End-to-End Protection Training

Training Goal and Objectives

Goal: The student should understand the basics of end-to-end protection as it applies to ASIL rated CAN signals.

Objectives:

When the student is done with this course they will be able to:

- **Describe why end-to-end protection is needed**
- **Describe details of the current counter/checksum implementation, limitations, and where to get the requirements**
- **Describe AUTOSAR E2E including the profiles**
- **Describe the E2E approval process**

Functional Safety Motivations

- ***"Freedom from Interference"*** **is the primary motivation for End to End Protection**
  - **ISO 26262 Part 1 defines "Freedom from Interference" as the "absence of *cascading failures* between two or more *elements* that could lead to the violation of a *safety* requirement"**
- **In order to separate elements of different ASIL ratings, "Partitioning" of elements is used**
  - **Partitioning is the "separation of functions or *elements* to achieve a design"**
  - **"Partitioning can be used for *fault* containment to avoid *cascading failures*. To achieve *freedom from interference* between partitioned design *elements*, additional non-functional requirements can be introduced"**

Functional Safety Motivations

- **Partitions can exist on separate ECUs, and the ASIL-rated software components on these ECUs will need to exchange information**
  - **This exchange of information would be done through a network like CAN using signals**
- **ISO 26262 Part 6 lists out causes or effects of faults that can occur with exchange of information and may violate Freedom from Interference:**
  - **Repetition of information**
  - **Loss of information**
  - **Delay of information**
  - **Insertion of information**

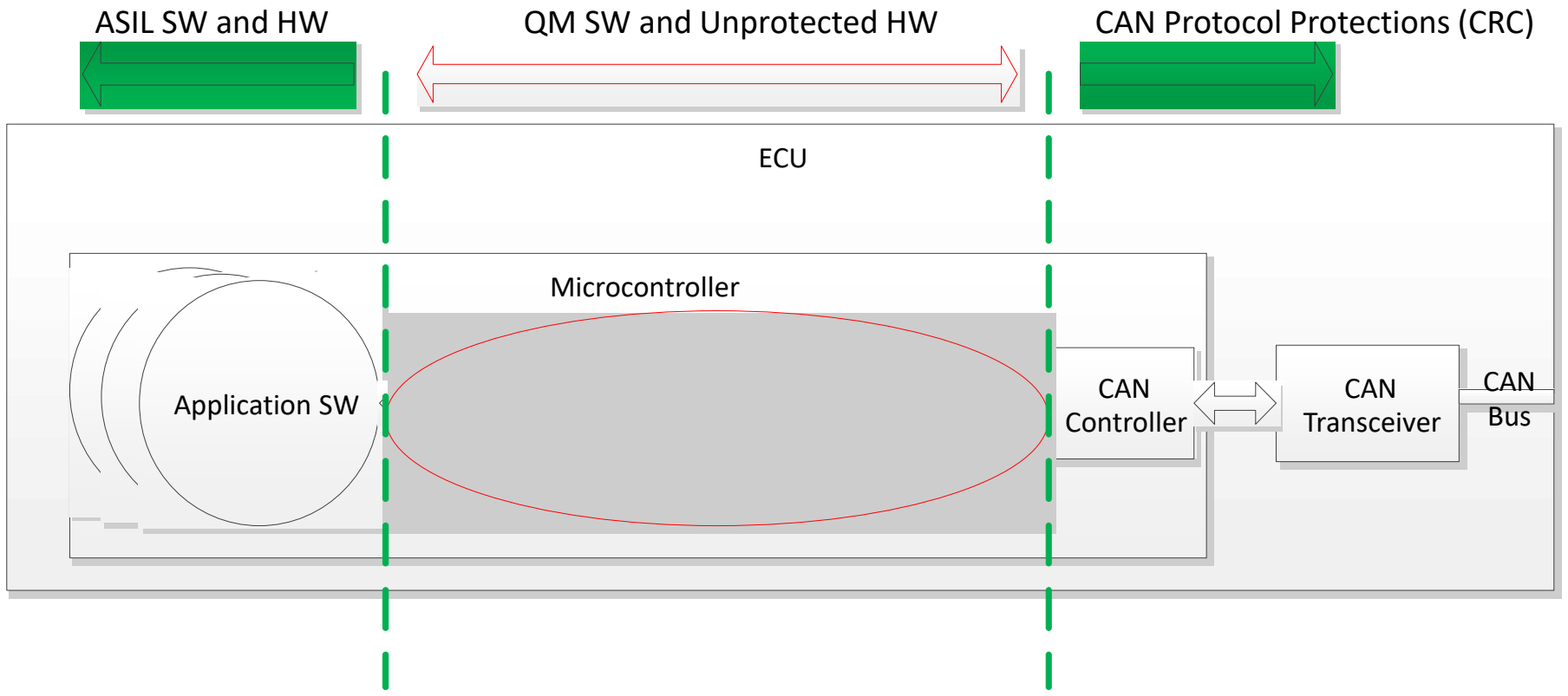
Functional Safety Motivations

- **ISO 26262-6 causes or effects of faults, continued:**
  - **Masquerade or incorrect addressing of information**
  - **Incorrect sequence of information**
  - **Corruption of information**
  - **Asymmetric information sent from a sender to multiple receivers**
  - **Information from a sender received by only a subset of the receivers**
  - **Blocking access to a communication channel**

- **“End-to-End Protection” or “E2E” is the term used for the software measures used to ensure integrity of exchanged information and prevent or detect faults identified in ISO 26262 Part 6**

Need for End-to-End Protection

- **E2E ensures the integrity of CAN messages and signals as they pass from the CAN bus, through QM software and unprotected hardware, and finally to the ASIL-rated application software.**
- **E2E is required for all ASILs (A through D)**



ASIL Rated for What?

- The statement that a CAN Message or Signal alone needs to be ASIL-rated alone lacks key information

- What critical failure mode of the CAN Message or Signal will violate an ASIL- rated Safety Goal?

- For example, a Feature using Vehicle Speed to detect whether the vehicle is stationary will have much different requirements than a Feature that is using Vehicle Speed to detect a vehicle stability event at high speed.
  - They may have very different accuracy requirements
  - One may only care about under reporting, the other about over reporting vehicle speed

Protect Against Systematic and Random Failures

In order for a CAN message to be ASIL-rated, we must protect against the two main types of failures:

1. **Systematic Failure – Most likely caused by a failure of the design or implementation of the hardware of software**

2. **Random Hardware Failure – Could be caused by EMC or cosmic rays**

Systematic Failure

A systematic failure…

is a failure in the
- requirements
- design
- implementation of HW/SW
- manufacture
- maintenance

has a deterministic relation to a certain cause

has a predictable behavior

May sometimes appear “pseudo-random” (unpredictable) due to complex triggering conditions

e.g., system has to be in dedicated operating state

e.g., race condition – sequence to memory access of competing processes leads to different results

Note: System failures caused by software are always systematic failures!

Random Hardware Failure

A random hardware failure…

is an unpredictable failure during the lifetime of a hardware element following a probability distribution

can be quantified with reasonable accuracy

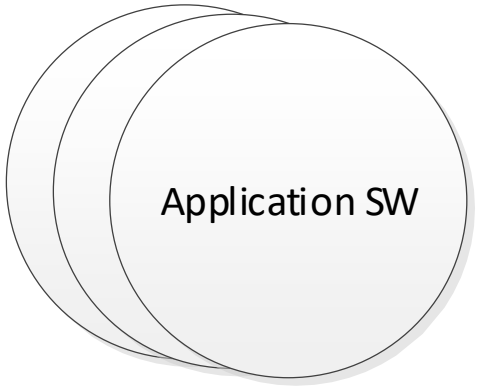
Examples:

Capacitor:	short, open, value changed
Fuse:	fails to open, slow to open, premature open
Memory:	data bit loss, short, open, slow transfer of data
Resistor failure:	short, open, value changed
Relay:	fails to trip, spurious trip, short
Transistor failure:	short, output low, parameter change, open, output high

Note: the kind of the random hardware failure may have different effects on system level

Application Software

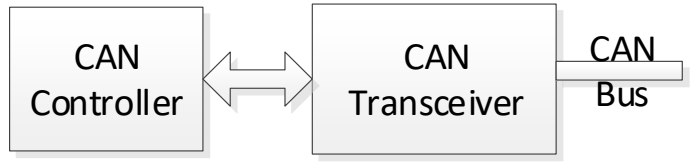
- **In order to meet the ASIL rated requirements for a Feature, the Application Software and Safety Architecture must be ASIL rated for the Feature.**
- **The Application Software will act as the endpoint sender or receiver of protected data**



CAN Controller and CAN Transceiver Protection

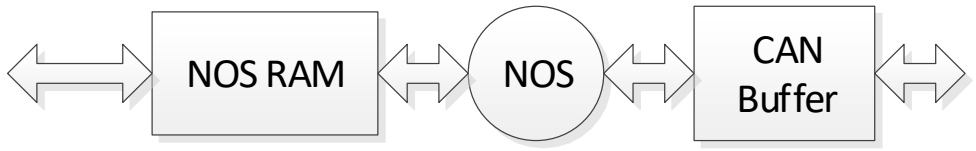
- **Each frame of the CAN Protocol contains a 15-bit CRC which is added by the sending CAN Controller and processed by the receiving CAN Controller of the frame.**

- **The potential for a CAN frame to become corrupted while on the CAN Bus and not be detected is extremely low.**



Network Operating System

- **Ford purchases our Network Operating System from Vector, called FNOS**



- **FNOS is implemented as QM**

- **Although it is possible for NOS RAM and CAN Buffer to have memory protection like ECC, most ECUs on the vehicle don't have this implemented. This leaves the possibility for potential random failures caused by the environment including strong EMC, and cosmic rays.**

- **End-to-End Protection closes these gaps**

E2E Requirements

- At Ford, the current requirements prescribe adding a Counter and Checksum to a message or group of messages to implement E2E.
  - FEDE requirements RQT-060900-006149 and RQT-060900-704589

- The Counter ensures freshness of data
  - Protects against replay, stuck, or lost messages

- The Checksum ensures correct data
  - Protects against corruption of the data

Counter

- The counter consists of 4 bits (hex 0 to F)
- Increments at a time basis (tick rate) of 10ms, independent of message periodicity
- Rolls over back to 0 every 160ms
- Must be an equation based on real time and therefore must be derived from the ECU clock
- Due to tick rate and counter size, supports message periodicity of 100ms or faster only

Checksum

- 8-bit Checksum (hex 0 to FF)
- Algorithm: Sum with One's Complement
- Inputs: The data and the counter
  - It should be noted that the signal value must be the exact value that is being sent on CAN. If there is any scaling done to the signal, the post-scaling signal must be used in the calculation.
- If the signal exceeds 8 bits in length, the signal must be broken into two low byte and high byte
  - The low byte is the first 8 (least significant) bits
  - The high byte is the remaining bits after the 8th
- If the checksum calculation exceeds this size, exceeding bits are to be ignored

Checksum Example

Signal 1 = 507

Signal 2 = 15

Counter = 1

Signal 1: 111111011b

Signal 2: 1111b

Counter: 1b

Step 1 – Break Up Data that Exceeds 8 Bits

Sig1:

Segment 1 (take first 8): 11111011b

Segment 2 (take next 8): 00000001b

Sig2: 00001111b (no change needed)

Step 2 – Sum

Add lowest order (right most) bits of signal(s) and counter

Sig 1 Hi Byte		1	1	1	1	1	0	1	1 b	
Sig 1 Low Byte		0	0	0	0	0	0	0	1 b	
Sig 2		0	0	0	0	1	1	1	1 b	
Counter		0	0	0	0	0	0	0	1 b	+
Checksum	1	0	0	0	0	1	1	0	0 b	

Step 3 – Clear Excess

Clear off any bit(s) after 8th

Checksum	0	0	0	0	1	1	0	0 b

Step 4 – One’s Complement

	0	0	0	0	1	1	0	0 b	
									~
Checksum =	1	1	1	1	0	0	1	1 b	

- Convert back to Decimal = 243

Move to AUTOSAR E2E

Although the current Counter and Checksum ensures integrity, it has some limitations:

- **The hamming distance (minimum number of errors that could cause the checksum to be correct with corrupted data) may not be suitable for all applications**
- **The 4-bit counter with 10ms tick prevents use in for signal rates slower than 100 ms**
- **Additional commodities have new Functional Safety requirements which may require longer data or slower transmission rates (Body, Electrical, ADAS, etc.)**
- **A working group within Ford has been tasked with defining E2E requirements which utilize AUTOSAR End-to-End Protection**

Why AUTOSAR?

- AUTOSAR (*AUT*omotive *O*pen *S*ystem *AR*chitecture) is a worldwide development partnership of vehicle manufacturers, suppliers, service providers, and companies from the automotive electronics, semiconductor and software industry
  - Ford is one of nine “Core Partners”
- The primary target of AUTOSAR is automotive software applications in Electronic Control Units (ECUs)
- AUTOSAR defines a software platform including End-to-End Protection algorithms and implementation
  - This provides something closer to an “industry standard” for E2E

AUTOSAR E2E Profiles

- AUTOSAR E2E Protection is defined in what are called “Profiles”
  - Each profile has a similar structure but varies in design and level of protection
- All Profiles utilize CRC (Cyclic Redundancy Check), Counter, and Data ID mechanisms for protection
  - Cyclic Redundancy Check: an error-detecting code based on polynomial division
    - » Effectively replaces the Checksum, but is applied to a byte string containing signals and provides a greater hamming distance
    - » Relatively lightweight (does not require hardware acceleration)
  - Counter is incremented upon every transmission
  - Data ID is an identifier for the protected string of signals

AUTOSAR E2E Profiles

- **The CRC, Counter, and Data ID mechanisms are used to detect faults or effects of faults identified by ISO 26262-6**

E2E Mechanism	Detected Communication Faults
Counter	Repetition, loss, insertion, incorrect sequence, blocking
Transmission on a regular basis and timeout monitoring using E2E Supervision[1]	Loss, delay, blocking
Data ID + CRC	Masquerade and incorrect addressing, insertion
CRC	Corruption, asymmetric information[2]

[1] Implementation by sender or receiver in addition to E2E mechanisms
[2] For a set of data protected by same CRC

AUTOSAR E2E Profiles

- **Different Profiles are meant to protect different data. This can be used for bus types which have longer maximum data lengths than classical CAN, like CAN-FD or Ethernet.**
- **The Ford working group has agreed on Profile 1A for protected data up to 8 bytes in length**
  - **Multiple E2E-protected packets can exist in a CAN or CAN-FD frame**
- **Profile 5 is proposed for longer data, but needs a use case defined**

Profile	Unit	1a	2	4	5	7
Length of DataID	Bit	16	8	32	16	32
Explicit Data ID	Bit	0	0	32	0	32
Length of Counter	Bit	4	4	16	8	32
Length of CRC	Bit	8	8	32	16	64
CRC Polynomial	Hex	1D	2F	F4ACFB13	1021	42F0E1EB A9EA3693

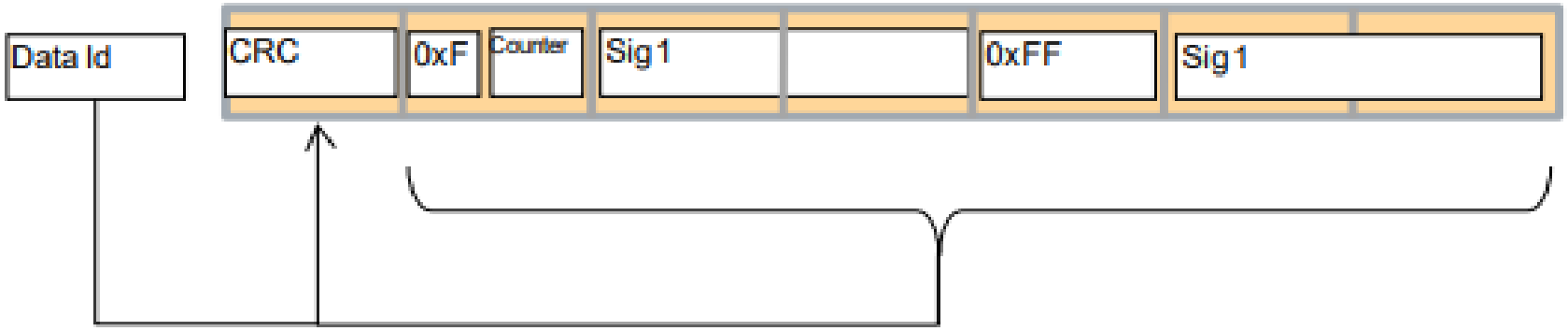
CAN					
LIN					
CAN-FD					
Ethernet					

Profile(s)	Hamming Distance (d) for length (n)				
1	n	1 -14	15 - 21	22 - 255	> 255
	d	5	4	3	2
2	n	1 - 127	> 127		
	d	4	2		
4	n	1 - 32768	>32768		
	d	4	2		
5 & 6	n	1 - 32760	>32760		
	d	4	2		

AUTOSAR E2E Implementation

- **E2E CRC and Counter are placed in network message frame. Depending on profile and variant, Data ID may or not be entirely present in frame**
- **The CRC is calculated over a byte string consisting of the Data ID, Counter, and functional signals (including unused space)**

For standard variant 1A:



CRC := CRC8 over (1) Data Id, (2) all serialized signal (including empty areas, excluding CRC byte itself)

AUTOSAR E2E Implementation

- In a traditional AUTOSAR Architecture, libraries are used for the needed E2E Profile(s) and CRC calculation. These libraries are invoked by the application through a Protection Wrapper or Transformer.

- For non-AUTOSAR architectures, different solutions will need to be developed which comply with Ford requirements.

- AUTOSAR E2E data can be captured in CMDB, GSDB, and DBC formats.
  - Protected signals along with CRC and Counter are structured in Signal Groups
- Eventually native AUTOSAR formats will be supported (ECU Extract ARXML).

Future Plans for AUTOSAR E2E

- **Update or creation of related FEDE requirements**
  - **RQT-003701-023579 exists but will be modified or replaced**
  - **Release Candidate requirements have been created and is planned to be done by end of July 2020**

- **Identification of which signals, ECUs, and vehicle programs will be affected**
  - **This is still under discussion, but proposal is first FNV3-based program**

Approval Process of End-to-End Protection

- **The Global Functional Safety Technical Governance Board currently approves all E2E uses**
  - **This process will continue to be used for AUTOSAR E2E**
- **This ensures that E2E is really required and that they are applied consistently across the vehicle**