Application   **RTE**   **Driver(3)**

seq InitSequence

App_Init

RTE_Init

Driver_Init
- Driver Configuration
- Driver Responds

Rte_DrvStatus_Rx(Status)
- Store status in the RTE buffer

seq Driver ISR Sequence

Driver_ISR
- Read data from the device (from a register)
- Reset DrvAliveCntr (Alive counter)

Rte_DrvData_Rx(Data)
- Store Driver Data in the RTE buffer

Rte_AppDrv_Cbk()
- Increment DrvMsg Counter

Rte_ReadDrvData(&Data)

seq Driver_MainCyclic

Driver_MainCyclic
- Increment DrvAliveCntr (Alive counter)

alt
[AliveCntr > 3]

Rte_DrvStatus_Rx(FALSE)
- Store status in the RTE buffer

alt
[AliveCntr <= 3]

Rte_DrvStatus_Rx(TRUE)
- Store status in the RTE buffer

seq App_MainCyclic

App_MainCyclic

Rte_Read_DrvStatus(&Status)

alt
[DrvSts == TRUE]

Process DrvData
Execute App Logic

alt
[DrvSts == FALSE]

Panic()

Below functions (with periodicity) are triggered from the Test System

App_Init
Drv_Init
App_MainCyclic - 1000ms
Drv_MainCyclic - 500ms
Drv_ISR - 250ms

Key Requirements:
- Alive Counter is a watchdog concept to monitor the driver
- Drv_ISR Emulated behavior of a Device Interrupt
- Device data is fetched via interrupt methodology by the driver
- Device data is processed and status is monitored via Polling methodology by application
- Rte is a runtime environment to provide abstraction between HW and application layer
- Rte also provides buffers for the status and data