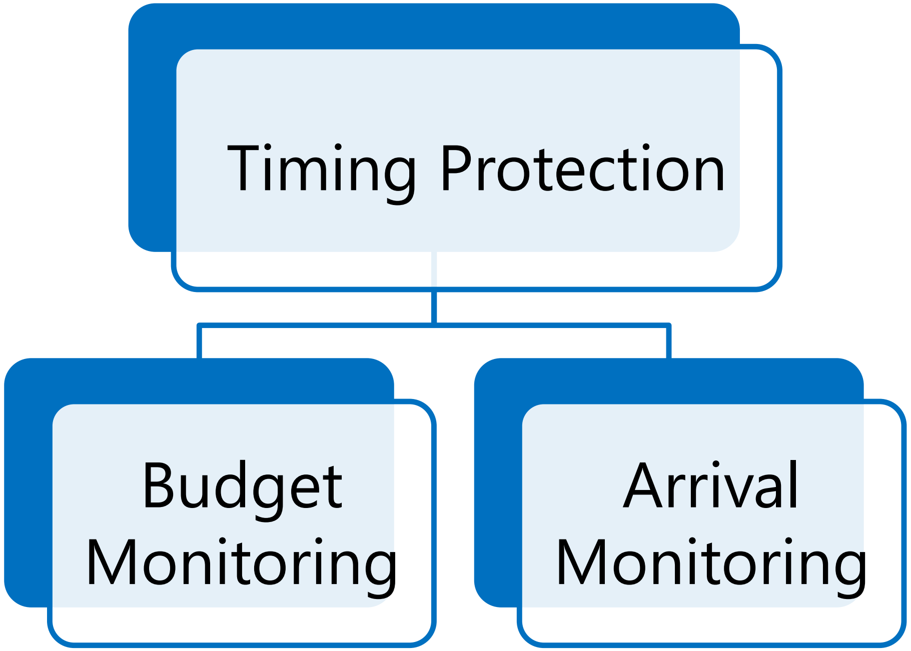
AUTOSAR OS SC2 – Overview and Use Case

KPIT

Scope of the session

- AUTOSAR OS Timing Protection Features
- Sample Use Case for Timing Protection Functionality
- Scalability Class 2(SC2) of AUTOSAR OS and configurations

Timing Protection Mechanism



Timing Protection – Budget Monitoring

There are four types of Budget Monitoring

- ❑ Execution Budget
- ❑ Resource Lock Budget
- ❑ Os Interrupt Lock Budget (CAT2)
- ❑ All Interrupt Lock Budget (CAT1 + CAT2)

Execution Budget

- ❑ Ensures that no Task/ISR holds the CPU for more than the configured Execution Budget time

- ❑ The monitoring begins as soon as a Task/ISR goes to a "run" state and stops when a Task/ISR is terminated

- ❑ In case a Task/ISR is pre-empted by a higher priority Task/ISR, the execution budget is paused and when the Task/ISR again acquires the "running" state execution budget is resumed

Use Case:

Detection of faulty tasks which holds excess CPU time

Timing Protection – Arrival Monitoring

- ❑ Ensures that no Task/ISR is activated before the configured time frame

- ❑ The timer monitors the inter-arrival time for a Task/ISR and if the inter-arrival time is lesser than the configured time the Application is notified

Use Case:

It helps in detection of babbling idiot Tasks/Interrupts

Sample Use-Case for Timing Protection : Execution Budget

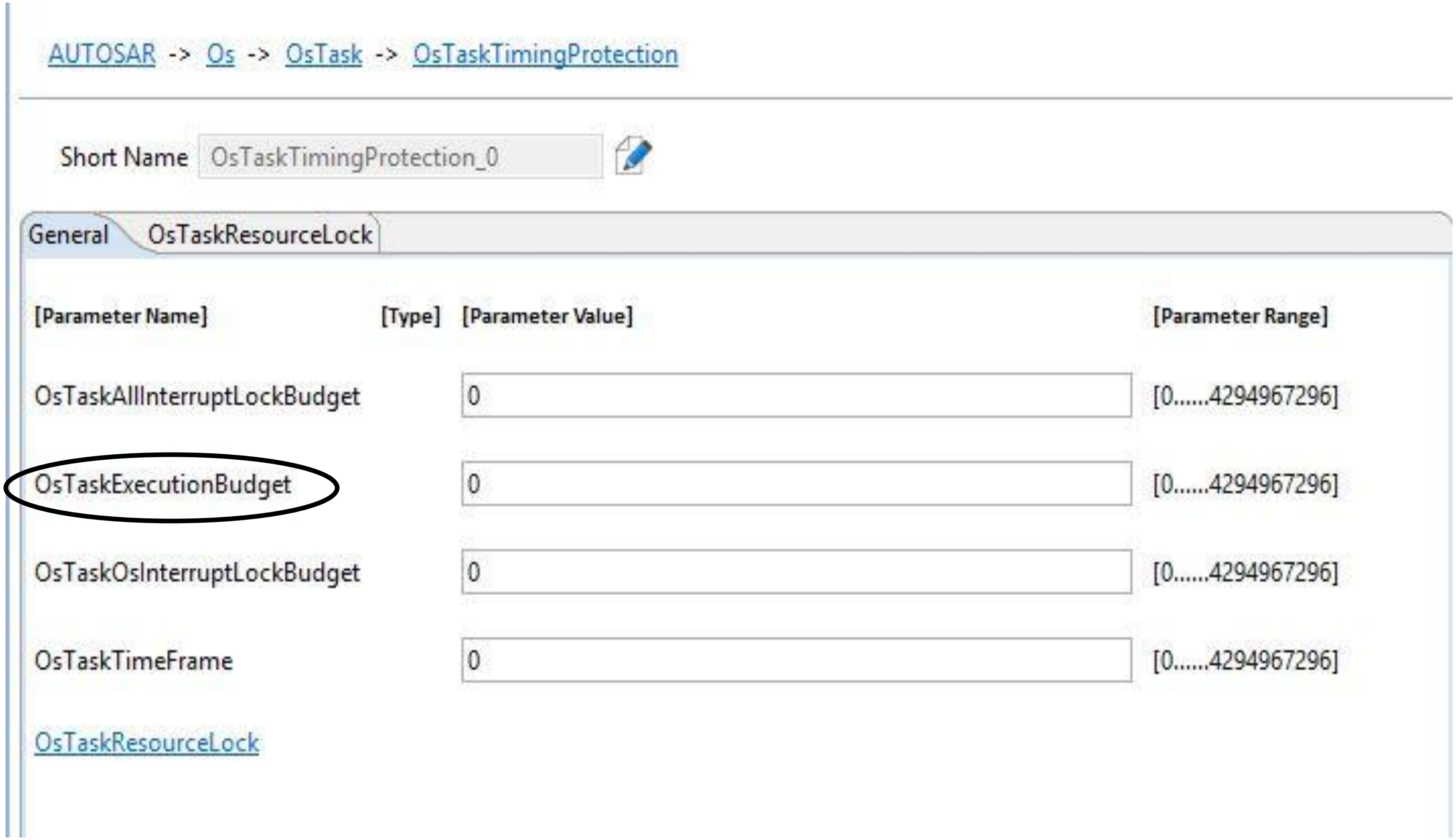
- Runnable RE_Read_ADC_Data() is invoked from TASK_RTE_READ_ADC
- This runnable reads ADC output through I/O hardware abstraction(IoHwAb)
- It is specified that one read operation takes maximum of 100 uS.

APPLICATION CONSTRAINT : The read cycle should take no longer than 100 uS

SAFETY VIEW : If time taken for this read is greater than 100 uS, it could be a faulty situation

AUTOSAR Solution : Use Execution budget for the Task

Configuration of Execution Budget



AUTOSAR -> Os -> OsTask -> OsTaskTimingProtection

Short Name OsTaskTimingProtection_0

[Parameter Name]	[Type]	[Parameter Value]	[Parameter Range]
OsTaskAllInterruptLockBudget		0	[0......4294967296]
OsTaskExecutionBudget		0	[0......4294967296]
OsTaskOsInterruptLockBudget		0	[0......4294967296]
OsTaskTimeFrame		0	[0......4294967296]

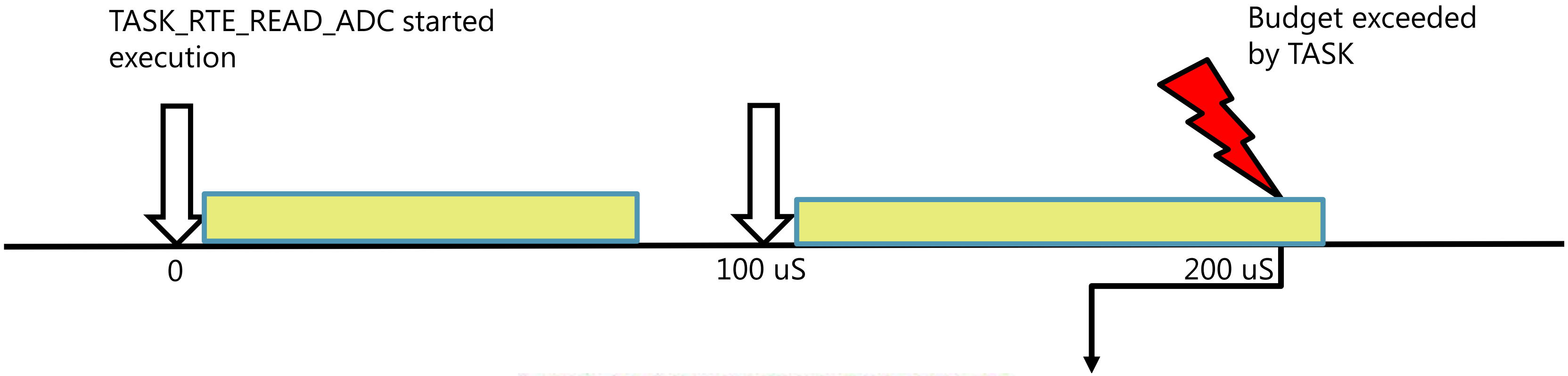
OsTaskResourceLock

Configuration of Execution Budget

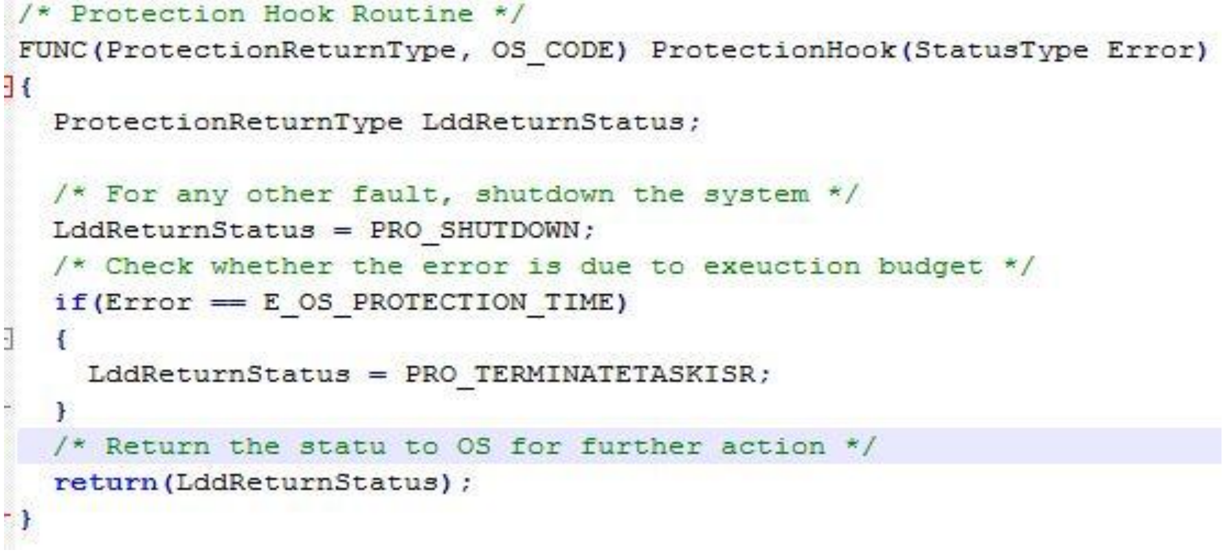
- For the mentioned scenario, create an instance of OsTaskTimingProtection of the TASK_RTE_READ_ADC.
- As this task is to be guarded for execution limit, i.e. execution for greater than 100 uS is a faulty situation, the OsTaskExecutionBudget needs to be configured as 0.0001 (100 uS)
- Whenever the execution for this task exceeds 100 uS (limit), in this case, it would indicate that Runnable entity took longer than expected time. So, OS shall produce a fault and hit ProtectionHook.
- The user can then decide action in this scenario by appropriately providing return value from ProtectionHook to the Operating System

- Sample on the next slide shows the protection hook and associated action of killing the faulty task

Visualization of the scenario



🟨 Execution time for Task < 100 uS



```
/* Protection Hook Routine */
FUNC(ProtectionReturnType, OS_CODE) ProtectionHook(StatusType Error)
{
  ProtectionReturnType LddReturnStatus;

  /* For any other fault, shutdown the system */
  LddReturnStatus = PRO_SHUTDOWN;
  /* Check whether the error is due to exeuction budget */
  if(Error == E_OS_PROTECTION_TIME)
  {
    LddReturnStatus = PRO_TERMINATETASKISR;
  }
  /* Return the statu to OS for further action */
  return(LddReturnStatus);
}
```

Sample Use-Case for Timing Protection : Time Frame

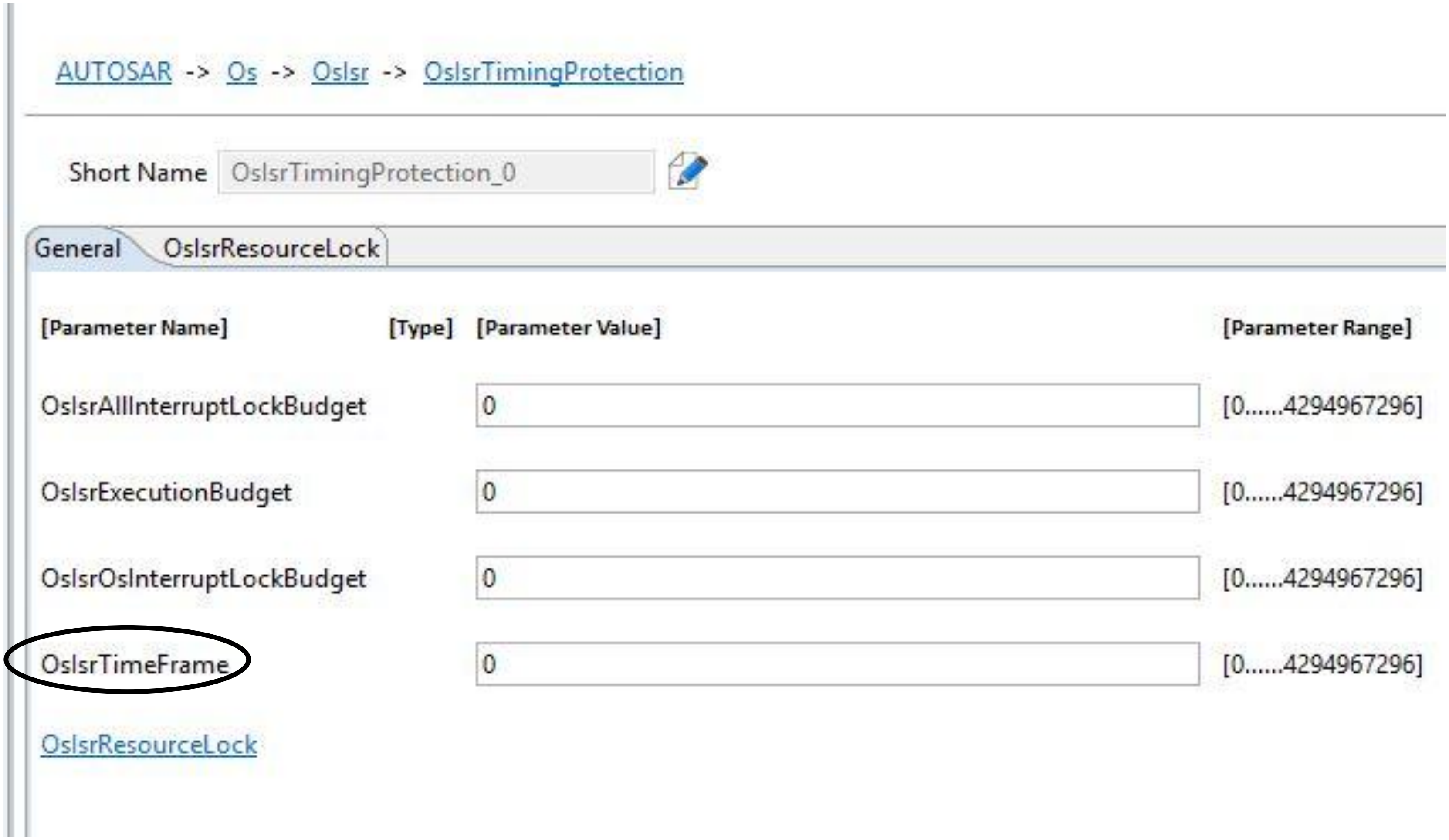
- ISR_READ_CAN_DATA is a CAN Rx ISR and as per configuration, it is invoked every 200 uS.
- ISR_READ_CAN_DATA execution time is 20 uS

APPLICATION CONSTRAINT : The ISR must receive the data on CAN channel every 200 uS

SAFETY VIEW : If the ISR is invoked between 200 uS gap, it is false triggering due to reasons like hardware fault

AUTOSAR Solution : Use Time Frame for the ISR

Configuration of Time Frame



AUTOSAR -> Os -> OsIsr -> OsIsrTimingProtection

Short Name: OsIsrTimingProtection_0

[Parameter Name]	[Type]	[Parameter Value]	[Parameter Range]
OsIsrAllInterruptLockBudget		0	[0......4294967296]
OsIsrExecutionBudget		0	[0......4294967296]
OsIsrOsInterruptLockBudget		0	[0......4294967296]
OsIsrTimeFrame		0	[0......4294967296]

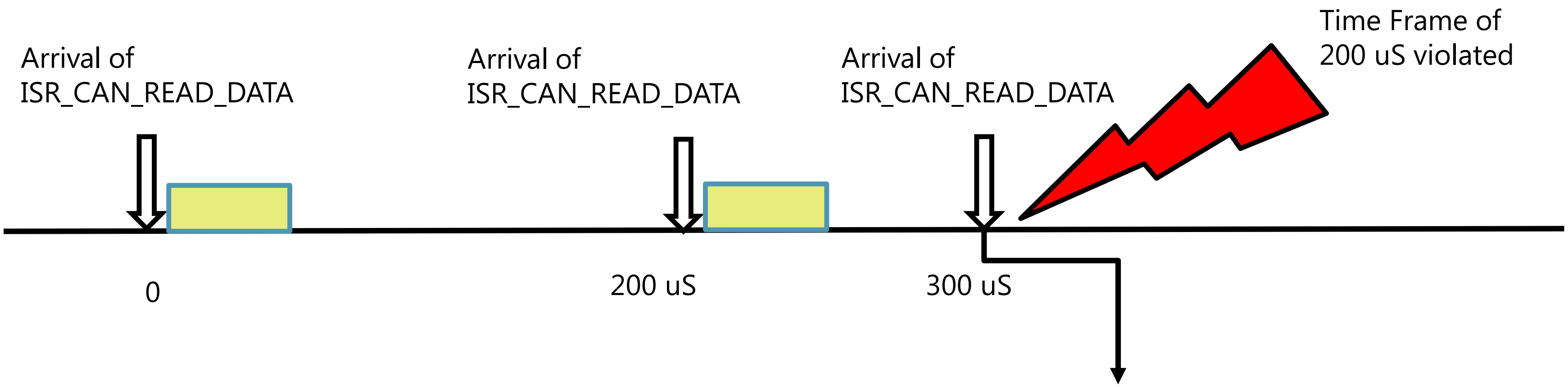
OsIsrResourceLock

Configuration for Time Frame

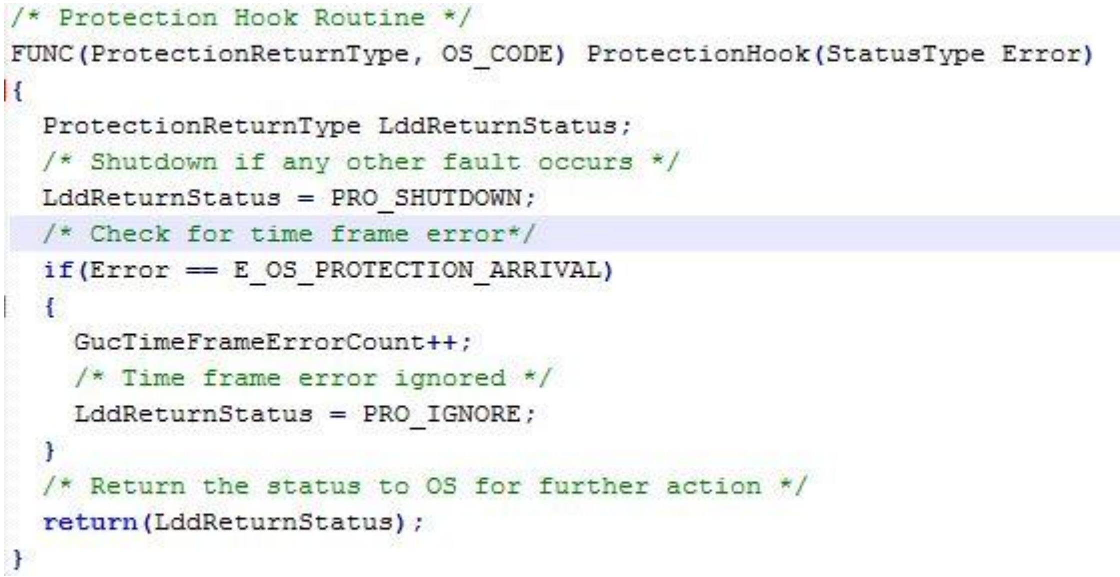
- For the scenario, create an instance of OsIsrTimingProtection container for ISR_READ_CAN_DATA
- The ISR needs to be configured for inter arrival spacing and hence, time frame OsIsrTimeFrame needs to be configured for it. The TimeFrame for the ISR needs to be configured as 0.0002 (200 uS).
- As the ISR appears, the Operating system shall monitor its next arrival and only allow the next arrival of ISR if time spacing between 2 arrivals is greater than 200 uS.
- If ISR appears in the specified time frame of 200 uS, OS shall report to the protection hook and take further action based on return value of protection hook.

- Sample shows ISR time frame scenario and protection hook to ignore the error.

Visualization of the scenario


Arrival of ISR_CAN_READ_DATA
Arrival of ISR_CAN_READ_DATA
Arrival of ISR_CAN_READ_DATA
Time Frame of 200 uS violated
0
200 uS
300 uS

🟨 Execution time for ISR : 20 uS



```
/* Protection Hook Routine */
FUNC(ProtectionReturnType, OS_CODE) ProtectionHook(StatusType Error)
{
  ProtectionReturnType LddReturnStatus;
  /* Shutdown if any other fault occurs */
  LddReturnStatus = PRO_SHUTDOWN;
  /* Check for time frame error*/
  if(Error == E_OS_PROTECTION_ARRIVAL)
  {
    GucTimeFrameErrorCount++;
    /* Time frame error ignored */
    LddReturnStatus = PRO_IGNORE;
  }
  /* Return the status to OS for further action */
  return(LddReturnStatus);
}
```

Conclusions

- Scenarios like overrun by TASK/ISR can be guarded by using execution budget monitoring of the TASK/ISR
- The TASK/ISR can be spaced in terms of their arrival and hence any random triggering due to faults can be detected. Time Frame for TASK/ISR can be used for the same

Questions

Thank you

www.kpit.com