Zephyr™ Project
Developer Summit
June 8-10, 2021 • @ZephyrIoT

Conduct FMEA in the safety analysis of Zephyr

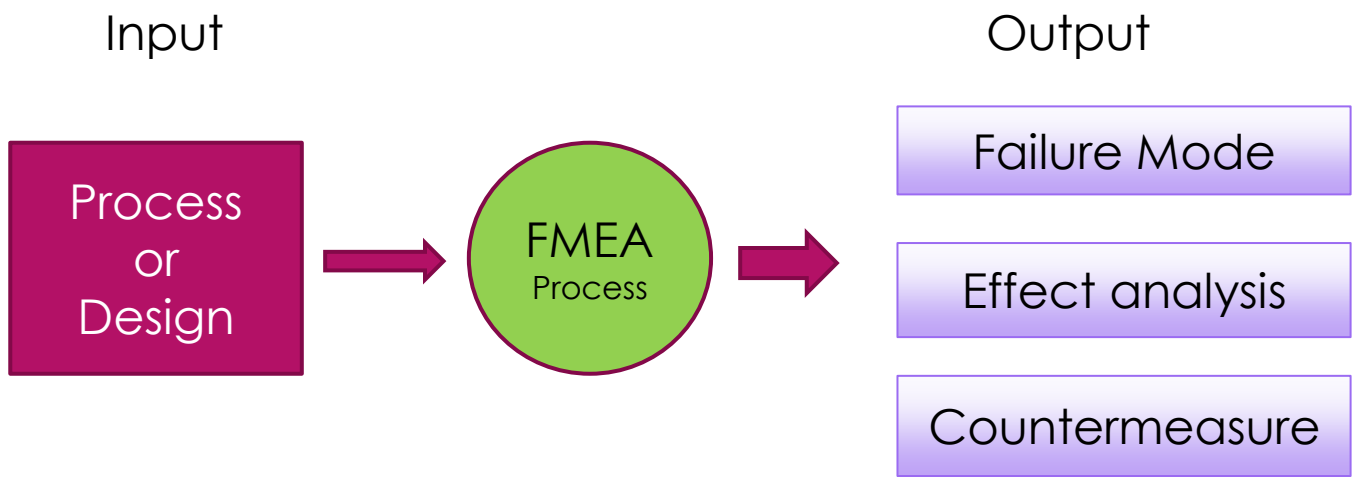
ENJIA MAI, INTEL

Outline

- What is FMEA?
- Why we apply the FMEA in zephyr project?
- How we do the SW FMEA?
- Show examples of SW FMEA.
- Conclusion
- Q & A

What is FMEA

- FMEA stands for Failure Mode and Effect Analysis
  - ***Failure modes***
  - ***Effects analysis***



- Definitions
  - FMEA is a step-by-step approach for identifying all possible failures in a design, a manufacturing or assembly process, or a product or service.
  - The latest official documents: AIAG / VDA FMEA handbook (2019)

What is FMEA

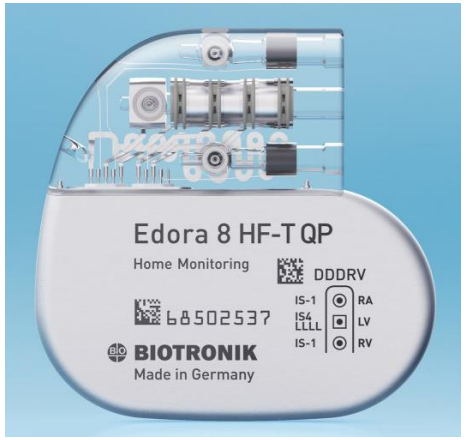
- FMEA characteristics
  - Identify potential failure modes in a system
  - Estimate the Risk of the potential failure
  - The result of analysis will be documented
  - An FMEA is often a qualitative analysis
  - Often as the first step of a system reliability study

- FMEA variations
  - PFMEA (process)
  - DFMEA (design). SW FMEA is one kind of it.

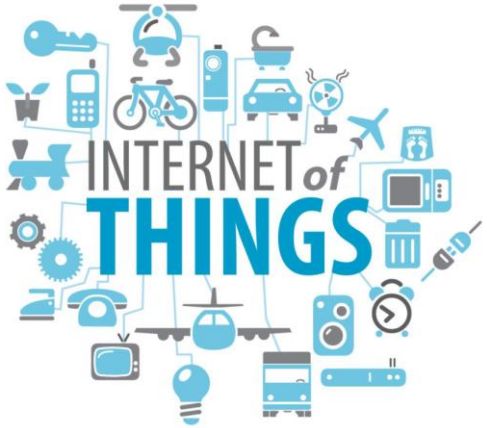
Why we apply the FMEA in zephyr project

Zephyr™ Project
Developer Summit

- The reason of conducting SW FMEA
  - Function safety certification (IEC-61508)
  - Quality of the zephyr project



- The advantage applying the SW FMEA
  - Check the design and documents
  - Find out the potential failure mode and prevention
  - Better adapting to develop safety-critical product

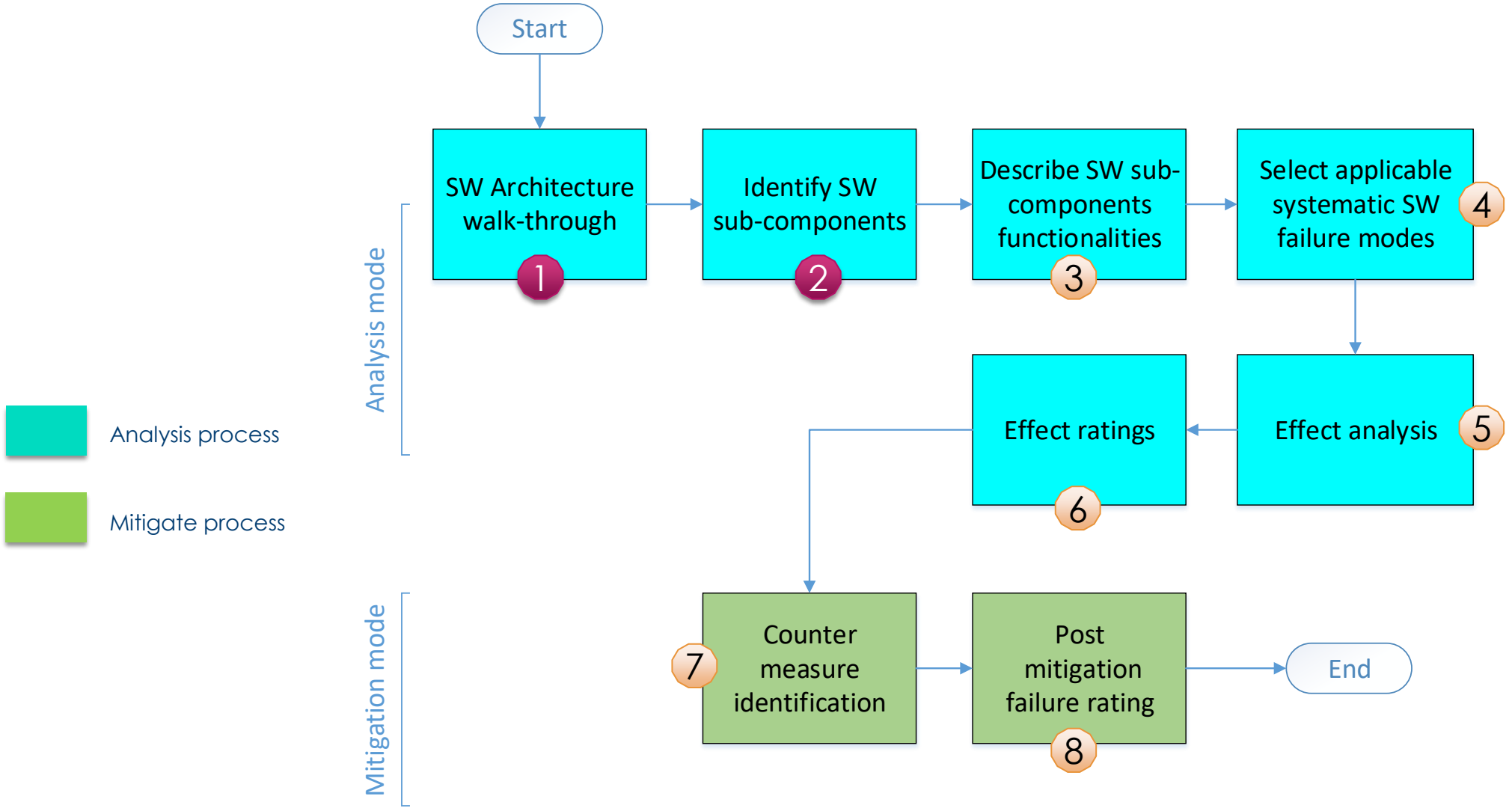


How we do the SW FMEA

- The generic method of implementing SW FMEA:
  - Take the interface between HW/SW or SW/SW module as the analysis object.
  - Enumerate the failure mode of the interface, analyze the impact of the failure mode on the software module or software requirements, especially the functional safety-related parts influences.
  - After clarifying the cause of the failure, control measures are applied to the cause of the failure.

- The target working products of SW FMEA
  - SW FMEA report
  - Countermeasures apply to code base
  - Update on requirement documents

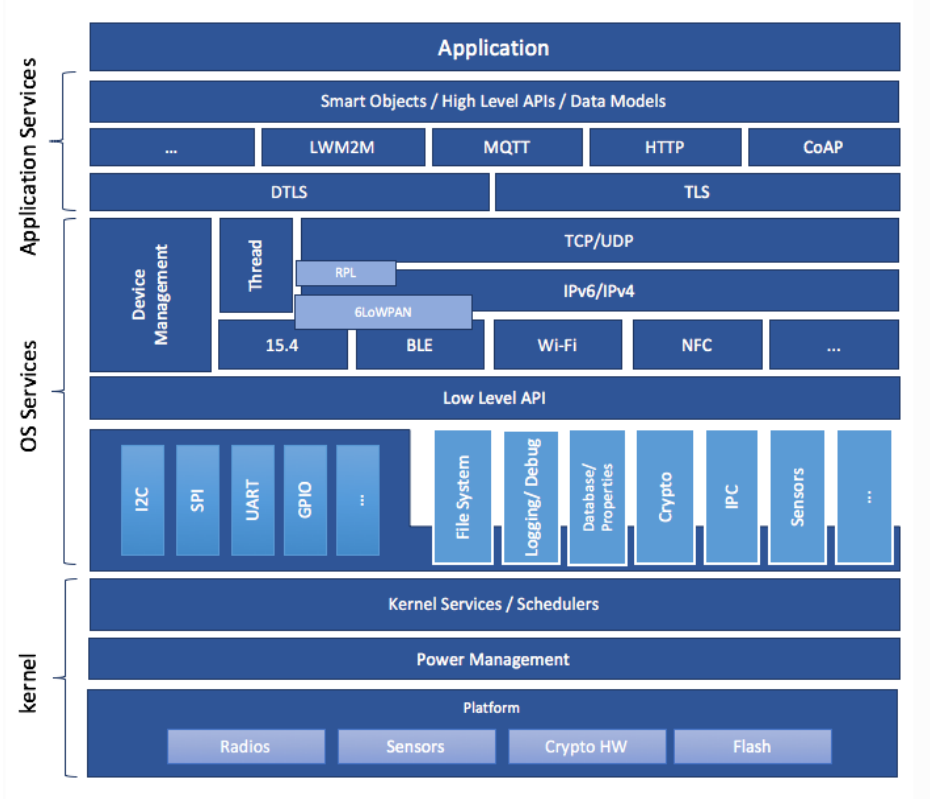
SW FMEA flow for single item



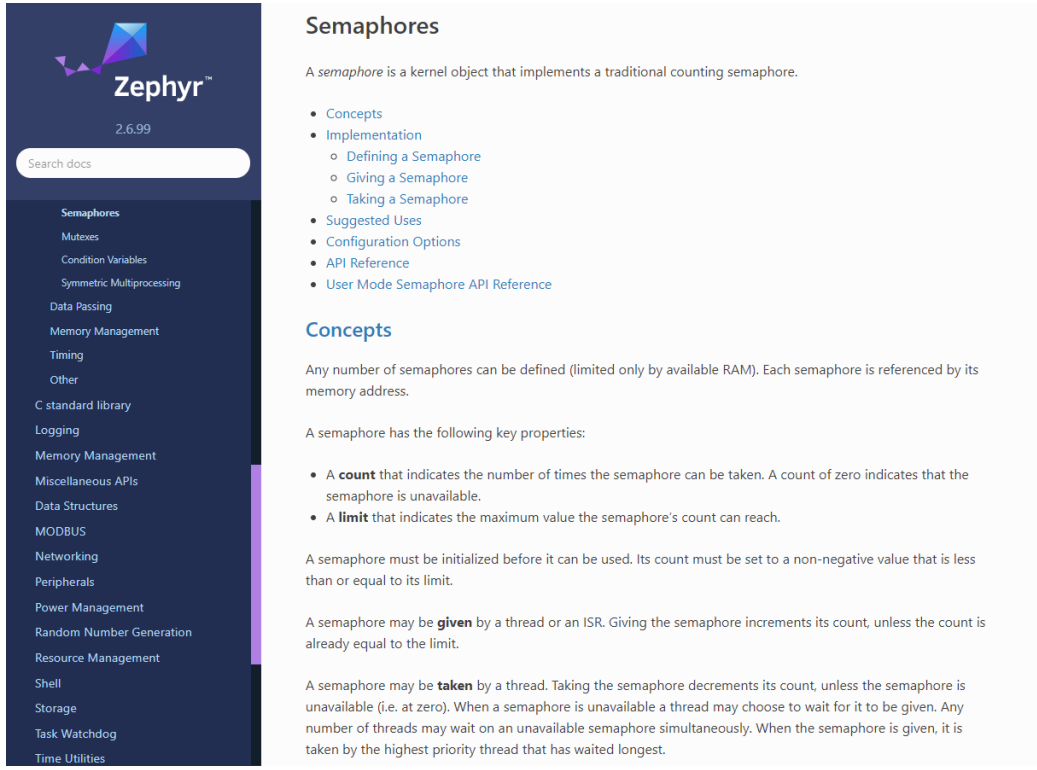
7

Example of applying SW FMEA

- Failure mode and effect analysis on the semaphore component
  - Step 1 & 2: SW architecture Walk through and identify SW component



SW architecture



SW Architecture Specification

Step 1 & 2

- SW Architecture spec and components analysis
  - https://docs.zephyrproject.org/latest/reference/kernel/synchronization/semaphores.html

Concepts

Any number of semaphores can be defined (limited only by available RAM). Each semaphore is referenced by its memory address.

A semaphore has the following key properties:

- A **count** that indicates the number of times the semaphore can be taken. A count of zero indicates that the semaphore is unavailable.
- A **limit** that indicates the maximum value the semaphore's count can reach.

A semaphore must be initialized before it can be used. Its count must be set to a non-negative value that is less than or equal to its limit.

A semaphore may be **given** by a thread or an ISR. Giving the semaphore increments its count, unless the count is already equal to the limit.

A semaphore may be **taken** by a thread. Taking the semaphore decrements its count, unless the semaphore is unavailable (i.e. at zero). When a semaphore is unavailable a thread may choose to wait for it to be given. Any number of threads may wait on an unavailable semaphore simultaneously. When the semaphore is given, it is taken by the highest priority thread that has waited longest.

Note

The kernel does allow an ISR to take a semaphore, however the ISR must not attempt to wait if the semaphore is unavailable.

Implementation

Defining a Semaphore

Step 3 & 4

- Step3: Descript SW subcomponent functionalities
  - Refer to the SAS then descript the functionalities.

  - **Functional description**
    - The kernel does allow an ISR to take a semaphore, however the ISR must not attempt to wait if the semaphore is unavailable. Refer to function k_sem_take()

- Step 4: Select applicable systematic SW failure mode
  - Base on single functionality, to analyze the potential failure mode of this functionality. Then decide its failure mode.

  - **Failure mode:**
    - Delay interrupt request
      - Take the semaphore in interrupt cannot wait for too long, or even do a scheduling

• Step 5: Effect analysis

- Effect analysis
  - Analyze what the effects will happen, include the dependency components , while this failure mode happens.
  - **Effect of failure:**
    - Wrong Data
  - **Cause of failure:**
    - Incorrect usage
  - **Control detection:**
    - Add an assert to check when it is in ISR, we cannot give waiting time as a parameter.

Step 6: Effect Rate

- Effect Rate
  - RPN (Risk Priority Number) show the probability of potential risk.
  - RPN = Severity x Occurrence x Detectability

    = 7 x 5 x 3 = 105

Severity: how serious their consequences

Value	Description	Definition
1	None	No effect
2	Very Minor	Function operable with minimal interference
3	Minor	Function operable with some degradation of performance
4	Very Low	Function operable with significant degradation of performance
5	Low	Function inoperable without damage
6	Moderate	Function inoperable with minor damage
7	High	Function inoperable with equipment damage
8	Very High	Function inoperable with destructive failure without compromising safety
9	Critical	Very high severity ranking when a potential failure mode affects function operation with warning
10	Catastrophic	Very high severity ranking when a potential failure mode affects fuction operation without warning

Detectability: how easily they can be detected

Value	Description	Definition
1	AlmostCertain	Design control will detect potential cause/mechanism and subsequent failure mode
2	VeryHigh	Very high chance the design control will detect potential cause/mechanism and subsequent failure mode
3	High	High chance the design control will detect potential cause/mechanism and subsequent failure mode
4	ModeratelyHigh	Moderately High chance the design control will detect potential cause/mechanism and subsequent failure mode
5	Moderate	Moderate chance the design control will detect potential cause/mechanism and subsequent failure mode
6	Low	Low chance the design control will detect potential cause/mechanism and subsequent failure mode
7	VeryLow	Very low chance the design control will detect potential cause/mechanism and subsequent failure mode
8	Remote	Remote chance the design control will detect potential cause/mechanism and subsequent failure mode
9	VeryRemote	Very remote chance the design control will detect potential cause/mechanism and subsequent failure mode
10	AbsoluteUncertainty	Design control cannot detect potential cause/mechanism and subsequent failure mode

Occurrence: how frequently they occur

Value	Description	Definition
1	Very Improbable	So unlikely, it can be assumed occurrence may not be experienced
2	Improbable	So unlikely, it can be assumed occurrence may not be experienced (fortified)
3	Very Remote	Unlikely but possible to occur in the usage of the SW
4	Remote	Unlikely but possible to occur in the usage of the SW (fortified)
5	Occasional	Likely to occur some time in the usage of the SW
6	Moderate Occasional	Likely to occur some time in the usage of the SW (fortified)
7	Probable	Will occur several times in the usage of the SW
8	Very Probable	Will occur several times in the usage of the SW (fortified)
9	Frequent	Likely to occur frequently
10	Very Frequent	Likely to occur frequently (fortified)

Step 7: Countermeasure identification

- Countermeasure identification
  - Design the countermeasure which could solve or mitigate the risk.
  - **Control prevention:**
    - Taking semaphore operation cannot wait in ISR
  - **Countermeasure:**
    - Add an __ASSERT to catch the incorrect usage of taking semaphore.
      (Taking semaphore in interrupt context with a delay)

```
int z_impl_k_sem_take(struct k_sem *sem, k_timeout_t timeout)
{
        int ret = 0;

        __ASSERT(((arch_is_in_isr() == false) ||
                  K_TIMEOUT_EQ(timeout, K_NO_WAIT)), "");

        k_spinlock_key_t key = k_spin_lock(&lock);
        sys_trace_semaphore_take(sem);

        if (likely(sem->count > 0U)) {
                sem->count--;
                k_spin_unlock(&lock, key);
                ret = 0;
                goto out;
        }

        if (K_TIMEOUT_EQ(timeout, K_NO_WAIT)) {
                k_spin_unlock(&lock, key);
                ret = -EBUSY;
                goto out;
        }

        ret = z_pend_curr(&lock, key, &sem->wait_q, timeout);

out:
        sys_trace_end_call(SYS_TRACE_ID_SEMA_TAKE);
        return ret;
}
```

• Step 8 : Post mitigation failure rate

- Post mitigation failure rate
  - Rate the RPN again after applying the mitigation(countermeasures)
  - RPN = Severity x Occurrence x Detectability
    = 7 x 5 x 1 = 35

The Risk Priority Number should be lower than acceptable value

Maturity	RPN threshold			
	SIL 1	SIL 2		SIL 3
	ASIL A	ASIL B	ASIL C	ASIL D
New / immature SW	200 150	150 100	100 80	80 60

Example SW FMEA worksheet

Zephyr™ Project
Developer Summit

Analysis															
Id	SW Unit/ SubUnit	Function ality	Funct. Descr.	Failure Mode	FM Description	Effect of Failure	Severity	Causes of Failure	Control Prevention	Occurrence	Control Detection	Detectability	RPN	Recommended Actions	Responsibility / Action Owner	Target Completion Date
SW FMEA79	Semahpore	Kernel	The kernel does allow an ISR to take a semaphore, however the ISR must not attempt to wait if the semaphore is unavailable. Mapping to function k_sem_take()	Delayed interrupt request	Take the semphore in interrupt cannot wait for too long, or even do a scheduling.	Wrong Data	7 : High	Other interrupt would be delayed or missing.	Make taking semaphore operation cannot wait in ISR	5 : Occasional	Add an assert to check when it is in ISR, we cannot give waitig time as a parameter.	3 : High	105	Add an assert to detect and check.	Andy Ross	Mar/20/2020

After mitigation									
Actions Taken	Effective Date	Severity - Post	Occurrence Post	Detectability - Post	RPN - Post	Maturity Level	Maturity Level Rationale	Comment	Uuid
Add an assert to detect and check, once it happens, trigger an exception .	Mar/20/2020	7 : High	5 : Occasional	1 : AlmostCertain	35	Proposed	Use a assert to detect it and trigger an exception means system will be halted. Is there any alternative ?	commit 7832738ae985a63febb8f82e7c4e34824f48486e	{b71045f2-db38-4e6c-88a2-1f23979541ef}

Analysis process

Mitigate process

Conclusion

- Conducting FMEA is also aim to:
  - Figure out some existing safety mechanisms
  - Figure out existing countermeasures in current code base
  - Address some potential failure mode

- Let's work together to make zephyr project better
  - The impacts and challenges

Zephyr™ Project
Developer Summit

Thank you for your listening!

Q & A


Zephyr™ Project
Developer Summit
June 8-10, 2021 ▪ @ZephyrIoT