, it nevertheless provides a useful starting point for dealing with EMC in complex systems.

H3. Summary of the industry guidance on software development in relationship to EMC

H3.1 Emphasis on safety-related systems

The MISRA guidelines are concerned with the development of safety-related automotive embedded systems. The EMC recommendations in the guidelines are given in that context. However, many of the concerns identified and the recommendations are equally applicable to other applications. In many cases, the degree of development rigour recommended in the guidelines relate to one of four SILs which need to be first determined for any new system [1]. As noted above, the EMCATT project has gone further in identifying an overall EMC development philosophy associated with the four Safety Integrity Levels.

H3.2 Effects on embedded microprocessors

The guidelines consider the interference effects on various aspects of processing:

- Digital and analogue inputs, including the special case of external interrupts
- Communications lines
- Corruption of various types of memory
- Loss of control of the processor:
- Misinterpretation/disruption of micro-code and corrupted clock pulses
- Corruption of data, inc. on address buses and stack-pointer corruption
- Floating point co-processing
- Open and closed-loop control systems.

H3.3 General recommendations

In general, systems must be engineered for their intended operational environment [4, 5]. Consequently, developers must be familiar with that environment and must recognise that solutions cannot be added on, but must be built in at the appropriate stage during the development. The development process itself must be Quality Assurance based [6] .

The primary defence of a system against EMI must lie in hardware design [7, 8] which is reflected in the recommendations in the guidelines. However software solutions are recommended for mitigating the effects of EMI which may have overcome the hardware defences. Specifically, software can be used for the following functions related to EMI:

- Digital filtering of data
- Comparing of data with constant, or inferred values to identify errors
- Performing error detection and correction on data in memory
- Detecting and correcting errors in communications data
- Dynamically adjusting scaling to optimise signal-to-noise ratios
- Managing failures and safe state transitions.

Full system validation must include EMC testing to recognised standards [9]. The hardware that the software runs on should be validated to meet the EMC requirements specific to its operational

environment, as outlined in the section “Guidelines on Controlling EMC to achieve Functional Safety” in the Core of this report.

H4. Specific recommendations

H4.1 Input processing and external interrupts

External interrupts have been identified as a particularly sensitive and critical form of input, primarily because excessive or uncontrolled interrupts have the potential to disrupt all processing. The MISRA Guidelines identify several interference related problems associated with external interrupts [2 §3.4.1.5]:

a) Spurious interrupts due to noise can saturate the processor, inhibiting other parts of the software from initiating fall-back operations.

b) Asynchronous interrupts make it difficult for the software design to cater for all high load scenarios, which may result in further data corruption.

c) Several concurrent processes initiated by interrupts that compete for resources can reach deadlock in high load situations.

To aid robust interrupt handling the guidelines [2 §3.4.1.6] and the associated report [10] make a variety of recommendations. These include:

a) Using sufficiently independent and diverse mechanisms that are available in both hardware and software to protect and back up the interrupt handling, such as masking, prioritization and supervisor modes.

b) Maintaining strong separation between interrupt routines and other non-interrupt code.

c) Avoiding using the program control stack for data.

d) Setting stack lengths to the theoretical maximums.

e) Using an existing proven real-time kernel where possible.

f) Following suppliers' recommendations for interrupt design.

g) Employing personnel with extensive experience of designing interrupt handlers.

h) For the higher integrity levels, using an independent monitor to initiate a safe state.

H4.2 Queue-based systems

Systems that use external interrupts frequently rely on queue-based designs to cope with the random arrival of real-time data. An understanding of queue mechanisms, their limitations, and the recommended protection techniques are essential to ensure robust operation in the presence of noise and unpredictable inputs [2 §3.4.1.4].

H4.3 Digital communications and multiplexing

A primary requirement of all digital communications is to transfer information free from errors. The automotive industry has led the way in standardising robust low-cost communications protocols designed to cope with noisy and hostile electrical environments [11, 12]. The MISRA Guidelines give a variety of software development recommendations associated with the use of these standard protocols [2 §3.4.5].

H4.4 Concerns with floating point and EMI

Floating point co-processors can be corrupted by interference and fail independently from the main processor. Therefore, some additional diagnostics requirements are recommended for on-board floating point devices [2 §3.4.2.3]. These are similar to those required for off-chip RAM and other separate devices.

H4.5 On-board diagnostics and fault management

The MISRA Guidelines give a wide range of recommendations associated with on-board diagnostics and fault management [2 §3.4.6/8, 13]. For example, detected faults should result in one or more of a range of actions:

a) Warning the operator or user

b) Storing fault data until an appropriate reset

c) Setting a safe state until repair.

H4.6 The imposition of safe states

The degree of independence used to manage safe states should be determined by the SIL for the system and may range from a hardware watchdog-timer to a second processor using diverse technologies. The guidelines recommend at least an independent hardware watchdog designed to ensure a reset and re-initialisation in the event of corruption.

H4.7 Fault masking versus false alarms

Sophisticated error detection with immediate and perhaps transparent corrective action can result in the user being unaware of any problem and unwittingly taking even greater risks. This is particularly true in closed-loop control systems. Conversely, an excessive number of false alarms can reduce customer and user confidence in a system [14].

H4.8 Run-time errors associated with a high-level language

The automotive industry is heavily committed to the use of the C high-level language, for the following reasons:

- C is often the only high-level language with cross-compiler support for the target microprocessors
- There is a ready availability of both skilled personnel and tool support for C development
- It is cost-effective.

An issue arises with high-level language code when particular data, or the volume of data, supplied to the code causes errors in the run-time execution. Some high-level languages have optional run-time checks built into the executable code to detect such errors and take the appropriate action. However, C is generally poor in providing such run-time checks. This is one of the reasons why code generated by a C compiler tends to be small and efficient as C compilers generally do not provide run-time checking for such common problems as arithmetic exceptions (e.g. divide by zero), overflow, validity of addresses calculated in pointer-arithmetic and array bound errors. The MISRA Guidelines for the use of the C language [3, 15] identify the C features associated with these concerns and make recommendations [3 Rule 4].

H4.9 Dynamic heap memory allocation

There are a range of “unspecified”, “undefined” and “implementation-defined” behaviours associated with the use of dynamic memory allocation in conjunction with real-time and random inputs. The MISRA C Guidelines recommend that functions such as *calloc*, *malloc*, *realloc* and *free* should not be used in such systems [3 Rule 118]. Similarly, “overlapping variable storage”, that is the technique of using memory to store some data, then using the same memory to store different data at some other time during the execution of the program, is not recommended. *Unions* for example, are one of the various mechanisms [3 Rule 109] that can display this behaviour.

H4.10 Using auto-code generating tools and standard operating systems

The techniques described are probably not provided by the automatic code generating facility of control system modelling tools, nor will they be part of proprietary real-time operating systems. Therefore, one criteria for the selection of such commercial products could be whether such techniques can be built into these products by users.

H5. Conclusions

Achieving EMC of automotive electronics, and indeed of electronics in other applications, has to be regarded from a “total system” viewpoint. Development issues including hardware design and embedded software cannot proceed in isolation. Whilst hardware measures will remain the “front line” defence against EMI, software design can play an important part in mitigating the effects of EMI and in ensuring the correct and safe operation of systems in hostile environments. Further work is required to develop the concept of SILs for EMC and how the traditional EMC engineering activities can be related to them.

H6. References

[1] IEC 61508, Functional safety of electrical/electronic/programmable electronic safety related systems, 1999.

[2] Development Guidelines for Vehicle Based Software, ISBN 0 9524156 0 7, The Motor Industry Research Association, Nuneaton, November 1994.

[3] Guidelines for the use of the C Language in Vehicle Based Software, ISBN 0 9524156 9 0, The Motor Industry Research Association, Nuneaton, April 1998.

[4] P H Jesty et al., Electromagnetic Compatibility of Advanced Transport Telematics, DRIVE II Project V2064, Final Reports, 1995.

[5] S Brown, Dangers of Interference - EMC and Safety, IEE Review, Vol. 17 No1 1993.

[6] ISO 9000 Quality systems - Model for quality assurance in design and development, production, installation and servicing, 1987.

[7] D Morgan, A Handbook for EMC Testing and Measurement, Peter Perigrinus for the IEE, 1994.

[8] Guidelines for the Achievement of EMC in Motor Vehicles (2nd. Edition), Society of Motor Manufacturers and Traders, August 1992.

[9] European EMC Directives 89/336/EEC, 95/54/EC.

[10] Noise, EMC and Real-time, MISRA Report, The Motor Industry Research Association, Nuneaton, February 1995.

[11] ISO DIS 11898, Road Vehicles - Interchange of Digital Information, Controller Area Network (CAN).

[12] SAE Recommended Practice J1850, Class-B Communications Networks, August 1991.

[13] Diagnostics and Integrated Vehicle Systems, MISRA Report, The Motor Industry Research Association, Nuneaton, February 1995.

[14] D W Newman, “Binding the Security and Safety Aspects of Vehicle Based Software”, Proceedings of Security Automotive ’96.

[15] G F McCall, P D Edwards and R S Rivett, “Safer use of the C Programming Language in the Automotive Industry”, Proceedings of Autotech 1997.