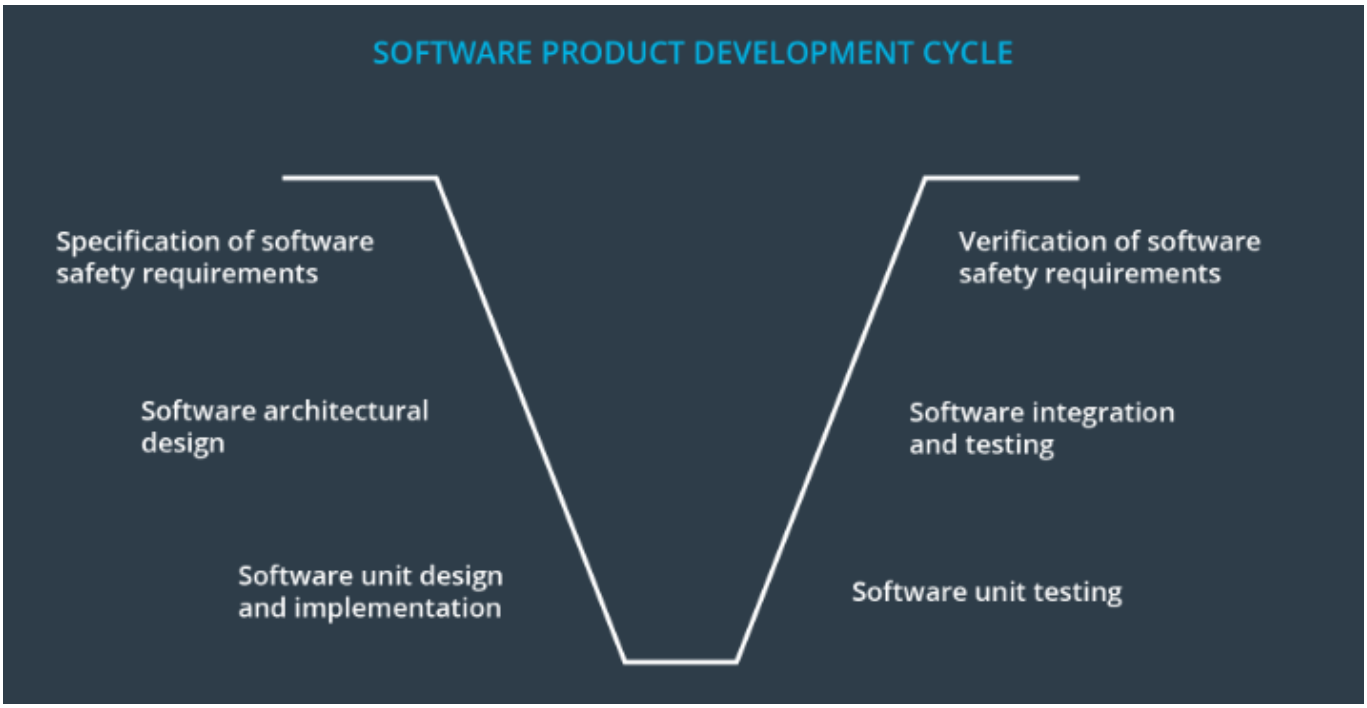
Functional Safety in Software development

Software safety

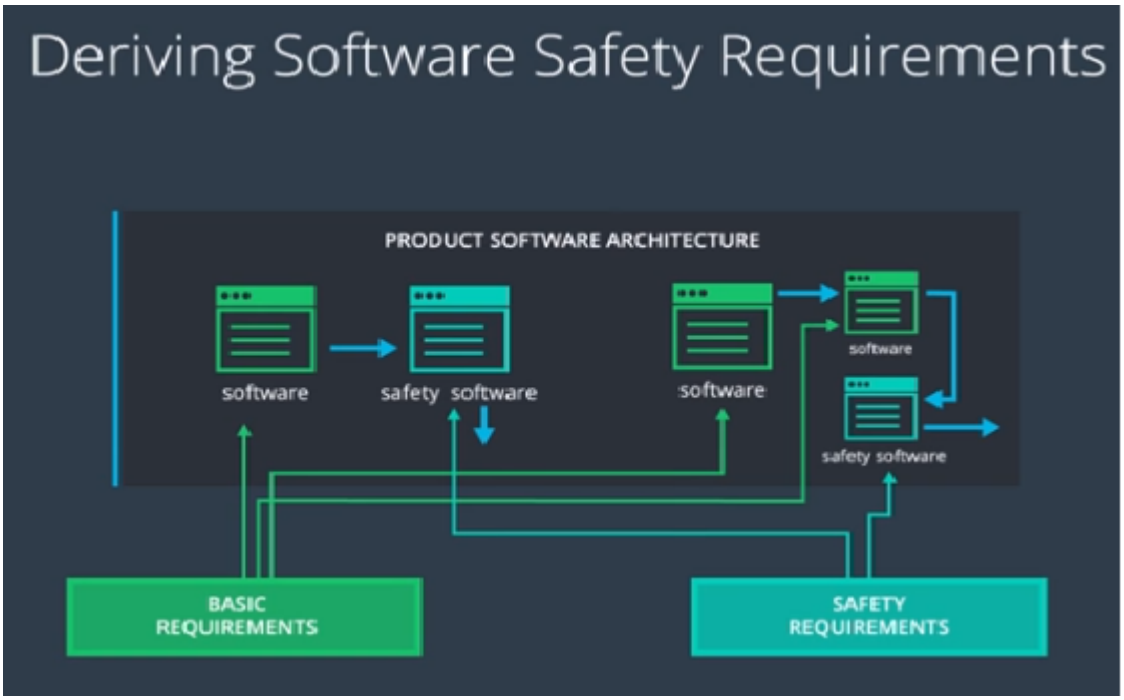
The V model

Functional safety in general can be an iterative process where developing a new section of the V model can lead to changes in a previous section and vice versa.

On the software side, there is an architectural design section as well as a unit design section:



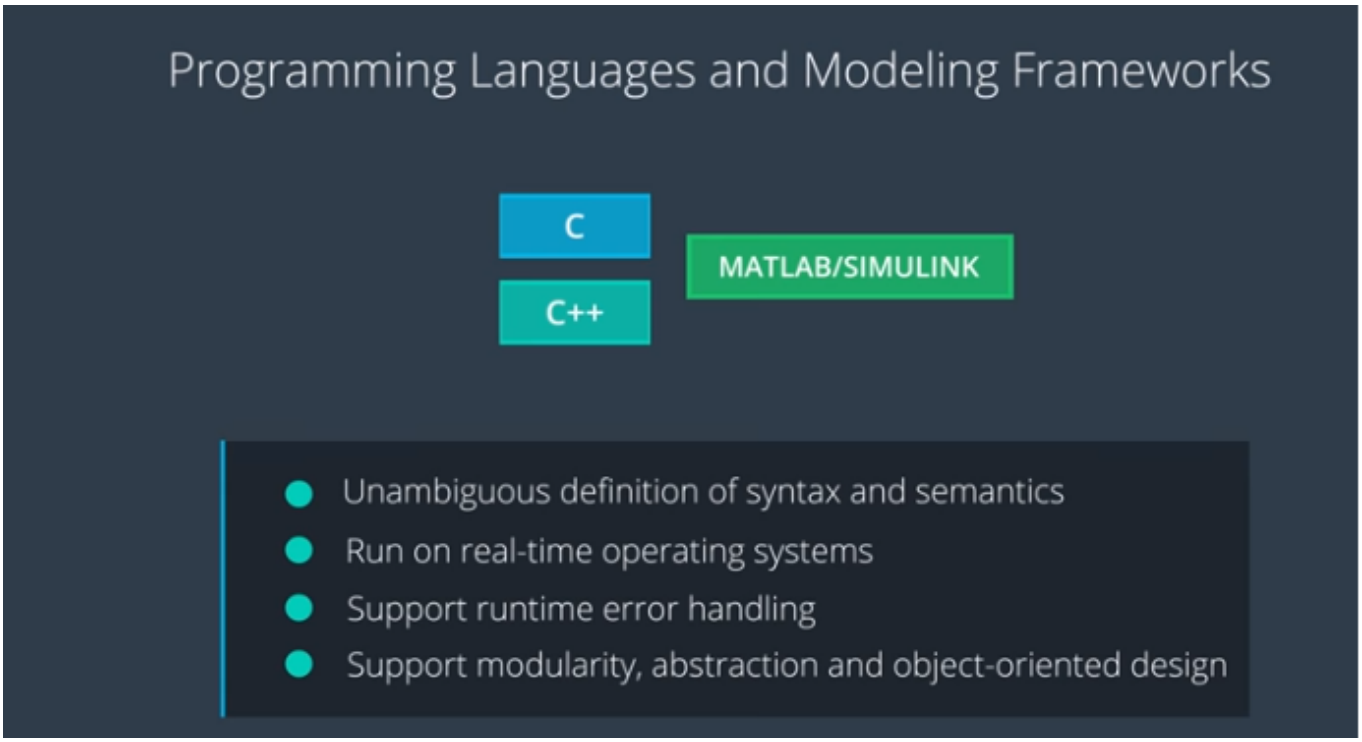
- specifying safety requirements
- designing an architecture and allocating the requirements to the architecture
- software testing
- software integration



Architectural Design vs Unit Design

- The software architectural design is a higher level view of software components.
- A unit is a smaller part of a software architecture.
- A unit could be a software driver to read raw data from a camera sensor.

The programming language



- A strongpoint for C++ is the ability to write high-speed software with many input-output operations.
- On the other hand, C++ will allow you to store a floating-point number in a boolean variable.
- C++ does not provide much in terms of run-time error checking.

MISRA

- The MISRA C++ standard discusses a subset of C++ that is appropriate to safety critical applications.
- The standard contains a set of rules for how to use the C++ language in automotive applications.

Software Tools

Ensure MISRA compliance for the following tools

- Compilers are one example of software tools.
- Version control software.
- Testing tools.
- Graphical modelling tools that automatically generate code

Software Tool Confidence Level

- The functional safety standard requires that you qualify software tools to make sure they are appropriate for safety critical applications
- ISO 26262 describes a metric for measuring in the tools. The metric is called tool confidence level or TCL.
    - Tool Impact (TI) - Whether the tool itself could malfunction and violate a safety goal
    - Tool Error Detection Capability (TD) - If the tool malfunctions, is the malfunction detected or stopped
- Use the TI and TD metrics to calculate a tool's confidence level
- Software blocks with higher ASIL require TCL1, which is the highest confidence.
- TCL3 is the lowest confidence rating.
- Lower confidence tools with TCL2 and 3 ratings need to be qualified.
- Qualifying involves running the tool through rigorous testing to prove that it does not cause any errors.

Mechanism for ensuring freedom from spatial interference

- Memory protection unit (MPU)
- Dual storage of relevant data
- Redundancy checks such as CRC to make sure data does not inadvertently change.
- Micro-controllers with memory error detection and correction capabilities
- Operating systems that allow software units to have their own virtual memory space

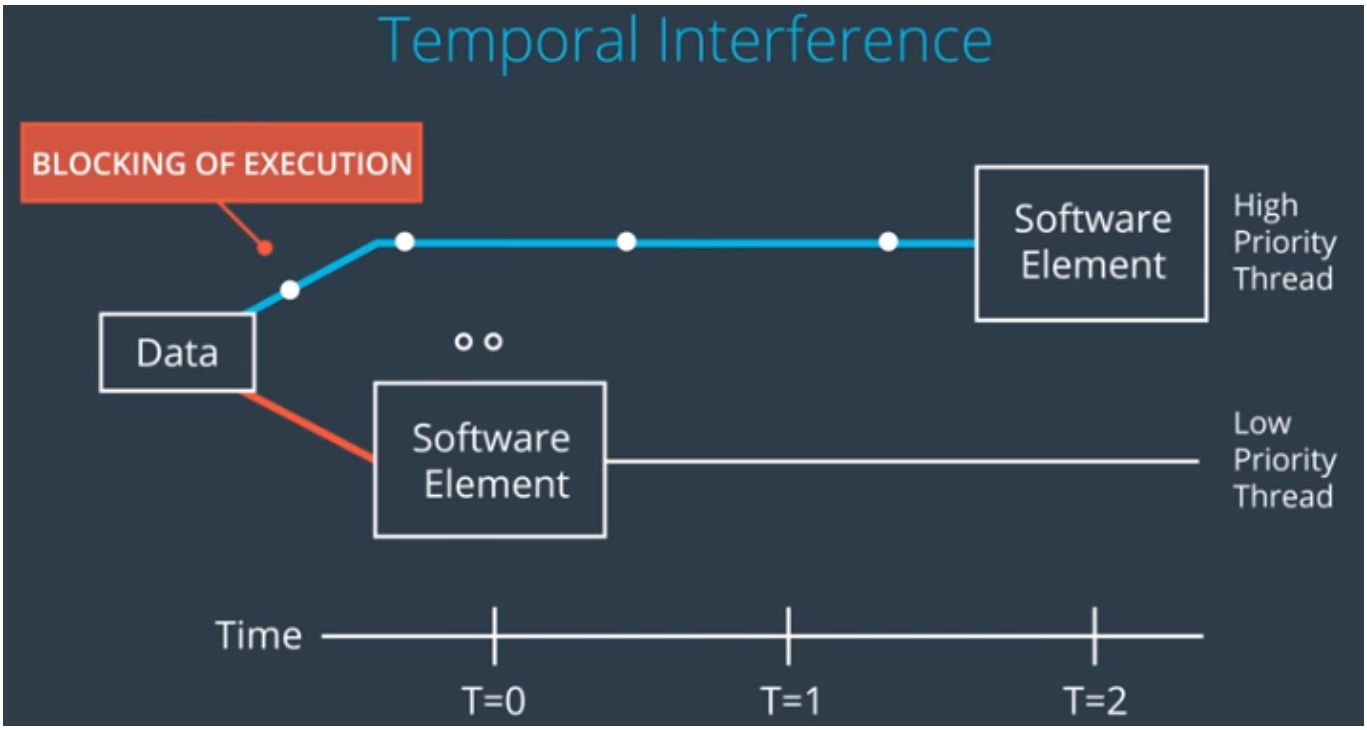
MPU

MPU is a prevention method because it stops elements from accessing memory to which they should not have access. An MPU can be programmed to set up the proper read, write and execute permissions between software elements.

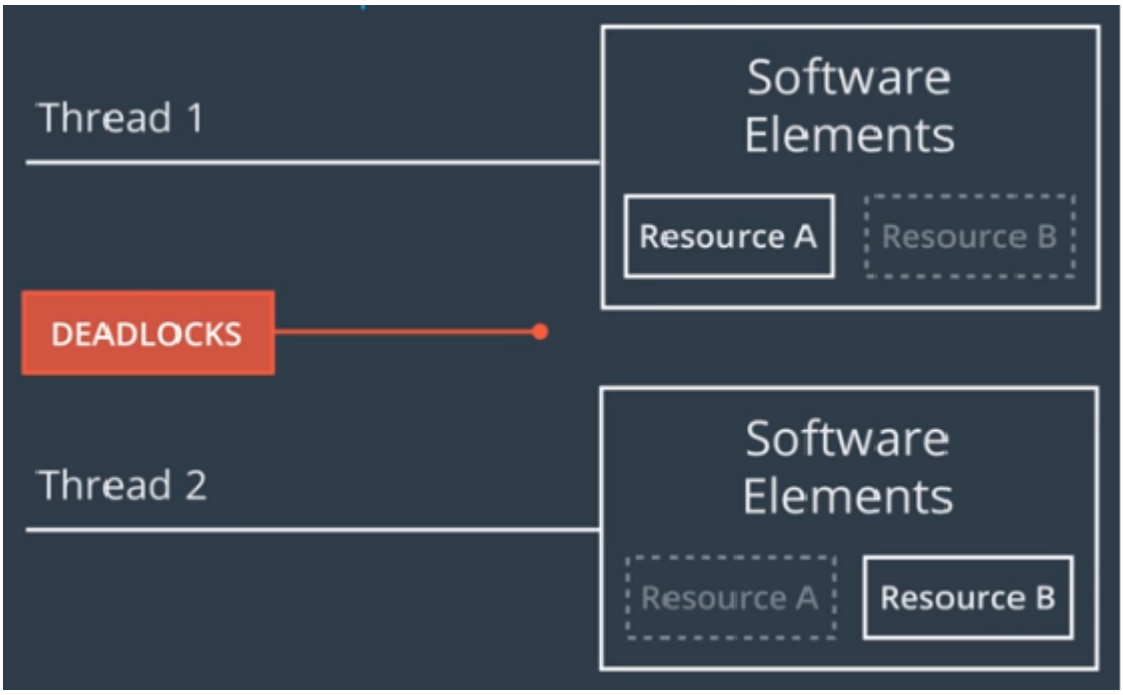
Dual Storage

Dual storage of relevant data like with a 2's complement is a detection method.

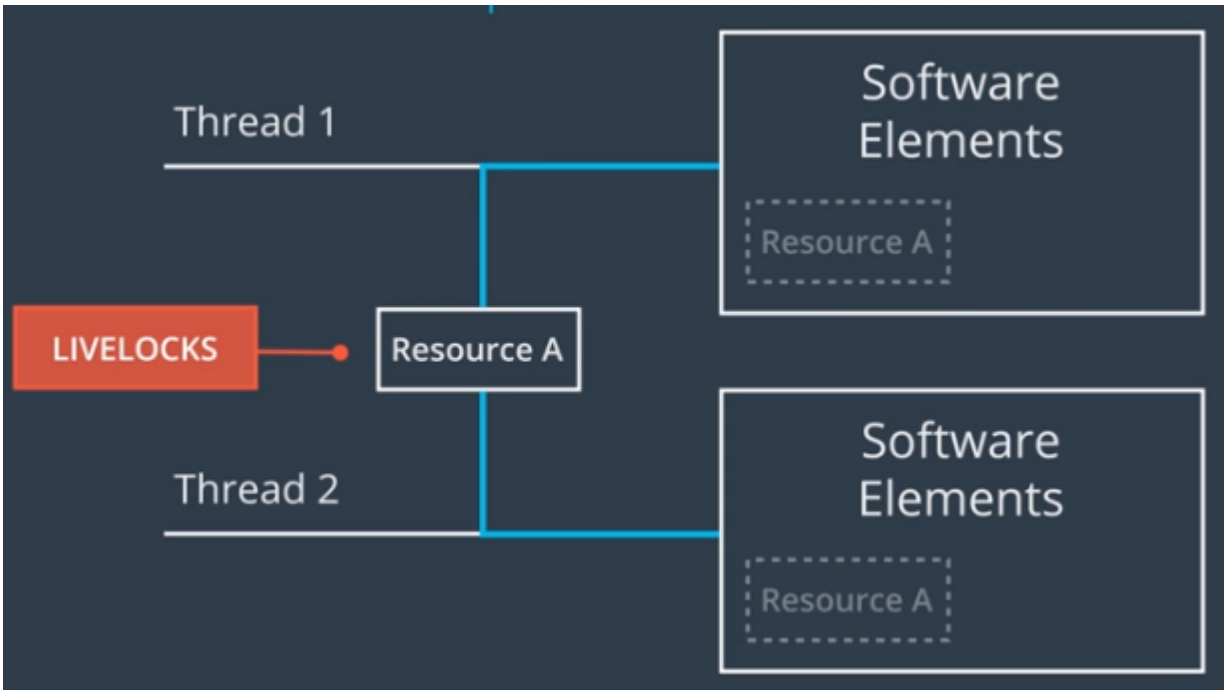
Mechanisms for Freedom from Temporal Interference



Deadlocks Vs Livelocks



Thread 1 needs Resource B and have Resource A. But Thread 2 keeps interrupting Thread 1 to grab Resource B. Same way Thread 2 needs Resource A and have Resource B is called deadlocks



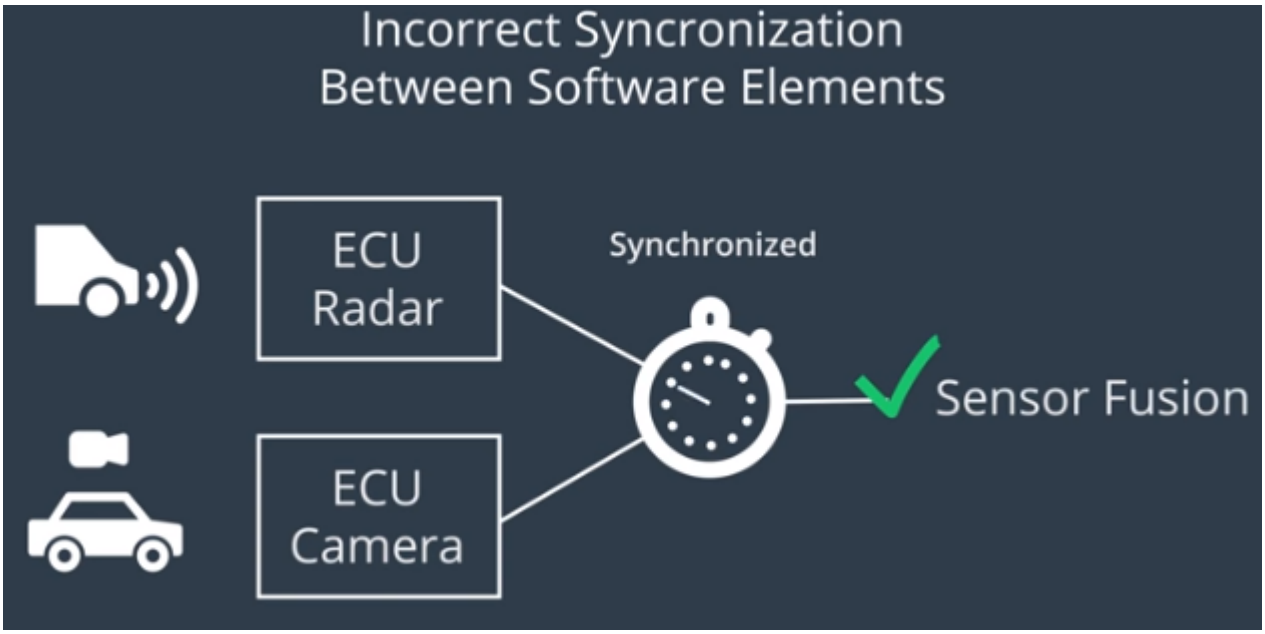
Both the threads have courtesy to go the other thread to go first.

Priority Ceiling

For addressing deadlocks, disabling OS interrupts that would stop process preemption, is inefficient and could compromise the overall response time and system latency. An alternative is a feature that is provided by a Real Time Operating System (RTOS) is a *priority ceiling*.

Synchronization

Clock synchronization between two Electronic Control Unit (ECU)



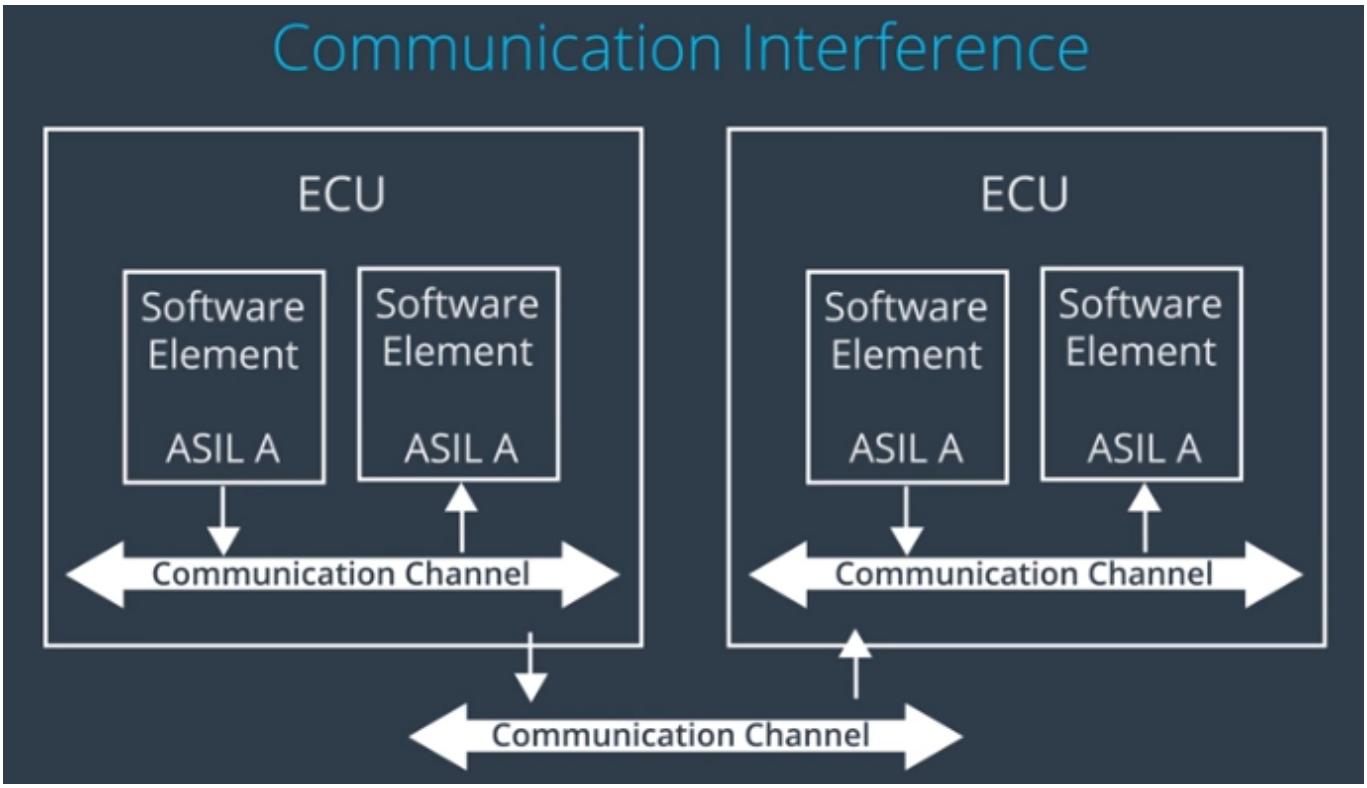
Mechanism to prevent

- Cyclic execution scheduling
- Fixed priority based scheduling
- Time triggered scheduling
- Monitoring of processor execution time
- Program sequence monitoring
- Arrival rate monitoring.

3 Safety Mechanism

- Alive supervision - Number of time the element got executed.
- Deadline monitoring - How long it takes to execute the software.
- Control flow monitoring - Software executed in the correct order.

Mechanisms for Ensuring Freedom from Communication Interference



There are many causes for communication faults. These causes would be analyzed in a software safety analysis or sometimes in a technical safety analysis:

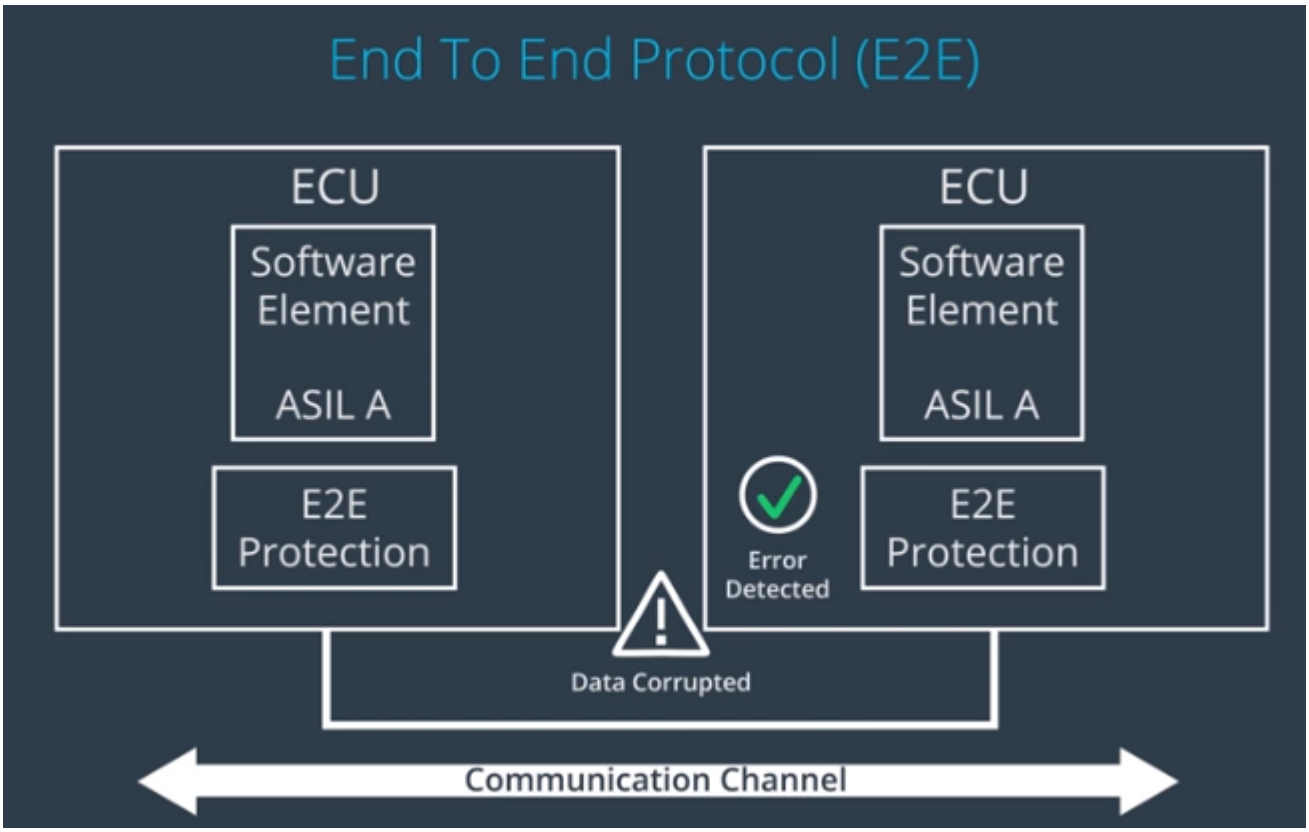
- Repetition of information
- Loss of information
- Delay of information
- Insertion of information
- Masquerade or incorrect addressing of information
- Incorrect sequence of information
- Corruption of information

Mechanisms

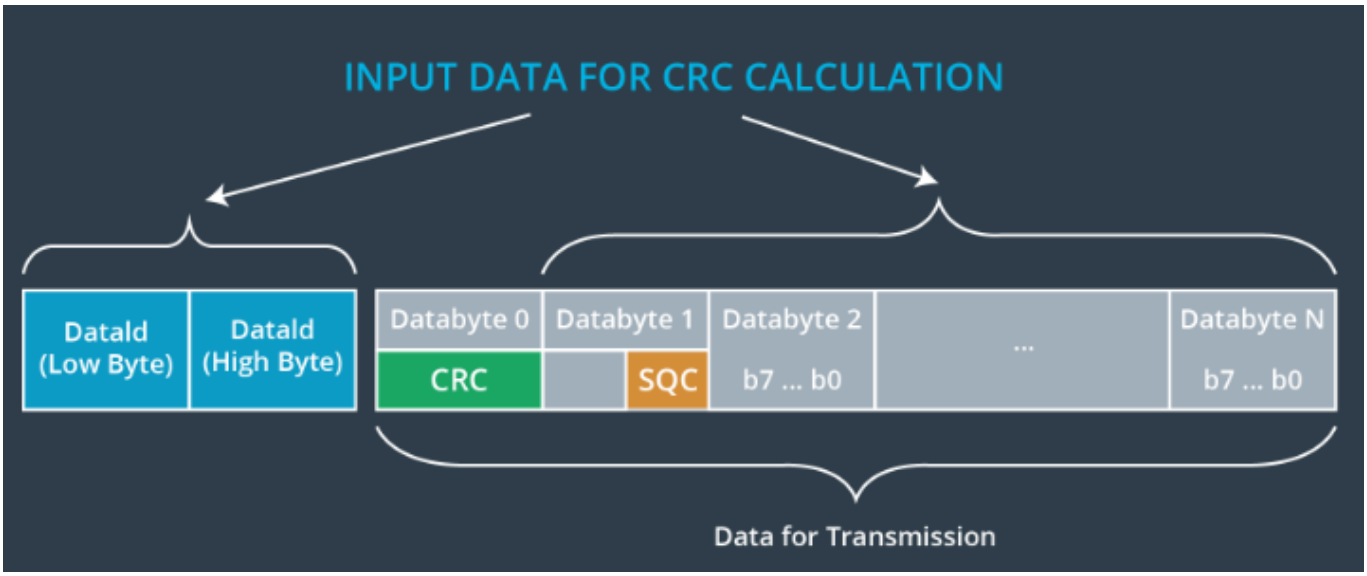
- Loopback of information
- Acknowledgement of information
- Appropriate configuration of I/O pins
- Bus arbitration by priority
- E2E protocol

E2E Mechanism

The technical safety requirement could be refined into a software safety requirement that the data shall be protected by an End2End mechanism



E2E Protocol



- The mechanism involves adding two extra data-bytes called a CRC (Cyclical Redundancy Check) and an SQC (Sequence Counter) when transmitting data.
- To calculate the CRC, run a mathematical formula on the data to be transmitted.
- Attach the CRC result to the data prior to transmission.
- When the data is received, the mathematical formula is run on the data set again.
- The CRC attached to the data and the CRC calculated on the receiving end should be the same; otherwise, data data has probably been corrupted in transmission.
- The SQC is just a counter that gets sent along with the data.
- That way the receiver can make sure that messages haven't been lost.

Software Partitioning and Safety Monitoring

- Safety monitoring and software partitions are software mechanisms commonly solved with design patterns.
- Software partition - hardware feature called MPU along with dual data storage.
