FCPA 2022  
  
Safe Coding with MISRA

Student Workbook 14

January 27, 2022

Author

Paul Kimball  
Interface Associates

1. Automotive Safety

Risk

* Risk = Probability X Severity
* Degrees of Severity
  + Annoyance
  + Inconvenience
  + Loss of functionality
  + Loss of product
  + Danger to health
  + Danger to life
  + Mass Destruction
* Degrees of Probability
  + Impossible
  + Eliminated
  + Rare
  + Unlikely
  + Possible
  + Likely
  + Certain
* You can make a risk assessment for any particular event

Life is Risky, Isn't It?

* Yes, but...
* How much risk can you afford?
  + In lives and injuries
  + In dollars
  + In time spent in litigation, repairs
  + In product functionality
  + In customer loyalty

Safety

* Safety = Controllability / Risk
* We can control risk by using sound engineering principles
* Software Safety:
  + Clear coding standards
  + Best practices
  + Design reviews
  + Rigorous Testing
  + Factors of safety in design
  + Extra capacity
  + Back-up systems
  + Fail-safe mechanisms
  + Support infrastructure
  + Training and skill
  + Willingness to accept help

So, "Safety is Job One"?

* Yes, but...
* How much Safety can you afford?
  + Dollars
  + Time
  + Infrastructure
  + Hardware
  + Materials
  + People
  + Complexity

Safety in a Problem Domain

* Domains with safety concerns
  + Financial
  + Aerospace
  + Automotive
  + Health and Medicine
  + Software Development
  + Security
  + etc.
* Each domain has its own
  + Risk classifications
  + Probability classifications
  + Mitigation Strategies
  + Costs
* There are many industry-specific specifications that define risk and safety considerations
  + ISO ASIL
  + ISO
  + DoD
  + NASA

Standards

* Systemic Approach
  + ISO 26262 - "Functional Safety for Road Vehicles"
* Focused on Software
  + MISRA
  + AUTOSAR
  + SEI CERT

Automotive Safety - ISO 26262

## From ISO 26262 "Functional Safety for Road Vehicles":

* + **Part 9 - Automotive Safety Integrity Level (ASIL)**
  + Severity

S0 No Injuries

S1 Light to moderate injuries

S2 Severe to life-threatening (survival probable) injuries

S3 Life-threatening (survival uncertain) to fatal injuries

* + Exposure

E0 Incredibly unlikely

E1 Very low probability (injury could happen in rare operating conditions)

E2 Low probability

E3 Medium probability

E4 High probability (injury could happen under most operating conditions)

* + Controllability

C0 Controllable in general

C1 Simply controllable

C2 Normally controllable (most drivers could act to prevent injury)

C3 Difficult to control or uncontrollable

* So S3 E4 C3 is super dangerous...

Refining ASIL

* ASIL is further refined for particular subsystems
* SAE International (formerly Society of Automotive Engineers)
  + J2980\_201804 "Considerations for ISO 26262 ASIL Hazard Classification"

Methodology for determining ASIL specifically

Vehicle motion control systems

Cars up to 3.5 metric tons

OMG, just your average self-driving cars!!

* + J3018\_201909 "Safety-Relevant Guidance for On-Road Testing of SAE Level 3, 4, and 5 Prototype Automated Driving System (ADS)-Operated Vehicles"
  + J3061\_201601 "Cybersecurity Guidebook for Cyber-Physical Vehicle Systems"
  + J2808\_201701 "Lane Departure Warning Systems: Information for the Human Interface"
  + **Many more at https://www.sae.org/standards**

# Controlling Software Risks

## I never knew...

### Self-driving cars could be dangerous

#### ... because they run on Software!

### I'm a software developer! How can I help!?

## Write code that is:

### Correct

### Easy to Read

### Self-documenting

## Problem: It's easy to make mistakes in C and C++

### Well-known language quirks

### Hardware differences

### Misunderstanding by programmers

## An accepted approach: Control risk by restricting programmers to a "safer" subset of the language

### Provide rules to guide proper usage

### Use automated tools to encourage (enforce) compliance

# MISRA

* MISRA Consortium Limited (formerly Motor Industry Software Reliability Association)

### Independent not-for-profit

### Based in UK

### Ford Motor Company Ltd is a member

### Manufacturers, component suppliers and engineering consultancies

## Promotes best practice in developing safety- and security-related electronic systems and other software-intensive applications

## Specifications

### MISRA C:1998/2004/2012

### MISRA C++:2008

### MISRA specs often defer to ISO 26262

# AUTOSAR

## AUTOSAR (AUTomotive Open System ARchitecture)

### Worldwide partnership

### vehicle manufacturers,

### suppliers,

### service providers

### companies from the automotive electronics, semiconductor and software industry

## Specifications

### AUTOSAR C++14

#### Guidelines for the use of the C++14 language in critical and safety-related systems

### AUTOSAR specs often defer to MISRA

# Enforcement

## Several main ways to enforce

* Source Code Generation
  + Produce problem-free code to start

### MATLAB, etc.

## Static Analysis

* + Scan code for well-known errors defined in MISRA, AUTOSAR, SEI CERT
  + Automated Tools from many vendors

#### - Parasoft

#### - Perforce

#### - Axivion

#### - CppCheck

## Automated Testing

### Define tests early

### Run tests frequently on code, hardware,

### Generate tests to check for all (!?) possible errors

#### Possible only in tightly constrained problems- e.g. wire harness

Some Specific Rules

* Let's look at the MISRA Spec
* From Annu Chun:
  + Casting should be reviewed.

|  |  |
| --- | --- |
| **2012** | **Rules changed to Advisory under Generated code** |
| 5.3 | An identifier declared in an inner scope shall not hide an identifier declared in an outer scope |
| 8.4 | A compatible declaration shall be visible when an object or function with external linkage is defined |
| 8.14 | The restrict type qualifier shall not be used |
| 15.2 | The goto statement shall jump to a label declared later in the same function |
| 15.3 | Any label referenced by a goto statement shall be declared in the same block, or in a any block enclosing the goto statement |
| 16.1 | All switch statements shall be well-formed |
| 16.2 | A switch label shall only be used when the most closely-enclosing compound statement is the body of a switch statement |
| 2.7 | There should be no unused parameters in functions |
| 5.9 | Identifiers that define objects or functions with internal linkage should be unique |
| 7.2 | A 'u' or 'U' suffix shall be applied to all integer constants that are represented in an unsigned type |
| 7.3 | The lowercase character I shall not be used in a literal suffix |
| 13.3 | The full expression containing an increment (++) or decrement (--) operator should have no other potential side effects other than that caused by the increment or decrement operator |
| 20.5 | #undef should not be used |
|  |  |