|  |  |
| --- | --- |
| ERH  Communication Interaction Layer | |
| **Summary** | This is the Software Detailed Design Document for the *DAIMLER MMA* Project. |

|  |  |  |
| --- | --- | --- |
| **Author** | **Review** | **Approval** |
| Title: Obada Mirela Ioana | See Project Master Document for the roles and Project Members List for the name of people | See Project Master Document for the roles and Project Members List for the name of people |
|  |  |  |
| **Distribution** | | |
| See Project Master Document for the roles and Project Members List for the name of people | See Project Master Document for the roles and Project Members List for the name of people | See Project Master Document for the roles and Project Members List for the name of people |

Table of contents

[General Information 5](#_Toc163143205)

[1.1. Revision history \* 5](#_Toc163143206)

[1.2. Purpose and Scope 5](#_Toc163143207)

[1.3. Referenced documents 5](#_Toc163143208)

[1.3.1. External documents 5](#_Toc163143209)

[1.3.2. Internal Documents 5](#_Toc163143210)

[1.3.3. Terminology and definitions 6](#_Toc163143211)

[2. SW Module Detailed Design 7](#_Toc163143212)

[2.1. Overview 7](#_Toc163143213)

[2.2. Traceability 7](#_Toc163143214)

[3. Features 21](#_Toc163143215)

[3.1. Services 21](#_Toc163143216)

[3.1.1. ERH\_cyclic 21](#_Toc163143217)

[3.1.2. ERH\_Init 22](#_Toc163143218)

[3.1.3. erh\_CheckAecId 22](#_Toc163143219)

[3.1.4. erh\_ManageFailedCounterEvent 24](#_Toc163143220)

[3.1.5. erh\_ManagePassedCounterEvent 25](#_Toc163143221)

[3.1.6. erh\_ManageAECCommand 25](#_Toc163143222)

[3.1.7. ERH\_runGetAecStatus 26](#_Toc163143223)

[3.1.8. ERH\_runGetAecGroupStatus 27](#_Toc163143224)

[3.1.9. ERH\_ResetAllAECStatus 28](#_Toc163143225)

[3.1.10. erh\_UpdateGroup 29](#_Toc163143226)

[3.1.11. ERH\_runSetAecEvent 29](#_Toc163143227)

[3.2. Types 30](#_Toc163143228)

[3.2.1. erh\_stAECConfigurationType 30](#_Toc163143229)

[3.3. Variabiles 31](#_Toc163143230)

[3.3.1. erh\_au16AecCounter 31](#_Toc163143231)

[3.3.2. erh\_u32GroupsStatus 31](#_Toc163143232)

[3.3.3. erh\_au8AecStatus 31](#_Toc163143233)

[3.3.4. erh\_u8PendingEventAddIndex 31](#_Toc163143234)

[3.3.5. erh\_u8PendingEventRunIndex 31](#_Toc163143235)

[3.3.6. erh\_au8PendingEventCommand 32](#_Toc163143236)

[3.3.7. erh\_au8PendingEventAecIdentifier 32](#_Toc163143237)

[3.3.8. erh\_u8StAECWriteStatusResult 32](#_Toc163143238)

[3.3.9. erh\_bNvmRecordRequested 32](#_Toc163143239)

[3.3.10. erh\_u8OtherGroupValue 32](#_Toc163143240)

[3.3.11. erh\_cpstAecData 32](#_Toc163143241)

[3.3.12. ERH\_astAECsDefinitions 33](#_Toc163143242)

[3.3.13. erh\_au32GroupCounter 33](#_Toc163143243)

[3.3.14. erh\_u16DebounceCounter 33](#_Toc163143244)

[3.3.15. ERH\_u16AECsForControlDTC\_StorageGroup[KU8\_EIGHT] 33](#_Toc163143245)

[3.3.16. ERH\_u16AECsForECULevel\_StorageGroup[KU8\_FIFTEEN] 33](#_Toc163143246)

[3.3.17. ERH\_u16AECsForNetworkCom\_StorageGroup[KU8\_FIFTEEN] 34](#_Toc163143247)

[3.3.18. ERH\_u16AECsForPowerDistribution\_StorageGroup[KU8\_SIX] 34](#_Toc163143248)

[3.3.19. ERH\_u16DebounceCounter\_NetworkComunication 35](#_Toc163143249)

[3.3.20. ERH\_u16DebounceCounter\_NetworkComunication15 35](#_Toc163143250)

[3.3.21. ERH\_u16DebounceCounter\_PowerDistribution 35](#_Toc163143251)

[3.3.22. ERH\_u16StorageConditionStatus 35](#_Toc163143252)

[3.3.23. ERH\_u8Index\_ECULevel 35](#_Toc163143253)

[3.3.24. ERH\_u8Index\_NetworkCom 36](#_Toc163143254)

[3.3.25. ERH\_u8Index\_NoStorageCondition 36](#_Toc163143255)

[3.3.26. ERH\_u8Index\_PowerDistribution 36](#_Toc163143256)

[3.4. Macros 36](#_Toc163143257)

[3.4.1. KU8\_MAX\_PENDING\_EVENT 36](#_Toc163143258)

[3.4.2. KU8\_NUMBER\_OF\_GROUPS 36](#_Toc163143259)

[3.4.3. KU8\_NVP\_AEC\_RECORDING\_OK 37](#_Toc163143260)

[3.4.4. KU8\_NVP\_AEC\_RECORDING\_NOK 37](#_Toc163143261)

[3.4.5. KU8\_NVP\_AEC\_RECORDING\_IN\_PROGRESS 37](#_Toc163143262)

[3.4.6. ERH\_KU8\_NUMBER\_OF\_AEC 37](#_Toc163143263)

[3.4.7. KU8\_NVP\_BLOCK\_JOB\_PENDING 37](#_Toc163143264)

[3.4.8. NVP\_BLOCK\_ID\_AEC\_STATUS 37](#_Toc163143265)

[3.4.9. KU16\_COUNTER\_TRESHOLD\_IGN 38](#_Toc163143266)

[3.4.10. KU16\_ERH\_MASK\_DTC\_CONFIRMED 38](#_Toc163143267)

[3.4.11. BOOL\_ERH\_CLEAR\_STORAGE\_CONDITION 38](#_Toc163143268)

[3.4.12. BOOL\_ERH\_SET\_STORAGE\_CONDITION 38](#_Toc163143269)

[3.4.13. ERH\_SET\_STORAGE\_CONDITION\_MASK\_OFF(mask) 38](#_Toc163143270)

[3.4.14. ERH\_SET\_STORAGE\_CONDITION\_MASK\_ON(mask) 39](#_Toc163143271)

[3.4.15. KU16\_ERH\_MASK\_CONTROL\_DTC 39](#_Toc163143272)

[3.4.16. KU16\_ERH\_MASK\_DEBOUNCE\_TIMER\_NC 39](#_Toc163143273)

[3.4.17. KU16\_ERH\_MASK\_DEBOUNCE\_TIMER\_NC15 39](#_Toc163143274)

[3.4.18. KU16\_ERH\_MASK\_DEBOUNCE\_TIMER\_PD 39](#_Toc163143275)

[3.4.19. KU16\_ERH\_MASK\_DTC\_CONFIRMED 39](#_Toc163143276)

[3.4.20. KU16\_ERH\_MASK\_ECU\_LEVEL 40](#_Toc163143277)

[3.4.21. KU16\_ERH\_MASK\_NETWORK\_COM 40](#_Toc163143278)

[3.4.22. KU16\_ERH\_MASK\_NETWORK\_COM15 40](#_Toc163143279)

[3.4.23. KU16\_ERH\_MASK\_POWER\_DISTRIBUTION 40](#_Toc163143280)

[3.4.24. KU16\_ERH\_SC10\_0\_debounceTimer 40](#_Toc163143281)

[3.4.25. KU16\_ERH\_SC10\_1\_debounceTimer 40](#_Toc163143282)

[3.4.26. KU16\_ERH\_SC10\_2\_debounceTimer 41](#_Toc163143283)

[3.4.27. KU16\_ERH\_SC11\_VehicleStartup 41](#_Toc163143284)

[3.4.28. KU16\_ERH\_SC12\_ProductionMode 41](#_Toc163143285)

[3.4.29. KU16\_ERH\_SC1\_ControlDTC 41](#_Toc163143286)

[3.4.30. KU16\_ERH\_SC2\_IgnitonStatus 41](#_Toc163143287)

[3.4.31. KU16\_ERH\_SC3\_LocalVoltage 41](#_Toc163143288)

[3.4.32. KU16\_ERH\_SC4\_SystemVoltage 42](#_Toc163143289)

[3.4.33. KU16\_ERH\_SC5\_TransportationMode 42](#_Toc163143290)

[3.4.34. KU16\_ERH\_SC8\_CommunicationStatus 42](#_Toc163143291)

[3.4.35. KU16\_ERH\_TIMER\_TRESHOLD 42](#_Toc163143292)

[4. EEPROM 42](#_Toc163143293)

[5. Configuration 42](#_Toc163143294)

[6. Compilation Options 43](#_Toc163143295)

# Table of figures

[Figure 1 ERH\_cyclic 22](#_Toc163143195)

[Figure 2: ERH\_Init 22](#_Toc163143196)

[Figure 3: erh\_CheckAecId 24](#_Toc163143197)

[Figure 4: erh\_ManageFailedCounterEvent 25](#_Toc163143198)

[Figure 5: erh\_ManagePassedCounterEvent 25](#_Toc163143199)

[Figure 6: erh\_ManageAECCommand 26](#_Toc163143200)

[Figure 7: ERH\_runGetAecStatus 27](#_Toc163143201)

[Figure 8: ERH\_runGetAecGroupsStatus 28](#_Toc163143202)

[Figure 9: ERH\_ResetAllAECStatus 29](#_Toc163143203)

[Figure 10: erh\_UpdateGroup 29](#_Toc163143204)

# General Information

## Revision history \*

|  |  |  |  |
| --- | --- | --- | --- |
| **Revision** | **Date** | **Author(s)** | **Description/comment** |
| 1.1.5.1 | 9/9/2022 | Mirela Obada | First revision on Daimler MMA |
| 1.1.5.2 | 9/9/2022 | Mirela Obada | Update for 3.0 release |
| 1.1.5.3 | 12/9/2022 | Mirela Obada | Update traceability after SRM |
| 1.1.5.4 | 14/11/2022 | Mirela Obada | Update document for 4.0 |
| 1.1.5.5 | 15/11/2022 | Mirela Obada | Update with figures for all functions |
| 1.1.5.6 | 21/11/22 | Mirela Obada | Update traceability |
| 1.1.5.7 | 22/11/22 | Mirela Obada | Update after review |
| 1.1.5.8 | 20/12/2022 | Madalina Serban | Update after disable AECs implementation |
| 1.1.5.9 | 09/02/2023 | Madalina Serban | Update for R5.0 |
| 1.1.5.10 | 16/02/2023 | Madalina Serban | Fixes after review |
| 1.1.5.11 | 08/03/2023 | Mirela Obada | Update ERH\_Init() diagaram |
| 1.1.5.12 | 08/03/2023 | Mirela Obada | Fix finding after review |
| 1.1.5.13 | 18/04/2023 | Mirela Obada | Update for R6.0 |
| 1.1.5.14 | 05/05/2023 | Mirela Obada | Update after SRM |
| 1.1.5.15 | 08/05/2023 | Mirela Obada | Update diagram for erh\_UpdateGroup |
| 1.1.5.16 | 15/05/2023 | Mirela Obada | Update traceability |
| 1.1.5.17 | 21/06/2023 | Madalina Serban | Update for R6.1 |
| 1.1.5.18 | 22/06/2023 | Madalina Serban | Updated figure |
| 1.1.5.19 | 23/08/2023 | Mirela Obada | Update document for R07.0 |
| 1.1.5.20 | 25/08/2023 | Mirela Obada | Update traceability |
| 1.1.5.21 | 01/09/2023 | Mirela Obada | Update after review |
| 1.1.5.22 | 12/01/2024 | Madalina Serban | Updated document for R08.1 |
| 1.1.5.23 | 23.01.2024 | Mirela Obada | Update for 8.1 |
| 1.1.5.24 | 04/04/2024 | Mirela Obada | Update for 9.0 |
| 1.1.5.25 | 04/04/2024 | Mirela Obada | Update after reqtify |

*\*

*\* Template history is found in the CM tool used for templates*

## Purpose and Scope

The purpose of this document is to provide an overview of the ERH operation principle, and to present the implementation choices in terms of module and function splitting.

## Referenced documents

### External documents

|  |  |  |
| --- | --- | --- |
| **Id** | **Title** | **Reference** |
|  |  |  |
|  |  |  |
|  |  |  |

### Internal Documents

|  |  |  |
| --- | --- | --- |
| **Id** | **Title** | **Reference** |
|  | DAIMLER\_MMA\_SWarchitectureDesignInterfaceDescription.docx |  |
|  | ERH - Design Interface Description.docx |  |
|  | SBE\_4G\_NVP\_layout.xls |  |
|  |  |  |

### Terminology and definitions

|  |  |
| --- | --- |
| **Terminology** | **Meaning** |
| AAU | Atomic architectural unit |
| SW | software |
|  |  |

# SW Module Detailed Design

## Overview

The aim of the “ERH” component is to receive error notifications and proper functioning notifications from other modules and filter them to keep an up to date a list of error status.

Error can be handled on two different ways:

* Timer based: An internal timer is managed by the ERH for each AEC. The timer is started when a failed event is received and stopped when the AEC qualification time reached, or a passed event is received.
* Counter base: An internal counter is managed by the ERH for each AEC. The counter is incremented/decremented depending on the received events (failed or passed).

Also the ERH module is responsible for manage DTC qualification and disqualification depending on status of the AECs groups.

## Traceability

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirements** | **Criteria** | **Linked Runnable** | **Source** |
| DSG\_ERH\_00001 | The ERH component shall check the validity of the current AEC identifier. The AEC identifier should be in the following range to be valid: 0x00 - 0x30. | Requirement covered by [erh\_CheckAecId](#_erh_CheckAecId) function | ARCH\_SW\_ERH\_0050;  ARCH\_SW\_ERH\_0074; ARCH\_SW\_ERH\_0064; ARCH\_SW\_ERH\_0063; ARCH\_SW\_ERH\_0070; |
| DSG\_ERH\_00002 | The [erh\_CheckAecId](#_erh_CheckAecId) function shall return B\_TRUE (0xAA) value if the AEC identifier is valid, if not it shall return B\_FALSE (0x55) value. | Requirement covered by [erh\_CheckAecId](#_erh_CheckAecId) function | ARCH\_SW\_ERH\_0050;  ARCH\_SW\_ERH\_0074; ARCH\_SW\_ERH\_0064; ARCH\_SW\_ERH\_0063; ARCH\_SW\_ERH\_0070; |
| DSG\_ERH\_00003 | The ERH component shall convert the current AEC identifier to an index if it is valid | Requirement covered by [erh\_CheckAecId](#_erh_CheckAecId) function | ARCH\_SW\_ERH\_0050;  ARCH\_SW\_ERH\_0074; ARCH\_SW\_ERH\_0064; ARCH\_SW\_ERH\_0063; ARCH\_SW\_ERH\_0070; |
| DSG\_ERH\_00004 | The ERH component shall update the AEC index and the corresponding debouncing/deskill thresholds if the AEC identifier is valid. | Requirement covered by [erh\_CheckAecId](#_erh_CheckAecId) function | ARCH\_SW\_ERH\_0050;  ARCH\_SW\_ERH\_0074; ARCH\_SW\_ERH\_0064; ARCH\_SW\_ERH\_0063; ARCH\_SW\_ERH\_0070; |
| DSG\_ERH\_00005 | The ERH component shall manage the current AEC Status if a failed event occurred. | Requirement covered by [erh\_ManageFailedCounterEvent](#_erh_ManageFailedCounterEvent) function | ARCH\_SW\_ERH\_0001; ARCH\_SW\_ERH\_0004; ARCH\_SW\_ERH\_0006; ARCH\_SW\_ERH\_0008; ARCH\_SW\_ERH\_0010; ARCH\_SW\_ERH\_0012; ARCH\_SW\_ERH\_0014; ARCH\_SW\_ERH\_0016; ARCH\_SW\_ERH\_0018; ARCH\_SW\_ERH\_0019; ARCH\_SW\_ERH\_0020; ARCH\_SW\_ERH\_0021; ARCH\_SW\_ERH\_0022; ARCH\_SW\_ERH\_0023; ARCH\_SW\_ERH\_0024; ARCH\_SW\_ERH\_0025; ARCH\_SW\_ERH\_0026; ARCH\_SW\_ERH\_0027; ARCH\_SW\_ERH\_0028; ARCH\_SW\_ERH\_0029; ARCH\_SW\_ERH\_0030; ARCH\_SW\_ERH\_0031; ARCH\_SW\_ERH\_0032;  ARCH\_SW\_ERH\_0033; ARCH\_SW\_ERH\_0034; ARCH\_SW\_ERH\_0035;  ARCH\_SW\_ERH\_0036; ARCH\_SW\_ERH\_0037; ARCH\_SW\_ERH\_0038;ARCH\_SW\_ERH\_0039; ARCH\_SW\_ERH\_0040; ARCH\_SW\_ERH\_0041; ARCH\_SW\_ERH\_0042; ARCH\_SW\_ERH\_0043; ARCH\_SW\_ERH\_0044; ARCH\_SW\_ERH\_0045;ARCH\_SW\_ERH\_0046; ARCH\_SW\_ERH\_0047;ARCH\_SW\_ERH\_0048;ARCH\_SW\_ERH\_0080; ARCH\_SW\_ERH\_0086; ARCH\_SW\_ERH\_0087; ARCH\_SW\_ERH\_0088; ARCH\_SW\_ERH\_0089; ARCH\_SW\_ERH\_0090; ARCH\_SW\_ERH\_0092;  ARCH\_SW\_ERH\_0157; |
| DSG\_ERH\_00006 | The ERH component shall set the current AEC status to “qualified” if the qualification threshold has been reached. | Requirement covered by [erh\_ManageFailedCounterEvent](#_erh_ManageFailedCounterEvent) function | ARCH\_SW\_ERH\_0001; ARCH\_SW\_ERH\_0004; ARCH\_SW\_ERH\_0006; ARCH\_SW\_ERH\_0008; ARCH\_SW\_ERH\_0010; ARCH\_SW\_ERH\_0012; ARCH\_SW\_ERH\_0014; ARCH\_SW\_ERH\_0016; ARCH\_SW\_ERH\_0018; ARCH\_SW\_ERH\_0019; ARCH\_SW\_ERH\_0020; ARCH\_SW\_ERH\_0021; ARCH\_SW\_ERH\_0022; ARCH\_SW\_ERH\_0023; ARCH\_SW\_ERH\_0024; ARCH\_SW\_ERH\_0025; ARCH\_SW\_ERH\_0026; ARCH\_SW\_ERH\_0027; ARCH\_SW\_ERH\_0028; ARCH\_SW\_ERH\_0029; ARCH\_SW\_ERH\_0030; ARCH\_SW\_ERH\_0031; ARCH\_SW\_ERH\_0032;  ARCH\_SW\_ERH\_0033; ARCH\_SW\_ERH\_0034; ARCH\_SW\_ERH\_0035;  ARCH\_SW\_ERH\_0036; ARCH\_SW\_ERH\_0037; ARCH\_SW\_ERH\_0038;ARCH\_SW\_ERH\_0039; ARCH\_SW\_ERH\_0040; ARCH\_SW\_ERH\_0041; ARCH\_SW\_ERH\_0042; ARCH\_SW\_ERH\_0043; ARCH\_SW\_ERH\_0044; ARCH\_SW\_ERH\_0045;ARCH\_SW\_ERH\_0046; ARCH\_SW\_ERH\_0047;ARCH\_SW\_ERH\_0048;  ARCH\_SW\_ERH\_0080; ARCH\_SW\_ERH\_0086; ARCH\_SW\_ERH\_0087; ARCH\_SW\_ERH\_0088; ARCH\_SW\_ERH\_0089; ARCH\_SW\_ERH\_0090;  ARCH\_SW\_ERH\_0092; ARCH\_SW\_ERH\_0157; |
| DSG\_ERH\_00007 | The ERH component shall increment with one step the current debounce counter if the AEC qualification threshold has not been reached. | Requirement covered by [erh\_ManageFailedCounterEvent](#_erh_ManageFailedCounterEvent) function | ARCH\_SW\_ERH\_0001; ARCH\_SW\_ERH\_0004; ARCH\_SW\_ERH\_0006; ARCH\_SW\_ERH\_0008; ARCH\_SW\_ERH\_0010; ARCH\_SW\_ERH\_0012; ARCH\_SW\_ERH\_0014; ARCH\_SW\_ERH\_0016; ARCH\_SW\_ERH\_0018; ARCH\_SW\_ERH\_0019; ARCH\_SW\_ERH\_0020; ARCH\_SW\_ERH\_0021; ARCH\_SW\_ERH\_0022; ARCH\_SW\_ERH\_0023; ARCH\_SW\_ERH\_0024; ARCH\_SW\_ERH\_0025; ARCH\_SW\_ERH\_0026; ARCH\_SW\_ERH\_0027; ARCH\_SW\_ERH\_0028; ARCH\_SW\_ERH\_0029; ARCH\_SW\_ERH\_0030; ARCH\_SW\_ERH\_0031; ARCH\_SW\_ERH\_0032;  ARCH\_SW\_ERH\_0033; ARCH\_SW\_ERH\_0034; ARCH\_SW\_ERH\_0035;  ARCH\_SW\_ERH\_0036; ARCH\_SW\_ERH\_0037; ARCH\_SW\_ERH\_0038;ARCH\_SW\_ERH\_0039; ARCH\_SW\_ERH\_0040; ARCH\_SW\_ERH\_0041; ARCH\_SW\_ERH\_0042; ARCH\_SW\_ERH\_0043; ARCH\_SW\_ERH\_0044; ARCH\_SW\_ERH\_0045;ARCH\_SW\_ERH\_0046; ARCH\_SW\_ERH\_0047;ARCH\_SW\_ERH\_0048; ARCH\_SW\_ERH\_0086; ARCH\_SW\_ERH\_0087; ARCH\_SW\_ERH\_0088; ARCH\_SW\_ERH\_0089; ARCH\_SW\_ERH\_0090;  ARCH\_SW\_ERH\_0092; ARCH\_SW\_ERH\_0157; |
| DSG\_ERH\_00008 | If the qualification threshold has been reached the ERH component shall perform the following actions to update the AEC status:  -set ever tested status,  -set ever qualified status,  -set qualified status,  -clear in progress status | Requirement covered by [erh\_ManageFailedCounterEvent](#_erh_ManageFailedCounterEvent) function | ARCH\_SW\_ERH\_0001; ARCH\_SW\_ERH\_0004; ARCH\_SW\_ERH\_0006; ARCH\_SW\_ERH\_0008; ARCH\_SW\_ERH\_0010; ARCH\_SW\_ERH\_0012; ARCH\_SW\_ERH\_0014; ARCH\_SW\_ERH\_0016; ARCH\_SW\_ERH\_0018; ARCH\_SW\_ERH\_0019; ARCH\_SW\_ERH\_0020; ARCH\_SW\_ERH\_0021; ARCH\_SW\_ERH\_0022; ARCH\_SW\_ERH\_0023; ARCH\_SW\_ERH\_0024; ARCH\_SW\_ERH\_0025; ARCH\_SW\_ERH\_0026; ARCH\_SW\_ERH\_0027; ARCH\_SW\_ERH\_0028; ARCH\_SW\_ERH\_0029; ARCH\_SW\_ERH\_0030; ARCH\_SW\_ERH\_0031; ARCH\_SW\_ERH\_0032;  ARCH\_SW\_ERH\_0033; ARCH\_SW\_ERH\_0034; ARCH\_SW\_ERH\_0035;  ARCH\_SW\_ERH\_0036; ARCH\_SW\_ERH\_0037; ARCH\_SW\_ERH\_0038;ARCH\_SW\_ERH\_0039; ARCH\_SW\_ERH\_0040; ARCH\_SW\_ERH\_0041; ARCH\_SW\_ERH\_0042; ARCH\_SW\_ERH\_0043; ARCH\_SW\_ERH\_0044; ARCH\_SW\_ERH\_0045;ARCH\_SW\_ERH\_0046; ARCH\_SW\_ERH\_0047;ARCH\_SW\_ERH\_0048; ARCH\_SW\_ERH\_0086; ARCH\_SW\_ERH\_0087; ARCH\_SW\_ERH\_0088; ARCH\_SW\_ERH\_0089; ARCH\_SW\_ERH\_0090;  ARCH\_SW\_ERH\_0092; ARCH\_SW\_ERH\_0157; |
| DSG\_ERH\_00009 | The ERH component shall update the group status if the qualification threshold has been reached. | Requirement covered by [erh\_ManageFailedCounterEvent](#_erh_ManageFailedCounterEvent) function | ARCH\_SW\_ERH\_0001; ARCH\_SW\_ERH\_0004; ARCH\_SW\_ERH\_0006; ARCH\_SW\_ERH\_0008; ARCH\_SW\_ERH\_0010; ARCH\_SW\_ERH\_0012; ARCH\_SW\_ERH\_0014; ARCH\_SW\_ERH\_0016; ARCH\_SW\_ERH\_0018; ARCH\_SW\_ERH\_0019; ARCH\_SW\_ERH\_0020; ARCH\_SW\_ERH\_0021; ARCH\_SW\_ERH\_0022; ARCH\_SW\_ERH\_0023; ARCH\_SW\_ERH\_0024; ARCH\_SW\_ERH\_0025; ARCH\_SW\_ERH\_0026; ARCH\_SW\_ERH\_0027; ARCH\_SW\_ERH\_0028; ARCH\_SW\_ERH\_0029; ARCH\_SW\_ERH\_0030; ARCH\_SW\_ERH\_0031; ARCH\_SW\_ERH\_0032;  ARCH\_SW\_ERH\_0033; ARCH\_SW\_ERH\_0034; ARCH\_SW\_ERH\_0035;  ARCH\_SW\_ERH\_0036; ARCH\_SW\_ERH\_0037; ARCH\_SW\_ERH\_0038;ARCH\_SW\_ERH\_0039; ARCH\_SW\_ERH\_0040; ARCH\_SW\_ERH\_0041; ARCH\_SW\_ERH\_0042; ARCH\_SW\_ERH\_0043; ARCH\_SW\_ERH\_0044; ARCH\_SW\_ERH\_0045;ARCH\_SW\_ERH\_0046; ARCH\_SW\_ERH\_0047;ARCH\_SW\_ERH\_0048; ARCH\_SW\_ERH\_0086; ARCH\_SW\_ERH\_0087; ARCH\_SW\_ERH\_0088; ARCH\_SW\_ERH\_0089; ARCH\_SW\_ERH\_0090;  ARCH\_SW\_ERH\_0092; ARCH\_SW\_ERH\_0157; |
| DSG\_ERH\_00010 | The ERH component shall store the current AEC status in NVP variable. | Requirement covered by [erh\_ManageFailedCounterEvent](#_erh_ManageFailedCounterEvent) function | ARCH\_SW\_ERH\_0001; ARCH\_SW\_ERH\_0004; ARCH\_SW\_ERH\_0006; ARCH\_SW\_ERH\_0008; ARCH\_SW\_ERH\_0010; ARCH\_SW\_ERH\_0012; ARCH\_SW\_ERH\_0014; ARCH\_SW\_ERH\_0016; ARCH\_SW\_ERH\_0018; ARCH\_SW\_ERH\_0019; ARCH\_SW\_ERH\_0020; ARCH\_SW\_ERH\_0021; ARCH\_SW\_ERH\_0022; ARCH\_SW\_ERH\_0023; ARCH\_SW\_ERH\_0024; ARCH\_SW\_ERH\_0025; ARCH\_SW\_ERH\_0026; ARCH\_SW\_ERH\_0027; ARCH\_SW\_ERH\_0028; ARCH\_SW\_ERH\_0029; ARCH\_SW\_ERH\_0030; ARCH\_SW\_ERH\_0031; ARCH\_SW\_ERH\_0032;  ARCH\_SW\_ERH\_0033; ARCH\_SW\_ERH\_0034; ARCH\_SW\_ERH\_0035;  ARCH\_SW\_ERH\_0036; ARCH\_SW\_ERH\_0037; ARCH\_SW\_ERH\_0038;ARCH\_SW\_ERH\_0039; ARCH\_SW\_ERH\_0040; ARCH\_SW\_ERH\_0041; ARCH\_SW\_ERH\_0042; ARCH\_SW\_ERH\_0043; ARCH\_SW\_ERH\_0044; ARCH\_SW\_ERH\_0045;ARCH\_SW\_ERH\_0046; ARCH\_SW\_ERH\_0047;ARCH\_SW\_ERH\_0048; ARCH\_SW\_ERH\_0086; ARCH\_SW\_ERH\_0087; ARCH\_SW\_ERH\_0088; ARCH\_SW\_ERH\_0089; ARCH\_SW\_ERH\_0090;  ARCH\_SW\_ERH\_0092; ARCH\_SW\_ERH\_0157; |
| DSG\_ERH\_00011 | The ERH component shall reset all AEC status and keep only “ever qualified” and “ever failed” status if the AEC deskill threshold has been reached. | Requirement covered by [erh\_ManageFailedCounterEvent](#_erh_ManageFailedCounterEvent) function | ARCH\_SW\_ERH\_0001; ARCH\_SW\_ERH\_0004; ARCH\_SW\_ERH\_0006; ARCH\_SW\_ERH\_0008; ARCH\_SW\_ERH\_0010; ARCH\_SW\_ERH\_0012; ARCH\_SW\_ERH\_0014; ARCH\_SW\_ERH\_0016; ARCH\_SW\_ERH\_0018; ARCH\_SW\_ERH\_0019; ARCH\_SW\_ERH\_0020; ARCH\_SW\_ERH\_0021; ARCH\_SW\_ERH\_0022; ARCH\_SW\_ERH\_0023; ARCH\_SW\_ERH\_0024; ARCH\_SW\_ERH\_0025; ARCH\_SW\_ERH\_0026; ARCH\_SW\_ERH\_0027; ARCH\_SW\_ERH\_0028; ARCH\_SW\_ERH\_0029; ARCH\_SW\_ERH\_0030; ARCH\_SW\_ERH\_0031; ARCH\_SW\_ERH\_0032;  ARCH\_SW\_ERH\_0033; ARCH\_SW\_ERH\_0034; ARCH\_SW\_ERH\_0035;  ARCH\_SW\_ERH\_0036; ARCH\_SW\_ERH\_0037; ARCH\_SW\_ERH\_0038;ARCH\_SW\_ERH\_0039; ARCH\_SW\_ERH\_0040; ARCH\_SW\_ERH\_0041; ARCH\_SW\_ERH\_0042; ARCH\_SW\_ERH\_0043; ARCH\_SW\_ERH\_0044; ARCH\_SW\_ERH\_0045;ARCH\_SW\_ERH\_0046; ARCH\_SW\_ERH\_0047;ARCH\_SW\_ERH\_0048;  ARCH\_SW\_ERH\_0080; ARCH\_SW\_ERH\_0086; ARCH\_SW\_ERH\_0087; ARCH\_SW\_ERH\_0088; ARCH\_SW\_ERH\_0089; ARCH\_SW\_ERH\_0090;  ARCH\_SW\_ERH\_0092; ARCH\_SW\_ERH\_0157; |
| DSG\_ERH\_00012 | The ERH component shall manage the current AEC Status if a passed event occurred. | Requirement covered by [erh\_ManagePassedCounterEvent](#_erh_ManagePassedCounterEvent) function | ARCH\_SW\_ERH\_0001; ARCH\_SW\_ERH\_0004; ARCH\_SW\_ERH\_0006; ARCH\_SW\_ERH\_0008; ARCH\_SW\_ERH\_0010; ARCH\_SW\_ERH\_0012; ARCH\_SW\_ERH\_0014; ARCH\_SW\_ERH\_0016; ARCH\_SW\_ERH\_0018; ARCH\_SW\_ERH\_0019; ARCH\_SW\_ERH\_0020; ARCH\_SW\_ERH\_0021; ARCH\_SW\_ERH\_0022; ARCH\_SW\_ERH\_0023; ARCH\_SW\_ERH\_0024; ARCH\_SW\_ERH\_0025; ARCH\_SW\_ERH\_0026; ARCH\_SW\_ERH\_0027; ARCH\_SW\_ERH\_0028; ARCH\_SW\_ERH\_0029; ARCH\_SW\_ERH\_0030; ARCH\_SW\_ERH\_0031; ARCH\_SW\_ERH\_0032;  ARCH\_SW\_ERH\_0033; ARCH\_SW\_ERH\_0034; ARCH\_SW\_ERH\_0035;  ARCH\_SW\_ERH\_0036; ARCH\_SW\_ERH\_0037; ARCH\_SW\_ERH\_0038;ARCH\_SW\_ERH\_0039; ARCH\_SW\_ERH\_0040; ARCH\_SW\_ERH\_0041; ARCH\_SW\_ERH\_0042; ARCH\_SW\_ERH\_0043; ARCH\_SW\_ERH\_0044; ARCH\_SW\_ERH\_0045;ARCH\_SW\_ERH\_0046; ARCH\_SW\_ERH\_0047;ARCH\_SW\_ERH\_0048;  ARCH\_SW\_ERH\_0080; ARCH\_SW\_ERH\_0086; ARCH\_SW\_ERH\_0087; ARCH\_SW\_ERH\_0088; ARCH\_SW\_ERH\_0089; ARCH\_SW\_ERH\_0090;  ARCH\_SW\_ERH\_0092; ARCH\_SW\_ERH\_0157; |
| DSG\_ERH\_00013 | The ERH component shall update the group status if the AEC deskill threshold has been reached and the qualified status has been set for current AEC. | Requirement covered by [erh\_ManagePassedCounterEvent](#_erh_ManagePassedCounterEvent) function | ARCH\_SW\_ERH\_0001; ARCH\_SW\_ERH\_0004; ARCH\_SW\_ERH\_0006; ARCH\_SW\_ERH\_0008; ARCH\_SW\_ERH\_0010; ARCH\_SW\_ERH\_0012; ARCH\_SW\_ERH\_0014; ARCH\_SW\_ERH\_0016; ARCH\_SW\_ERH\_0018; ARCH\_SW\_ERH\_0019; ARCH\_SW\_ERH\_0020; ARCH\_SW\_ERH\_0021; ARCH\_SW\_ERH\_0022; ARCH\_SW\_ERH\_0023; ARCH\_SW\_ERH\_0024; ARCH\_SW\_ERH\_0025; ARCH\_SW\_ERH\_0026; ARCH\_SW\_ERH\_0027; ARCH\_SW\_ERH\_0028; ARCH\_SW\_ERH\_0029; ARCH\_SW\_ERH\_0030; ARCH\_SW\_ERH\_0031; ARCH\_SW\_ERH\_0032;  ARCH\_SW\_ERH\_0033; ARCH\_SW\_ERH\_0034; ARCH\_SW\_ERH\_0035;  ARCH\_SW\_ERH\_0036; ARCH\_SW\_ERH\_0037; ARCH\_SW\_ERH\_0038;ARCH\_SW\_ERH\_0039; ARCH\_SW\_ERH\_0040; ARCH\_SW\_ERH\_0041; ARCH\_SW\_ERH\_0042; ARCH\_SW\_ERH\_0043; ARCH\_SW\_ERH\_0044; ARCH\_SW\_ERH\_0045;ARCH\_SW\_ERH\_0046; ARCH\_SW\_ERH\_0047;ARCH\_SW\_ERH\_0048; ARCH\_SW\_ERH\_0086; ARCH\_SW\_ERH\_0087; ARCH\_SW\_ERH\_0088; ARCH\_SW\_ERH\_0089; ARCH\_SW\_ERH\_0090;  ARCH\_SW\_ERH\_0092; ARCH\_SW\_ERH\_0157; |
| DSG\_ERH\_00014 | The ERH component shall store the current AEC status in NVP. | Requirement covered by [erh\_ManagePassedCounterEvent](#_erh_ManagePassedCounterEvent) function | ARCH\_SW\_ERH\_0001; ARCH\_SW\_ERH\_0004; ARCH\_SW\_ERH\_0006; ARCH\_SW\_ERH\_0008; ARCH\_SW\_ERH\_0010; ARCH\_SW\_ERH\_0012; ARCH\_SW\_ERH\_0014; ARCH\_SW\_ERH\_0016; ARCH\_SW\_ERH\_0018; ARCH\_SW\_ERH\_0019; ARCH\_SW\_ERH\_0020; ARCH\_SW\_ERH\_0021; ARCH\_SW\_ERH\_0022; ARCH\_SW\_ERH\_0023; ARCH\_SW\_ERH\_0024; ARCH\_SW\_ERH\_0025; ARCH\_SW\_ERH\_0026; ARCH\_SW\_ERH\_0027; ARCH\_SW\_ERH\_0028; ARCH\_SW\_ERH\_0029; ARCH\_SW\_ERH\_0030; ARCH\_SW\_ERH\_0031; ARCH\_SW\_ERH\_0032;  ARCH\_SW\_ERH\_0033; ARCH\_SW\_ERH\_0034; ARCH\_SW\_ERH\_0035;  ARCH\_SW\_ERH\_0036; ARCH\_SW\_ERH\_0037; ARCH\_SW\_ERH\_0038;ARCH\_SW\_ERH\_0039; ARCH\_SW\_ERH\_0040; ARCH\_SW\_ERH\_0041; ARCH\_SW\_ERH\_0042; ARCH\_SW\_ERH\_0043; ARCH\_SW\_ERH\_0044; ARCH\_SW\_ERH\_0045;ARCH\_SW\_ERH\_0046; ARCH\_SW\_ERH\_0047;ARCH\_SW\_ERH\_0048;  ARCH\_SW\_ERH\_0080; ARCH\_SW\_ERH\_0086; ARCH\_SW\_ERH\_0087; ARCH\_SW\_ERH\_0088; ARCH\_SW\_ERH\_0089; ARCH\_SW\_ERH\_0090;  ARCH\_SW\_ERH\_0092; ARCH\_SW\_ERH\_0157; |
| DSG\_ERH\_00015 | The ERH component shall decrement with one step the current debounce counter if the AEC deskill threshold has not been reached. | Requirement covered by [erh\_ManagePassedCounterEvent](#_erh_ManagePassedCounterEvent) function | ARCH\_SW\_ERH\_0001; ARCH\_SW\_ERH\_0004; ARCH\_SW\_ERH\_0006; ARCH\_SW\_ERH\_0008; ARCH\_SW\_ERH\_0010; ARCH\_SW\_ERH\_0012; ARCH\_SW\_ERH\_0014; ARCH\_SW\_ERH\_0016; ARCH\_SW\_ERH\_0018; ARCH\_SW\_ERH\_0019; ARCH\_SW\_ERH\_0020; ARCH\_SW\_ERH\_0021; ARCH\_SW\_ERH\_0022; ARCH\_SW\_ERH\_0023; ARCH\_SW\_ERH\_0024; ARCH\_SW\_ERH\_0025; ARCH\_SW\_ERH\_0026; ARCH\_SW\_ERH\_0027; ARCH\_SW\_ERH\_0028; ARCH\_SW\_ERH\_0029; ARCH\_SW\_ERH\_0030; ARCH\_SW\_ERH\_0031; ARCH\_SW\_ERH\_0032;  ARCH\_SW\_ERH\_0033; ARCH\_SW\_ERH\_0034; ARCH\_SW\_ERH\_0035;  ARCH\_SW\_ERH\_0036; ARCH\_SW\_ERH\_0037; ARCH\_SW\_ERH\_0038;ARCH\_SW\_ERH\_0039; ARCH\_SW\_ERH\_0040; ARCH\_SW\_ERH\_0041; ARCH\_SW\_ERH\_0042; ARCH\_SW\_ERH\_0043; ARCH\_SW\_ERH\_0044; ARCH\_SW\_ERH\_0045;ARCH\_SW\_ERH\_0046; ARCH\_SW\_ERH\_0047;ARCH\_SW\_ERH\_0048; ARCH\_SW\_ERH\_0086; ARCH\_SW\_ERH\_0087; ARCH\_SW\_ERH\_0088; ARCH\_SW\_ERH\_0089; ARCH\_SW\_ERH\_0090;  ARCH\_SW\_ERH\_0092; ARCH\_SW\_ERH\_0157; |
| DSG\_ERH\_00016 | If at least one AEC in a group is currently qualified the qualification bit for this group shall be set true. | Requirement covered by [erh\_UpdateGroup](#_erh_UpdateGroup) function | ARCH\_SW\_ERH\_0074; |
| DSG\_ERH\_00017 | The ERH component shall update the current group counter if at least one AEC in a group has been qualified or deskilled. | Requirement covered by [erh\_UpdateGroup](#_erh_UpdateGroup) function | ARCH\_SW\_ERH\_0074; |
| DSG\_ERH\_00018 | If there are no more qualified AECs in a group, the qualification bit for this group shall be set to false. | Requirement covered by [erh\_UpdateGroup](#_erh_UpdateGroup) function | ARCH\_SW\_ERH\_0074; |
| DSG\_ERH\_00019 | The ERH component shall initialize the AECs counters to 0x00 | Requirement covered by ERH\_Init () function | ARCH\_SW\_ERH\_0060; |
| DSG\_ERH\_00020 | The ERH component shall initialize the status of AECs to 0x00 | Requirement covered by ERH\_Init () function | ARCH\_SW\_ERH\_0060; |
| DSG\_ERH\_00021 | The ERH component shall manage the DTC Status if an AEC passed event occurred depending on the Group of the AEC. | Requirement covered by [erh\_UpdateGroup](#_erh_UpdateGroup) function | ARCH\_SW\_ERH\_0083; ARCH\_SW\_ERH\_0084; ARCH\_SW\_ERH\_0085; ARCH\_SW\_ERH\_0081; |
| DSG\_ERH\_00022 | The ERH component shall manage the DTC Status if an AEC failed event occurred depending on the Group of the AEC. | Requirement covered by [erh\_UpdateGroup](#_erh_UpdateGroup) function | ARCH\_SW\_ERH\_0083; ARCH\_SW\_ERH\_0084; ARCH\_SW\_ERH\_0085; ARCH\_SW\_ERH\_0081; |
| DSG\_ERH\_00025 | If there are no more qualified AECs in a group, the qualification bit for this group shall be set to false. | Requirement covered by [erh\_UpdateGroup](#_erh_UpdateGroup) function | ARCH\_SW\_ERH\_0073; |
| DSG\_ERH\_00026 | The ERH component shall return the DCM\_E\_OK status if the Nvm writing process is successfully completed. | Requirement covered by [ERH\_ResetAllAECStatus](#_ERH_runSetAecEvent) service | ARCH\_SW\_ERH\_0075,ARCH\_SW\_ERH\_0076, |
| DSG\_ERH\_00027 | The ERH component shall return DCM\_E\_NOT\_OK status if the Nvm writing process is unsuccessfully completed. | Requirement covered by [ERH\_ResetAllAECStatus](#_ERH_runSetAecEvent) service | ARCH\_SW\_ERH\_0075,ARCH\_SW\_ERH\_0076, |
| DSG\_ERH\_00028 | The ERH component shall enable conditions for error handler mechanism | Requirement covered by [ERH\_](#_ERH_runSetAecEvent)Init() | ARCH\_SW\_ERH\_0100; |
| DSG\_ERH\_00029 | The ERH component shall provide the AEC status of a given AEC | ERH\_runGetAecStatus() | ARCH\_SW\_ERH\_0064; |
| DSG\_ERH\_00030 | The ERH component shall provide the AECs group status. | ERH\_runGetAecGroupsStatus() | ARCH\_SW\_ERH\_0090; |
| DSG\_ERH\_00031 | Each error code shall have the following properties:  - a single Autoliv Error Code  - an associated Status  - an Aec Qualification Count - an chrono stack priority | Requirement covered by [erh\_ManageAECCommand](#_erh_ManageAECCommand) function | ARCH\_SW\_ERH\_0050; ARCH\_SW\_ERH\_0091; ARCH\_SW\_BSW\_0004; ARCH\_SW\_BSW\_0005 |
| DSG\_ERH\_00032 | The ERH component shall be able to update a given AEC status applying the denouncing mechanism based on previous status of the AEC. | ERH\_runSetAecEvent() | ARCH\_SW\_ERH\_0080; ARCH\_SW\_ERH\_0081; ARCH\_SW\_ERH\_0083; ARCH\_SW\_ERH\_0050; |
| DSG\_ERH\_00033 | The ERH component shall read QUALIFIED status in NVP and update status tables according its value. | Requirement covered by [ERH\_](#_ERH_runSetAecEvent)Init() | ARCH\_SW\_ERH\_0084; ARCH\_SW\_ERH\_0085; |
| DSG\_ERH\_00034 | The ERH component shall update the AEC status block in NVM | ERH\_runSetAecEvent() | ARCH\_SW\_ERH\_0077 |
| DSG\_ERH\_00035 | The ERH component shall be able to read through  *Rte\_Read\_prrIgnitionStatus\_b8TriggerIgnitionStatus* the status(TRUE/FALSE) coming from CIL. | Requirement covered by [ERH\_](#_ERH_runSetAecEvent)cyclic() | ARCH\_SW\_ERH\_0093; ARCH\_SW\_ERH\_0094; ARCH\_SW\_ERH\_0095; ARCH\_SW\_ERH\_0110; ARCH\_SW\_ERH\_0118; |
| DSG\_ERH\_00036 | The ERH component shall be able to start a debounce counter if IGN status is “TRUE” and shall reset the counter if status is set to “FALSE”. | Requirement covered by [ERH\_](#_ERH_runSetAecEvent)cyclic() | ARCH\_SW\_ERH\_0093; ARCH\_SW\_ERH\_0094; ARCH\_SW\_ERH\_0095; ARCH\_SW\_ERH\_0110 |
| DSG\_ERH\_00037 | The debounce counter erh\_u16DebounceCounter shall be used to set a 10s delay before enable Dem Ignition cycle. | Requirement covered by [ERH\_](#_ERH_runSetAecEvent)cyclic() | ARCH\_SW\_ERH\_0093; ARCH\_SW\_ERH\_0094; ARCH\_SW\_ERH\_0095; ARCH\_SW\_ERH\_0110 |
| DSG\_ERH\_00038 | The ERH component shall be able to read through  *Rte\_Read\_prrIgnitionStatus\_b8TriggerIgnitionStatusEnableStorageCondition* the status(TRUE/FALSE) coming from CIL in order to set storage condition. | Requirement covered by [ERH\_](#_ERH_runSetAecEvent)cyclic() | ARCH\_SW\_ERH\_0117; |
| DSG\_ERH\_00039 | If ignition status is TRUE then SC2 shall be SET. | Requirement covered by [ERH\_](#_ERH_runSetAecEvent)cyclic() | ARCH\_SW\_ERH\_0117; |
| DSG\_ERH\_00040 | The ERH component shall get the status of “ControlDTCSetting” through  *Rte\_Mode\_prrControlAECSetting\_DcmControlDtcSetting* . | Requirement covered by [ERH\_](#_ERH_runSetAecEvent)cyclic() | ARCH\_SW\_ERH\_0119; |
| DSG\_ERH\_00041 | If u8DtcState state from “ ControlDTCSetting” is enabled then status for SC1 shall be set. | Requirement covered by [ERH\_](#_ERH_runSetAecEvent)cyclic() | ARCH\_SW\_ERH\_0119; |
| DSG\_ERH\_00042 | The ERH component shall be able to read through  *Rte\_Read\_isrSupBatStatus\_b8SupBatStatus* the status(TRUE/FALSE) coming from CIL in order to set storage condition. | Requirement covered by [ERH\_](#_ERH_runSetAecEvent)cyclic() | ARCH\_SW\_ERH\_0111 |
| DSG\_ERH\_00043 | If u8SupBatStatus is TRUE then for SC4 shall be checked OV&UV autotest status else System OV&UV autotest shall be checked. | Requirement covered by [ERH\_](#_ERH_runSetAecEvent)cyclic() | ARCH\_SW\_ERH\_0111 |
| DSG\_ERH\_00044 | The ERH component shall be able to read through  *Rte\_Call\_pcsGetComMStatus\_GetCurrentComMode* the status(NO\_COM/FULL\_COM) in order to set storage condition. | Requirement covered by [ERH\_](#_ERH_runSetAecEvent)cyclic() | ARCH\_SW\_ERH\_0112 |
| DSG\_ERH\_00045 | If ComM\_Status status is FULL\_COM then SC8 shall be set | Requirement covered by [ERH\_](#_ERH_runSetAecEvent)cyclic() | ARCH\_SW\_ERH\_0112 |
| DSG\_ERH\_00046 | The ERH component shall be able to read through  *Rte\_Read\_prrEngStartStatus\_b8EngStartStatus* the status(TRUE/FALSE) coming from CIL in order to set storage condition. | Requirement covered by [ERH\_](#_ERH_runSetAecEvent)cyclic() | ARCH\_SW\_ERH\_0113 |
| DSG\_ERH\_00047 | If u8EngStartStatus status is TRUE then SC11 shall be set | Requirement covered by [ERH\_](#_ERH_runSetAecEvent)cyclic() | ARCH\_SW\_ERH\_0113 |
| DSG\_ERH\_00048 | The ERH component shall be able to read through  *Rte\_Read\_prrProdModeStatus\_b8TriggerProdModeStatus* the status(TRUE/FALSE) coming from CIL in order to set storage condition. | Requirement covered by [ERH\_](#_ERH_runSetAecEvent)cyclic() | ARCH\_SW\_ERH\_0114 |
| DSG\_ERH\_00049 | If u8ProductionMode status is TRUE then SC12 shall be set | Requirement covered by [ERH\_](#_ERH_runSetAecEvent)cyclic() | ARCH\_SW\_ERH\_0114 |
| DSG\_ERH\_00050 | The ERH component shall be able to read through  *Rte\_Read\_prrTransportationModeStatus\_b8TransportationModeStatus* the status(TRUE/FALSE) coming from CIL in order to set storage condition. | Requirement covered by [ERH\_](#_ERH_runSetAecEvent)cyclic() | ARCH\_SW\_ERH\_0115 |
| DSG\_ERH\_00051 | If u8TransportationMode status is TRUE then SC5 shall be set | Requirement covered by [ERH\_](#_ERH_runSetAecEvent)cyclic() | ARCH\_SW\_ERH\_0115 |
| DSG\_ERH\_00052 | The ERH component shall be able to read through  *Rte\_Call\_pcsAutotestServices\_GetTestResult* the status(TRUE/FALSE) coming from CIL in order to set storage condition. | Requirement covered by [ERH\_](#_ERH_runSetAecEvent)cyclic() | ARCH\_SW\_ERH\_0116 |
| DSG\_ERH\_00053 | If autotest status for UV or OV is passed then SC3 shall be set. | Requirement covered by [ERH\_](#_ERH_runSetAecEvent)cyclic() | ARCH\_SW\_ERH\_0116 |
| DSG\_ERH\_00054 | If ECU Level storage condition group is not set then the status for the following AECs shall be inhibited:   * ERH\_KU8\_ENABLE\_OUT\_OF\_ORDER\_AEC\_CFG, * ERH\_KU8\_MOSFET\_HIGH\_SC\_AEC\_CFG, * ERH\_KU8\_MOSFET\_LOW\_SC\_AEC\_CFG, * ERH\_KU8\_MOTOR\_DISCONNECTED\_AEC\_CFG, * ERH\_KU8\_HALL\_SENSOR\_AEC\_CFG, * ERH\_KU8\_TEMPERATURE\_SENSOR\_FAILURE\_AEC\_CFG, * ERH\_KU8\_EXT\_WDG\_OUT\_OF\_ORDER\_AEC\_CFG, * ERH\_KU8\_HW\_SELF\_PROTECTION\_AEC\_CFG, * ERH\_KU8\_SW\_SELF\_PROTECTION\_AEC\_CFG, * ERH\_KU8\_END\_OF\_LIFE\_COMFORT\_AEC\_CFG, * ERH\_KU8\_END\_OF\_LIFE\_LOW\_AEC\_CFG, * ERH\_KU8\_END\_OF\_LIFE\_HIGH\_AEC\_CFG, * ERH\_KU8\_END\_OF\_LIFE\_MAX\_FORCE\_AEC\_CFG, * ERH\_KU8\_STEERING\_CONFIG\_AEC\_CFG, * ERH\_KU8\_VARIANT\_CODING\_AEC\_CFG | Requirement covered by [ERH\_](#_ERH_runSetAecEvent)cyclic() | ARCH\_SW\_ERH\_0096 |
| DSG\_ERH\_00055 | If Network Communication storage condition group is not set then the status for the following AECs shall be inhibited:   * ERH\_KU8\_TIMEOUT\_PRESAFE\_AEC\_CFG, * ERH\_KU8\_TIMEOUT\_BUCKLE\_AEC\_CFG, * ERH\_KU8\_IMPLAUSIBLE\_DATA\_PRESAFE\_AEC\_CFG, * ERH\_KU8\_IMPLAUSIBLE\_DATA\_IGNITION\_AEC\_CFG, * ERH\_KU8\_IMPLAUSIBLE\_DATA\_BUCKLE\_AEC\_CFG * ERH\_KU8\_IMPLAUSIBLE\_DATA\_ORC\_AEC\_CFG, * ERH\_KU8\_IMPLAUSIBLE\_DATA\_POWERTRAIN\_RDY\_AEC\_CFG, * ERH\_KU8\_IMPLAUSIBLE\_DATA\_POWERTRAIN\_DRV\_AEC\_CFG, * ERH\_KU8\_TIMEOUT\_BLTHD\_R\_AEC\_CFG, * ERH\_KU8\_TIMEOUT\_BLTHD\_L\_AEC\_CFG, * ERH\_KU8\_TIMEOUT\_POWERTRAIN\_AEC\_CFG, * ERH\_KU8\_TIMEOUT\_ODOSPEEDOMETER\_AEC\_CFG, * ERH\_KU8\_IMPLAUSIBLE\_DATA\_API\_AEC\_CFG, * ERH\_KU8\_TIMEOUT\_BELT\_ADJ\_AEC\_CFG, * ERH\_KU8\_TIMEOUT\_API | Requirement covered by [ERH\_](#_ERH_runSetAecEvent)cyclic() | ARCH\_SW\_ERH\_0097 |
| DSG\_ERH\_00056 | If Network Communication without Clam 15 storage condition group is not set then the status for the following AECs shall be inhibited:   * ERH\_KU8\_TIMEOUT\_IGNITION\_AEC\_CFG | Requirement covered by [ERH\_](#_ERH_runSetAecEvent)cyclic() | ARCH\_SW\_ERH\_0098 |
| DSG\_ERH\_00057 | If Power Distribution storage condition group is not set then the status for the following AECs shall be inhibited:   * ERH\_KU8\_OVO\_POWER\_AEC\_CFG, * ERH\_KU8\_UVO\_POWER\_AEC\_CFG, * ERH\_KU8\_UVO\_TENSIONING\_POWER\_AEC\_CFG, * ERH\_KU8\_OVO\_TENSIONING\_POWER\_AEC\_CFG, * ERH\_KU8\_SYSTEM\_OV\_AEC\_CFG, * ERH\_KU8\_SYSTEM\_UV\_AEC\_CFG | Requirement covered by [ERH\_](#_ERH_runSetAecEvent)cyclic() | ARCH\_SW\_ERH\_0099 |
| DSG\_ERH\_00058 | If SC1 is not set then yhe following AECs shall be inhibited:   * ERH\_KU8\_SPI\_ERROR\_AEC\_CFG, * ERH\_KU8\_HIGH\_SIDE\_SWITCH\_AEC\_CFG, * ERH\_KU8\_MOSFET\_OC\_AEC\_CFG, * ERH\_KU8\_WARM\_RESET\_AEC\_CFG, * ERH\_KU8\_MOTOR\_CURRENT\_AEC\_CFG, * ERH\_KU8\_MOTOR\_BLOCKED\_AEC\_CFG, * ERH\_KU8\_MOTOR\_ORDER\_AEC\_CFG, * ERH\_KU8\_MOTOR\_IN\_SC\_AEC\_CFG | Requirement covered by [ERH\_](#_ERH_runSetAecEvent)cyclic() | ARCH\_SW\_ERH\_0150 |
| DSG\_ERH\_00059 | If   * SC1: Control DTC settings * SC3: Local Voltage (KL30 input) * SC11: Vehicle Startup   Are SET then "ECU level" group shall be SET | Requirement covered by [ERH\_](#_ERH_runSetAecEvent)cyclic() | ARCH\_SW\_ERH\_0151 |
| DSG\_ERH\_00060 | If   * SC1: Control DTC settings * SC2: Ignition Status * SC4: System Voltage * SC5: Transportation mode * SC8: Communication Status * SC10: De-bounce Timer network * SC11: Vehicle Startup * SC12: production mode   Are SET then "Network Communication" group shall be SET | Requirement covered by [ERH\_](#_ERH_runSetAecEvent)cyclic() | ARCH\_SW\_ERH\_0152 |
| DSG\_ERH\_00061 | If   * SC1: Control DTC settings * SC4: System Voltage * SC5: Transportation mode * SC8: Communication Status * SC10: De-bounce Timer Network w/o KL15 * SC11: Vehicle Startup * SC12: production mode   Are SET then "Network Communication - Timeout monitoring without Clamp 15" group shall be SET | Requirement covered by [ERH\_](#_ERH_runSetAecEvent)cyclic() | ARCH\_SW\_ERH\_0153 |
| DSG\_ERH\_00062 | If   * SC1: Control DTC settings * SC2: Ignition Status * SC5: Transportation mode * SC10: De-bounce Timer Power * SC11: Vehicle Startup * SC12: production mode   Are SET then "Power Distribution" group shall be SET | Requirement covered by [ERH\_](#_ERH_runSetAecEvent)cyclic() | ARCH\_SW\_ERH\_0154 |
| DSG\_ERH\_00063 | A debounce timer of 5s shall be applicable for each of the following storage condition groups  Power Distribution, Network Communication without Clam 15 and Network Communication | Requirement covered by [ERH\_](#_ERH_runSetAecEvent)cyclic() | ARCH\_SW\_ERH\_0154 |
| DSG\_ERH\_00064 | For Power Distribution SC group if all SCs (except SC10) are set a debounce timer of 5s shall start. Only if the 5s are elapsed the SC group shall be SET. | Requirement covered by [ERH\_](#_ERH_runSetAecEvent)cyclic() | ARCH\_SW\_ERH\_0156 |
| DSG\_ERH\_00065 | For Network Communication without Clam 15 SC group if all SCs (except SC10) are set a debounce timer of 5s shall start. Only if the 5s are elapsed the SC group shall be SET. | Requirement covered by [ERH\_](#_ERH_runSetAecEvent)cyclic() | ARCH\_SW\_ERH\_0156 |
| DSG\_ERH\_00066 | For Network Communication SC group if all SCs (except SC10) are set a debounce timer of 5s shall start. Only if the 5s are elapsed the SC group shall be SET. | Requirement covered by [ERH\_](#_ERH_runSetAecEvent)cyclic() | ARCH\_SW\_ERH\_0156 |
| DSG\_ERH\_00067 | Dem\_GetEventUdsStatus is called to read status for all DTCs | Requirement covered by [ERH\_](#_ERH_runSetAecEvent)cyclic() | ARCH\_SW\_ERH\_0120 |
| DSG\_ERH\_00068 | If DTC status has bit 3 set then Rte\_Write\_ERH\_AC\_ErrorHandler\_psrDTCConfirmed\_b8DTCConfirmed shall write TRUE for DIA | Requirement covered by [ERH\_](#_ERH_runSetAecEvent)cyclic() | ARCH\_SW\_ERH\_0121 |
| DSG\_ERH\_00069 | erh\_UpdateGroup shall be called at init in order to update the status of the DTCs after a reset. (Avoinding unwanted 0x10 status after reset) | Requirement covered by [ERH\_](#_ERH_runSetAecEvent)Init() | ARCH\_SW\_ERH\_0158 |
| DSG\_ERH\_00070 | When update group occurs erh module shall send snapshot data AEC to Dem module | Requirement covered by ERH\_runSetAecEvent() | ARCH\_SW\_ERH\_0159 |

# Features

## Services

### ERH\_cyclic

|  |  |  |
| --- | --- | --- |
| Object | | |
| This service shall be used to set the debounce counter for Ignistion cycle start  This service shall be called periodically at 10ms. | | |
| **Prototype** | | |
| void ERH\_cycle(void) | | |
| **Input parameters** | | |
| Name | Type | Description |
| NA | NA | NA |
| **Output parameters** | | |
| Name | Type | Description |
| NA | NA | NA |
| **Return value** | | |
| Type | Description | |
| NA | void | |
| **Dynamic aspect** | | |
| Who(callers) | Description | |
| \* | \* | |
| **Static aspect** | | |
| NA | | |
| **Constrains** | | |
| NA | | |



Figure ERH\_cyclic

### ERH\_Init

|  |  |  |
| --- | --- | --- |
| Object | | |
| This service shall initialize all AEC status and associated counters to default values. The “EVER QUALIFIED” status should be updated according to its value from NVM.  This service shall be called only once during MCU initialization phase. | | |
| **Prototype** | | |
| void ERH\_Init(void) | | |
| **Input parameters** | | |
| Name | Type | Description |
| NA | NA | NA |
| **Output parameters** | | |
| Name | Type | Description |
| NA | NA | NA |
| **Return value** | | |
| Type | Description | |
| NA | void | |
| **Dynamic aspect** | | |
| Who(callers) | Description | |
| \* | \* | |
| **Static aspect** | | |
| NA | | |
| **Constrains** | | |
| NA | | |



Figure : ERH\_Init

### erh\_CheckAecId

|  |  |  |
| --- | --- | --- |
| **Object** | | |
| This function shall check the validity of the current AEC identifier. | | |
| **Prototype** | | |
| LOCAL Boolean **erh\_CheckAecId** (**const** uint8 **cu8AecId,** uint8 \*const **cpu8Index**, erh\_stAECConfigurationType\* const **cpstAecData**) | | |
| **Input Parameters** | | |
| Name | Type | Description |
| Cu8AecId | uint8 | The AEC identifier which must be checked |
| cpu8Index | uint8\* | The output array index |
| cpstAecData | stAECConfigurationType\* | The address of the AEC data structure which has the qualification/deskill thresholds and the AEC group data |
| **Output Parameters:** | | |
| Name | Type | Description |
| cpu8Index | uint8\* | The address where shall be stored the output index |
| cpstAecData | stAECConfigurationType\* | The address of the AEC data structure where are stored the qualification/deskill thresholds and the group status |
| **Return value** | | |
| Type | Description | |
| Boolean | - B\_TRUE - if the given AEC identifier is in range: 0x0 - 0x37  - B\_FALSE - if the given AEC identifier is out of range | |
| **Dynamic aspect** | | |
| Who (callers) | Description | |
| ERH component | ERH\_runGetAecStatus – interface used to get the AEC status to DemIf module  ERH\_runSetAecEvent – interface used to set the AEC status  ERH\_ResetAllAECStatus – interface used to clear all AEC status | |
| **Static aspect** | | |
| This function shall check the validity of the current AEC identifier and if it is valid, this function shall convert this identifier to an index used to access each AEC status.  This function shall send by the output parameters the converted index and the corresponding debouncing threshold only if the AEC identifier is valid. If the AEC identifier is out of range, this function shall return B\_FALSE (0x55) value. | | |
| **Constraints** | | |
| The AEC identifier should be in the following range to be valid: | | |

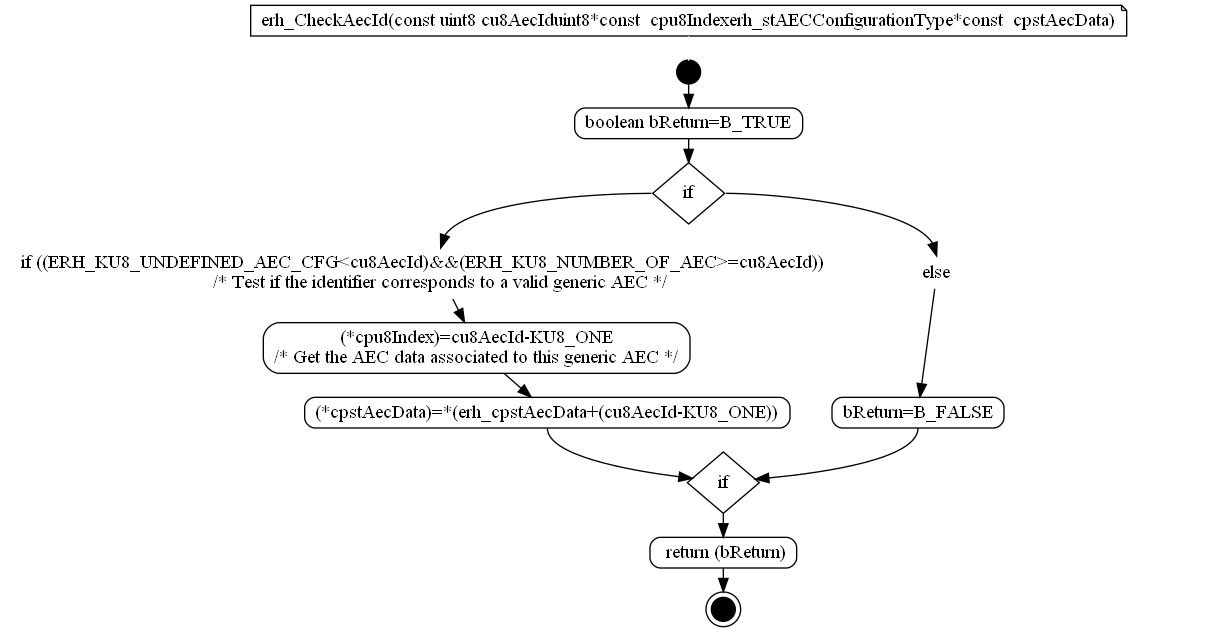


Figure : erh\_CheckAecId

### erh\_ManageFailedCounterEvent

|  |  |  |
| --- | --- | --- |
| **Object** | | |
| This function shall manage the current AEC counter on a failed event and update the AEC status | | |
| **Prototype** | | |
| LOCAL void **erh\_ManageFailedCounterEvent(**const uint8 **cu8Index**, const erh\_stAECConfigurationType\* const **cpstAecData)** | | |
| **Input Parameters** | | |
| Name | Type | Description |
| cu8Index | Uint8 | AEC ID related to the passed event |
| CpstAecData | stAECConfigurationType\* | Pointer to the corresponding AEC configuration |
| **Output Parameters :** | | |
| Name | Type | Description |
| NA | NA | NA |
| **Return value** | | |
| Type | Description | |
| NA | NA | |
| **Dynamic aspect** | | |
| Who (callers) | Description | |
| ERH component | **ERH\_runSetAecEvent** | |
| **Static aspect** | | |
| This function firstly checks if the AEC qualification threshold is not inhibited for the current AEC.  In the second step if the status of the current AEC is not qualified the debouncing counter shall be incremented and if the qualification limit has been reached, it shall be set to deskill value. The AEC status and group status shall be updated for the current AEC and stored in NVM.  In the last step if the AEC qualification threshold is inhibited the current AEC Status shall be updated with “AEC ever tested” status. | | |
| **Constraints** | | |
| None | | |



Figure : erh\_ManageFailedCounterEvent

### erh\_ManagePassedCounterEvent

|  |  |  |
| --- | --- | --- |
| **Object** | | |
| This function shall manage the current AEC debounce counter and update the current AEC status on a Passed event | | |
| **Prototype** | | |
| LOCAL void **erh\_ManagePassedCounterEvent** (const uint8 **cu8Index,** const erh\_stAECConfigurationType \* const **cpstAecData**) | | |
| **Input Parameters** | | |
| Name | Type | Description |
| cu8Index | Uint8 | AEC ID related to the passed event |
| CpstAecData | stAECConfigurationType\* | Pointer to the corresponding AEC configuration |
| **Output Parameters:** | | |
| Name | Type | Description |
| NA | NA | NA |
| **Return value** | | |
| Type | Description | |
| NA | NA | |
| **Dynamic aspect** | | |
| Who (callers) | Description | |
| ERH component | **ERH\_runSetAecEvent** | |
| **Static aspect** | | |
| This function firstly checks if deskill threshold is set for the current AEC and debouncing counter was incremented.  In the second step the current debouncing counter shall be decremented with one step and the AEC status shall be updated. The AEC status shall be set to “in progress state” if the fault threshold was reached during the qualification phase otherwise “in progress status” shall be cleared.  In the next step if the current debouncing counter reaches the zero value the AEC status for current AEC shall be updated.  At each step, the AEC local status shall be stored in NVM. | | |
| **Constraints** | | |
| None | | |



Figure : erh\_ManagePassedCounterEvent

### erh\_ManageAECCommand

|  |  |  |
| --- | --- | --- |
| **Object** | | |
| This function shall execute the request received through the input parameter “cu8Command”. | | |
| **Prototype** | | |
| LOCAL void **erh\_ManageAECCommand**(const u8AecCommandType **cu8Command,** const uint8 **cu8Index,** const erh\_stAECConfigurationType\* const **cpstAecData**) | | |
| **Input Parameters** | | |
| Name | Type | Description |
| cu8Command | u8AecCommandType | The received request |
| cu8Index | uint8 | AEC Index in the configuration table |
| cpstAecData | erh\_stAECConfigurationType\* | Pointer to the corresponding AEC data structure where are stored the qualification/deskill thresholds and the AEC group status |
| **Output Parameters:** | | |
| Name | Type | Description |
| cpstAecData | erh\_stAECConfigurationType\* | The address of the AEC data structure where are stored the qualification/deskill thresholds and the AEC group status |
| **Return value** | | |
| Type | Description | |
| NA | NA | |
| **Dynamic aspect** | | |
| Who (callers) | Description | |
| ERH component | **ERH\_runSetAecEvent –** called to update the AEC status | |
| **Static aspect** | | |
| This function manages 4 cases. In the first two cases if the input parameter “cu8Command” is ‘FAILED’ (0x00) or ‘PASSED’ (0x01) the AEC qualification counter and the AEC status shall be updated.  In another case if the input parameter “cu8Command” is ‘AEC\_RST\_COUNTER’ (0x02) the following actions shall be performed:  - the qualification counter shall be reset,  - only “ever tested” and “ever qualified” status shall be kept, the “qualified” and “in progress” AEC status shall be cleared,  - all AEC parameters will be stored in NVP  In the last case if the input parameter cu8Command will be ‘AEC\_RST\_ALL’ (0x03) the following actions will be performed:  - the qualification counter shall be cleared  - each AEC status shall be cleared except ‘ever tested bit’  - each AEC status shall be stored in NVP | | |
| **Constraints** | | |
| None | | |



Figure : erh\_ManageAECCommand

### ERH\_runGetAecStatus

|  |  |  |
| --- | --- | --- |
| Object | | |
| This service shall supply the AEC status to DemIf module if the AEC identifier is valid.  f the AEC identifier transmitted by the input parameter “u8AecIdentifier” is between 0x00 and 0x33 this service shall send by the output parameter “pu8AecStatus” the current AEC status, and if the AEC identifier is out of range the transmitted AEC status shall be set to "UNPLAUSIBLE\_STATUS". | | |
| **Prototype** | | |
| void ERH\_runGetAecStatus(u8AecIdentifierType u8AecIdentifier, u8AecStatusType \* pu8AecStatus) | | |
| **Input parameters** | | |
| Name | Type | Description |
| u8AecIdentifier | u8AecIdentifierType | The AEC Identifier |
| **Output parameters** | | |
| Name | Type | Description |
| pu8AecStatus | u8AecStatusType | The value for AEC status |
| **Return value** | | |
| Type | Description | |
| NA | void | |
| **Dynamic aspect** | | |
| Who(callers) | Description | |
| \* | \* | |
| **Static aspect** | | |
| NA | | |
| **Constrains** | | |
| NA | | |

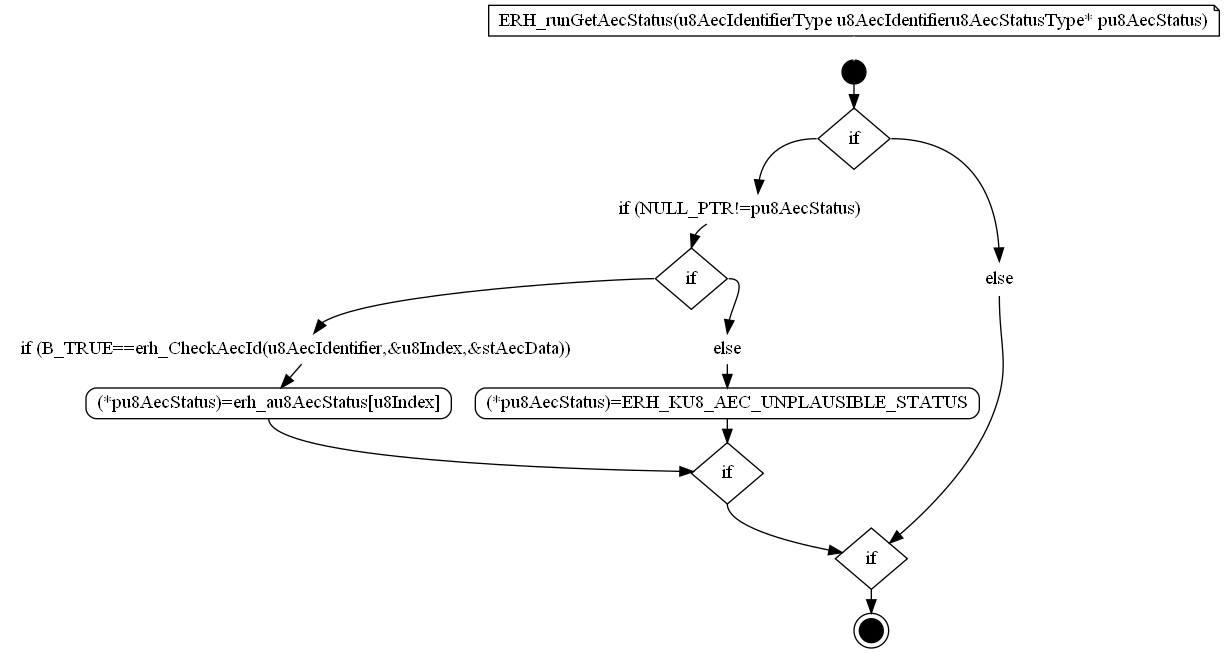


Figure : ERH\_runGetAecStatus

### ERH\_runGetAecGroupStatus

|  |  |  |
| --- | --- | --- |
| Object | | |
| This service shall provide the the current AEC groups status bitfield. | | |
| **Prototype** | | |
| void ERH\_runGetAecGroupStatus(u8AecIdentifierType u8AecIdentifier, u8AecStatusType \* pu8AecStatus) | | |
| **Input parameters** | | |
| Name | Type | Description |
| NA |  |  |
| **Output parameters** | | |
| Name | Type | Description |
| pu32AecGroupsStatus | u32AecGroupsType | The value for AECs group status |
| **Return value** | | |
| Type | Description | |
| NA | void | |
| **Dynamic aspect** | | |
| Who(callers) | Description | |
| \* | \* | |
| **Static aspect** | | |
| NA | | |
| **Constrains** | | |
| NA | | |

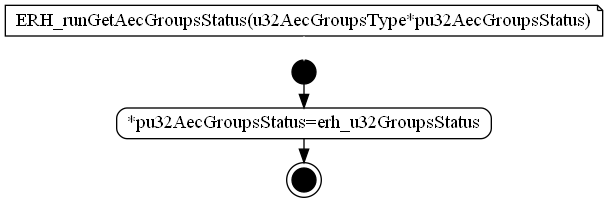


Figure : ERH\_runGetAecGroupsStatus

### ERH\_ResetAllAECStatus

|  |  |  |
| --- | --- | --- |
| Object | | |
| This service shall reset all AECs status. | | |
| **Prototype** | | |
| Std\_ReturnType ERH\_ResetAllAECStatus(void) | | |
| **Input parameters** | | |
| Name | Type | Description |
| NA | NA | NA |
| **Output parameters** | | |
| Name | Type | Description |
| NA | NA | NA |
| **Return value** | | |
| Type | Description | |
| Std\_ReturnType Result | NA | |
| **Dynamic aspect** | | |
| Who(callers) | Description | |
| \* | \* | |
| **Static aspect** | | |
| NA | | |
| **Constrains** | | |
| NA | | |



Figure : ERH\_ResetAllAECStatus

### erh\_UpdateGroup

|  |  |  |
| --- | --- | --- |
| **Object** | | |
| This function shall update the group status and the number of qualified AECs in the group. | | |
| **Prototype** | | |
| LOCAL void **erh\_UpdateGroup**(const uint8 **cu8AecId,** const erh\_stAECConfigurationType\* const **cpstAecData**) | | |
| **Input Parameters** | | |
| Name | Type | Description |
| cu8AecId | uint8 | The AEC identifier to update the right group |
| cpstAecData | erh\_stAECConfigurationType\* | Pointer to the corresponding AEC configuration |
| **Output Parameters:** | | |
| Name | Type | Description |
| NA | NA | NA |
| **Return value** | | |
| Type | Description | |
| NA | NA | |
| **Dynamic aspect** | | |
| Who (callers) | Description | |
| ERH component | **ERH\_Init -** interface used to initialize the all internal data to default value  **erh\_ManageAECCommand -** used to execute the received request  **erh\_ManageFailedCounterEvent -** used to manage the AEC counter on a failed event  **erh\_ManagePassedCounterEvent -** used to manage the AEC counter on a passed event  **ERH\_ResetAllAECStatus -** interface called to clear all AEC status | |
| **Static aspect** | | |
| The group status shall be set to “qualified” if at least one AEC in the current group is qualified, and if there is no more any qualified AEC in the group, the group status shall be set to disqualified. | | |
| **Constraints** | | |
| None | | |



Figure : erh\_UpdateGroup

### ERH\_runSetAecEvent

|  |  |  |
| --- | --- | --- |
| Object | | |
| o Basing of the previous AEC status. | | |
| **Prototype** | | |
| void ERH\_runSetAecEvent (u8AecIdentifierType u8AecIdentifier, u8AecCommandType u8Command) | | |
| **Input parameters** | | |
| Name | Type | Description |
| u8AecIdentifier | u8AecIdentifierType | u8AecIdentifierType u8AecIdentifier: the AEC identifier u8AecCommandType u8Command: KU8\_AEC\_FAILED: Update the error status regarding a test failed result KU8\_AEC\_PASSED : Update the error status regarding a test passed result |
| u8Command | u8AecCommandType | u8AecCommandType cu8Command: uint8\* cpu8Index: AEC Index in the configuration table stAECConfigurationType\* cpstAecData: Pointer to the corresponding AEC data structure where are stored the qualification/deskill thresholds and the AEC group status |
| **Output parameters** | | |
| Name | Type | Description |
| NA | NA | NA |
| **Return value** | | |
| Type | Description | |
| NA | None. | |
| **Dynamic aspect** | | |
| Who(callers) | Description | |
| \* | \* | |
| **Static aspect** | | |
| \* | | |
| **Constrains** | | |
|  | | |



Figure 10: ERH\_runSetAecEvent

## Types

### erh\_stAECConfigurationType

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Field Type** | **Field description** |
| u16AECDeskillThrs | uint16 | To store the AEC deskill threshold |
| u16AECQualificationThrs | uint16 | To store the AEC qualification threshold |
| u8AECGroup | uint8 | To store the AEC group (HW, battery, motor etc.) |
| u8ReservedByte1 | uint8 | Reserved Byte for future update |
| u8ReservedByte2 | uint8 | Reserved Byte for future update |
| u8ReservedByte3 | uint8 | Reserved Byte for future update |

## Variabiles

### erh\_au16AecCounter

|  |  |  |
| --- | --- | --- |
| Type | Value |  |
| uint16[] | NA | |
| **Description** | | |
| Array of current debouncing counters for AEC’s | | |
| **Definition** | | |
| LOCAL uint16 erh\_au16AecCounter[ERH\_KU8\_NUMBER\_OF\_AEC]; | | |

### erh\_u32GroupsStatus

|  |  |  |
| --- | --- | --- |
| Type | Value |  |
| Uint32 | KU32\_ZERO | |
| **Description** | | |
| Status of AEC group | | |
| **Definition** | | |
| LOCAL uint16 erh\_u16GroupsStatus; | | |

### erh\_au8AecStatus

|  |  |  |
| --- | --- | --- |
| Type | Value |  |
| uint8 | NA | |
| **Description** | | |
| Array of AEC status | | |
| **Definition** | | |
| LOCAL uint8 erh\_au8AecStatus[ERH\_KU8\_NUMBER\_OF\_AEC]; | | |

### erh\_u8PendingEventAddIndex

|  |  |  |
| --- | --- | --- |
| Type | Value |  |
| uint8 | KU8\_ZERO | |
| **Description** | | |
| Index pointing to the free spot of the pending event list | | |
| **Definition** | | |
| LOCAL uint8 erh\_u8PendingEventAddIndex; | | |

### erh\_u8PendingEventRunIndex

|  |  |  |
| --- | --- | --- |
| Type | Value |  |
| uint8 | KU8\_ZERO | |
| **Description** | | |
| Index pointing to the current item to process in the pending list | | |
| **Definition** | | |
| LOCAL uint8 erh\_u8PendingEventRunIndex; | | |

### erh\_au8PendingEventCommand

|  |  |  |
| --- | --- | --- |
| Type | Value |  |
| u8AecCommandType | NA | |
| **Description** | | |
| Array of u8AecCommandType | | |
| Definition | | |
| LOCAL u8AecCommandType erh\_au8PendingEventCommand[KU8\_MAX\_PENDING\_EVENT]; | | |

### erh\_au8PendingEventAecIdentifier

|  |  |  |
| --- | --- | --- |
| Type | Value |  |
| u8AecIdentifierType | NA | |
| **Description** | | |
| Array of u8AecIdentifierType | | |
| **Definition** | | |
| LOCAL u8AecIdentifierType erh\_au8PendingEventAecIdentifier[KU8\_MAX\_PENDING\_EVENT]; | | |

### erh\_u8StAECWriteStatusResult

|  |  |  |
| --- | --- | --- |
| Type | Value |  |
| uint8 | KU8\_ZERO | |
| **Description** | | |
| Result of the AEC status write process | | |
| Definition | | |
| LOCAL uint8 erh\_u8StAECWriteStatusResult; | | |

### erh\_bNvmRecordRequested

|  |  |  |
| --- | --- | --- |
| Type | Value |  |
| boolean | B\_FALSE | |
| **Description** | | |
| Switch used to indicate when the condition is met to store data in NVP | | |
| Definition | | |
| LOCAL boolean erh\_bNvmRecordRequested; | | |

### erh\_u8OtherGroupValue

|  |  |  |
| --- | --- | --- |
| Type | Value |  |
| uint8 | NA | |
| **Description** | | |
| The number of qualified AEC in the group | | |
| Definition | | |
| LOCAL uint8 erh\_u8OtherGroupValue; | | |

### erh\_cpstAecData

|  |  |  |
| --- | --- | --- |
| Type | Value |  |
| erh\_stAECConfigurationType | NA | |
| **Description** | | |
| Indirection pointer to access AEC configuration data | | |
| Definition | | |
| LOCAL const erh\_stAECConfigurationType \* const erh\_cpstAecData | | |

### ERH\_astAECsDefinitions

|  |  |  |
| --- | --- | --- |
| Type | Value |  |
| uint8 | NA | |
| **Description** | | |
| Array of AECs configuration | | |
| Definition | | |
| const extern uint8 ERH\_astAECsDefinitions[]; | | |

### erh\_au32GroupCounter

|  |  |  |
| --- | --- | --- |
| Type | Value |  |
| uint32 | NA | |
| **Description** | | |
| Array of counters for AEC group. | | |
| **Definition** | | |
| LOCAL uint32 erh\_au32GroupCounter[((uint8) 30)] | | |
| **Remarks** | | |
| None. | | |

### erh\_u16DebounceCounter

|  |  |  |
| --- | --- | --- |
| Type | Value |  |
| Uint16 | NA | |
| **Description** | | |
| Counter for debounce before start of Ign cycle | | |
| **Definition** | | |
| LOCAL uint16 erh\_au16DebounceCounter [((uint16) 0)] | | |
| **Remarks** | | |
| None. | | |

### ERH\_u16AECsForControlDTC\_StorageGroup[KU8\_EIGHT]

|  |  |  |
| --- | --- | --- |
| Type | Value |  |
| uint16 | NA | |
| **Description** | | |
| Initial value: | | |
| **Definition** | | |
| LOCAL uint16 ERH\_u16AECsForControlDTC\_StorageGroup[KU8\_EIGHT] | | |
| **= { ERH\_KU8\_SPI\_ERROR\_AEC\_CFG,** | | |
| ERH\_KU8\_HIGH\_SIDE\_SWITCH\_AEC\_CFG, ERH\_KU8\_MOSFET\_OC\_AEC\_CFG, ERH\_KU8\_WARM\_RESET\_AEC\_CFG, ERH\_KU8\_MOTOR\_CURRENT\_AEC\_CFG, ERH\_KU8\_MOTOR\_BLOCKED\_AEC\_CFG, ERH\_KU8\_MOTOR\_ORDER\_AEC\_CFG, ERH\_KU8\_MOTOR\_IN\_SC\_AEC\_CFG} | | |

### ERH\_u16AECsForECULevel\_StorageGroup[KU8\_FIFTEEN]

|  |  |  |
| --- | --- | --- |
| Type | Value |  |
| uint16 | NA | |
| **Description** | | |
| Initial value: | | |
| **Definition** | | |
| LOCAL uint16 ERH\_u16AECsForECULevel\_StorageGroup[KU8\_FIFTEEN] | | |
| **= {ERH\_KU8\_ENABLE\_OUT\_OF\_ORDER\_AEC\_CFG,** | | |
| ERH\_KU8\_MOSFET\_HIGH\_SC\_AEC\_CFG, ERH\_KU8\_MOSFET\_LOW\_SC\_AEC\_CFG, ERH\_KU8\_MOTOR\_DISCONNECTED\_AEC\_CFG, ERH\_KU8\_HALL\_SENSOR\_AEC\_CFG, ERH\_KU8\_TEMPERATURE\_SENSOR\_FAILURE\_AEC\_CFG, ERH\_KU8\_EXT\_WDG\_OUT\_OF\_ORDER\_AEC\_CFG, ERH\_KU8\_HW\_SELF\_PROTECTION\_AEC\_CFG, ERH\_KU8\_SW\_SELF\_PROTECTION\_AEC\_CFG, ERH\_KU8\_END\_OF\_LIFE\_COMFORT\_AEC\_CFG, ERH\_KU8\_END\_OF\_LIFE\_LOW\_AEC\_CFG, ERH\_KU8\_END\_OF\_LIFE\_HIGH\_AEC\_CFG, ERH\_KU8\_END\_OF\_LIFE\_MAX\_FORCE\_AEC\_CFG, ERH\_KU8\_STEERING\_CONFIG\_AEC\_CFG, ERH\_KU8\_VARIANT\_CODING\_AEC\_CFG} | | |

### ERH\_u16AECsForNetworkCom\_StorageGroup[KU8\_FIFTEEN]

|  |  |  |
| --- | --- | --- |
| Type | Value |  |
| uint16 | NA | |
| **Description** | | |
| Initial value: | | |
| **Definition** | | |
| LOCAL uint16 ERH\_u16AECsForNetworkCom\_StorageGroup[KU8\_FIFTEEN] | | |
| **= {ERH\_KU8\_TIMEOUT\_PRESAFE\_AEC\_CFG,** | | |
| ERH\_KU8\_TIMEOUT\_BUCKLE\_AEC\_CFG, ERH\_KU8\_IMPLAUSIBLE\_DATA\_PRESAFE\_AEC\_CFG, ERH\_KU8\_IMPLAUSIBLE\_DATA\_IGNITION\_AEC\_CFG, ERH\_KU8\_IMPLAUSIBLE\_DATA\_BUCKLE\_AEC\_CFG, ERH\_KU8\_IMPLAUSIBLE\_DATA\_ORC\_AEC\_CFG, ERH\_KU8\_IMPLAUSIBLE\_DATA\_POWERTRAIN\_RDY\_AEC\_CFG, ERH\_KU8\_IMPLAUSIBLE\_DATA\_POWERTRAIN\_DRV\_AEC\_CFG, ERH\_KU8\_TIMEOUT\_BLTHD\_R\_AEC\_CFG, ERH\_KU8\_TIMEOUT\_BLTHD\_L\_AEC\_CFG, ERH\_KU8\_TIMEOUT\_POWERTRAIN\_AEC\_CFG, ERH\_KU8\_TIMEOUT\_ODOSPEEDOMETER\_AEC\_CFG, ERH\_KU8\_IMPLAUSIBLE\_DATA\_API\_AEC\_CFG, ERH\_KU8\_TIMEOUT\_BELT\_ADJ\_AEC\_CFG, ERH\_KU8\_TIMEOUT\_API} | | |

### ERH\_u16AECsForPowerDistribution\_StorageGroup[KU8\_SIX]

|  |  |  |
| --- | --- | --- |
| Type | Value |  |
| uint16 | NA | |
| **Description** | | |
| Initial value: | | |
| **Definition** | | |
| LOCAL uint16 ERH\_u16AECsForPowerDistribution\_StorageGroup[KU8\_SIX] | | |
| **= {ERH\_KU8\_OVO\_POWER\_AEC\_CFG,** | | |
| ERH\_KU8\_UVO\_POWER\_AEC\_CFG, ERH\_KU8\_UVO\_TENSIONING\_POWER\_AEC\_CFG, ERH\_KU8\_OVO\_TENSIONING\_POWER\_AEC\_CFG, ERH\_KU8\_SYSTEM\_OV\_AEC\_CFG, ERH\_KU8\_SYSTEM\_UV\_AEC\_CFG} | | |

### ERH\_u16DebounceCounter\_NetworkComunication

|  |  |  |
| --- | --- | --- |
| Type | Value |  |
| uint16 | KU16\_ZERO | |
| **Description** | | |
| The Counter of Network Comunication storage Group. | | |
| **Definition** | | |
| LOCAL uint16 ERH\_u16DebounceCounter\_NetworkComunication = KU16\_ZERO | | |
| **Remarks** | | |
| None. | | |

### ERH\_u16DebounceCounter\_NetworkComunication15

|  |  |  |
| --- | --- | --- |
| Type | Value |  |
| uint16 | KU16\_ZERO | |
| **Description** | | |
| The Counter of Network Comunication 15 storage Group. | | |
| **Definition** | | |
| LOCAL uint16 ERH\_u16DebounceCounter\_NetworkComunication15 = KU16\_ZERO | | |
| **Remarks** | | |
| None. | | |

### ERH\_u16DebounceCounter\_PowerDistribution

|  |  |  |
| --- | --- | --- |
| Type | Value |  |
| uint16 | KU16\_ZERO | |
| **Description** | | |
| The Counter of Power Distribution storage Group. | | |
| **Definition** | | |
| LOCAL uint16 ERH\_u16DebounceCounter\_PowerDistribution = KU16\_ZERO | | |
| **Remarks** | | |
| None. | | |

### ERH\_u16StorageConditionStatus

|  |  |  |
| --- | --- | --- |
| Type | Value |  |
| uint16 | NA | |
| **Description** | | |
| The Status mask used for Storage Conditions. | | |
| **Definition** | | |
| LOCAL uint16 ERH\_u16StorageConditionStatus | | |
| **Remarks** | | |
| None. | | |

### ERH\_u8Index\_ECULevel

|  |  |  |
| --- | --- | --- |
| Type | Value |  |
| uint8 | KU8\_ZERO | |
| **Description** | | |
| This variable is used to ... | | |
| **Definition** | | |
| LOCAL uint8 ERH\_u8Index\_ECULevel = KU8\_ZERO | | |

### ERH\_u8Index\_NetworkCom

|  |  |  |
| --- | --- | --- |
| Type | Value |  |
| uint8 | KU8\_ZERO | |
| **Description** | | |
| This variable is used to ... | | |
| **Definition** | | |
| LOCAL uint8 ERH\_u8Index\_NetworkCom = KU8\_ZERO | | |

### ERH\_u8Index\_NoStorageCondition

|  |  |  |
| --- | --- | --- |
| Type | Value |  |
| uint8 | KU8\_ZERO | |
| **Description** | | |
| This variable is used to ... | | |
| **Definition** | | |
| LOCAL uint8 ERH\_u8Index\_NoStorageCondition = KU8\_ZERO | | |

### ERH\_u8Index\_PowerDistribution

|  |  |  |
| --- | --- | --- |
| Type | Value |  |
| uint8 | KU8\_ZERO | |
| **Description** | | |
| This variable is used to ... | | |
| **Definition** | | |
| LOCAL uint8 ERH\_u8Index\_PowerDistribution = KU8\_ZERO | | |

## Macros

### KU8\_MAX\_PENDING\_EVENT

|  |  |
| --- | --- |
| Name | Value |
| KU8\_MAX\_PENDING\_EVENT | 8 |
| **Definition** | |
| #define KU8\_MAX\_PENDING\_EVENT ((uint8)8) | |
| **Description** | |
| Maximum number of pending events allowed | |

### KU8\_NUMBER\_OF\_GROUPS

|  |  |
| --- | --- |
| Name | Value |
| KU8\_NUMBER\_OF\_GROUPS | 30 |
| **Definition** | |
| #define KU8\_NUMBER\_OF\_GROUPS ((uint8)30) | |
| **Description** | |
| Size array with the number of AEC groups | |

### KU8\_NVP\_AEC\_RECORDING\_OK

|  |  |
| --- | --- |
| Name | Value |
| KU8\_NVP\_AEC\_RECORDING\_OK | 0 |
| **Definition** | |
| #define KU8\_NVP\_AEC\_RECORDING\_OK ((uint8)0) | |
| **Description** | |
| The NVP Recording OK value of "Ever qualified" AEC informations | |

### KU8\_NVP\_AEC\_RECORDING\_NOK

|  |  |
| --- | --- |
| Name | Value |
| KU8\_NVP\_AEC\_RECORDING\_NOK | 0xFF |
| **Definition** | |
| #define KU8\_NVP\_AEC\_RECORDING\_NOK ((uint8)0xFF) | |
| **Description** | |
| The NVP Recording “Not OK” value of "Ever qualified" AEC informations | |

### KU8\_NVP\_AEC\_RECORDING\_IN\_PROGRESS

|  |  |
| --- | --- |
| Name | Value |
| KU8\_NVP\_AEC\_RECORDING\_IN\_PROGRESS | 0x55 |
| **Definition** | |
| #define KU8\_NVP\_AEC\_RECORDING\_IN\_PROGRESS ((uint8)0x55) | |
| **Description** | |
| The NVP Recording “In Progress” value of "Ever qualified" AEC informations | |

### ERH\_KU8\_NUMBER\_OF\_AEC

|  |  |
| --- | --- |
| Name | Value |
| ERH\_KU8\_NUMBER\_OF\_AEC | ((uint8)0x36) |
| **Definition** | |
| #define ERH\_KU8\_NUMBER\_OF\_AEC ((uint8)0x36) | |
| **Description** | |
| Number of all AECs | |

### KU8\_NVP\_BLOCK\_JOB\_PENDING

|  |  |
| --- | --- |
| Name | Value |
| KU8\_NVP\_BLOCK\_JOB\_PENDING | (4U) |
| **Definition** | |
| #define KU8\_NVP\_BLOCK\_JOB\_PENDING (4U) | |
| **Description** | |
| Define for block job pending | |

### NVP\_BLOCK\_ID\_AEC\_STATUS

|  |  |
| --- | --- |
| Name | Value |
| NVP\_BLOCK\_ID\_AEC\_STATUS | ( 25u ) |
| **Definition** | |
| #define NVP\_BLOCK\_ID\_AEC\_STATUS ( 25u ) | |
| **Description** | |
| Define for NVP block where are IDs of AEC status | |

### KU16\_COUNTER\_TRESHOLD\_IGN

|  |  |
| --- | --- |
| Name | Value |
| KU16\_COUNTER\_TRESHOLD\_IGN | ( 1000u ) |
| **Definition** | |
| #define KU16\_COUNTER\_TRESHOLD\_IGN ( 1000u ) | |
| **Description** | |
| Define for debounce counter. | |

### KU16\_ERH\_MASK\_DTC\_CONFIRMED

|  |  |
| --- | --- |
| Name | Value |
| KU16\_ERH\_MASK\_DTC\_CONFIRMED | ( 0x008 ) |
| **Definition** | |
| #define KU16\_ERH\_MASK\_DTC\_CONFIRMED ((uint16)0x008) | |
| **Description** | |
| Define for DTC Confirmed bit status | |

### BOOL\_ERH\_CLEAR\_STORAGE\_CONDITION

|  |  |
| --- | --- |
| Name | Value |
| BOOL\_ERH\_CLEAR\_STORAGE\_CONDITION | (0u) |
| **Definition** | |
| #define BOOL\_ERH\_CLEAR\_STORAGE\_CONDITION (0u) | |
| **Description** | |
| Storage Condition off | |

### BOOL\_ERH\_SET\_STORAGE\_CONDITION

|  |  |
| --- | --- |
| Name | Value |
| BOOL\_ERH\_SET\_STORAGE\_CONDITION | (1u) |
| **Definition** | |
| #define BOOL\_ERH\_SET\_STORAGE\_CONDITION (1u) | |
| **Description** | |
| Storage Condition on | |

### ERH\_SET\_STORAGE\_CONDITION\_MASK\_OFF(mask)

|  |  |
| --- | --- |
| Name | Value |
| ERH\_SET\_STORAGE\_CONDITION\_MASK\_OFF(mask) | (mask) U16\_BIT\_CLEAR(ERH\_u16StorageConditionStatus, (mask)) |
| **Definition** | |
| #define ERH\_SET\_STORAGE\_CONDITION\_MASK\_OFF(mask) U16\_BIT\_CLEAR(ERH\_u16StorageConditionStatus, (mask)) | |
| **Description** | |
| Macro used for bit clearing inhibition status. | |

### ERH\_SET\_STORAGE\_CONDITION\_MASK\_ON(mask)

|  |  |
| --- | --- |
| Name | Value |
| ERH\_SET\_STORAGE\_CONDITION\_MASK\_ON(mask) | (mask) U16\_BIT\_SET(ERH\_u16StorageConditionStatus, (mask)) |
| **Definition** | |
| #define ERH\_SET\_STORAGE\_CONDITION\_MASK\_ON(mask) U16\_BIT\_SET(ERH\_u16StorageConditionStatus, (mask)) | |
| **Description** | |
| Macro used for bit setting inhibition status. | |

### KU16\_ERH\_MASK\_CONTROL\_DTC

|  |  |
| --- | --- |
| Name | Value |
| KU16\_ERH\_MASK\_CONTROL\_DTC | (1u) |
| **Definition** | |
| #define KU16\_ERH\_MASK\_CONTROL\_DTC (1u) | |
| **Description** | |
| CONTROL\_DTC mask | |

### KU16\_ERH\_MASK\_DEBOUNCE\_TIMER\_NC

|  |  |
| --- | --- |
| Name | Value |
| KU16\_ERH\_MASK\_DEBOUNCE\_TIMER\_NC | ((uint16) 0x63B) |
| **Definition** | |
| #define KU16\_ERH\_MASK\_DEBOUNCE\_TIMER\_NC ((uint16) 0x63B) | |
| **Description** | |
| DEBOUNCE\_TIMER\_NC mask | |

### KU16\_ERH\_MASK\_DEBOUNCE\_TIMER\_NC15

|  |  |
| --- | --- |
| Name | Value |
| KU16\_ERH\_MASK\_DEBOUNCE\_TIMER\_NC15 | ((uint16) 0x639) |
| **Definition** | |
| #define KU16\_ERH\_MASK\_DEBOUNCE\_TIMER\_NC15 ((uint16) 0x639) | |
| **Description** | |
| DEBOUNCE\_TIMER\_NC15 mask | |

### KU16\_ERH\_MASK\_DEBOUNCE\_TIMER\_PD

|  |  |
| --- | --- |
| Name | Value |
| KU16\_ERH\_MASK\_DEBOUNCE\_TIMER\_PD | ((uint16) 0x633) |
| **Definition** | |
| #define KU16\_ERH\_MASK\_DEBOUNCE\_TIMER\_PD ((uint16) 0x633) | |
| **Description** | |
| NA | |

### KU16\_ERH\_MASK\_DTC\_CONFIRMED

|  |  |
| --- | --- |
| Name | Value |
| KU16\_ERH\_MASK\_DTC\_CONFIRMED | ((uint16)0x008) |
| **Definition** | |
| #define KU16\_ERH\_MASK\_DTC\_CONFIRMED ((uint16)0x008) | |
| **Description** | |
| DTC\_CONFIRMED mask | |

### KU16\_ERH\_MASK\_ECU\_LEVEL

|  |  |
| --- | --- |
| Name | Value |
| KU16\_ERH\_MASK\_ECU\_LEVEL | (517u) |
| **Definition** | |
| #define KU16\_ERH\_MASK\_ECU\_LEVEL (517u) | |
| **Description** | |
| ECU\_LEVEL storage condition group mask | |

### KU16\_ERH\_MASK\_NETWORK\_COM

|  |  |
| --- | --- |
| Name | Value |
| KU16\_ERH\_MASK\_NETWORK\_COM | ((uint16)0x6BB) |
| **Definition** | |
| #define KU16\_ERH\_MASK\_NETWORK\_COM ((uint16)0x6BB) | |
| **Description** | |
| NETWORK\_COM storage condition group mask | |

### KU16\_ERH\_MASK\_NETWORK\_COM15

|  |  |
| --- | --- |
| Name | Value |
| KU16\_ERH\_MASK\_NETWORK\_COM15 | ((uint16)0x739) |
| **Definition** | |
| #define KU16\_ERH\_MASK\_NETWORK\_COM15 ((uint16)0x739) | |
| **Description** | |
| NETWORK\_COM15 storage condition group mask | |

### KU16\_ERH\_MASK\_POWER\_DISTRIBUTION

|  |  |
| --- | --- |
| Name | Value |
| KU16\_ERH\_MASK\_POWER\_DISTRIBUTION | ((uint16)0x673) |
| **Definition** | |
| #define KU16\_ERH\_MASK\_POWER\_DISTRIBUTION ((uint16)0x673) | |
| **Description** | |
| POWER\_DISTRIBUTION mask | |

### KU16\_ERH\_SC10\_0\_debounceTimer

|  |  |
| --- | --- |
| Name | Value |
| KU16\_ERH\_SC10\_0\_debounceTimer | (64u) |
| **Definition** | |
| #define KU16\_ERH\_SC10\_0\_debounceTimer (64u) | |
| **Description** | |
| SC10\_0\_debounceTimer mask | |

### KU16\_ERH\_SC10\_1\_debounceTimer

|  |  |
| --- | --- |
| Name | Value |
| KU16\_ERH\_SC10\_1\_debounceTimer | (128u) |
| **Definition** | |
| #define KU16\_ERH\_SC10\_1\_debounceTimer (128u) | |
| **Description** | |
| SC10\_1\_debounceTimer mask | |

### KU16\_ERH\_SC10\_2\_debounceTimer

|  |  |
| --- | --- |
| Name | Value |
| KU16\_ERH\_SC10\_2\_debounceTimer | (256u) |
| **Definition** | |
| #define KU16\_ERH\_SC10\_2\_debounceTimer (256u) | |
| **Description** | |
| SC10\_2\_debounceTimer mask | |

### KU16\_ERH\_SC11\_VehicleStartup

|  |  |
| --- | --- |
| Name | Value |
| KU16\_ERH\_SC11\_VehicleStartup | (512u) |
| **Definition** | |
| #define KU16\_ERH\_SC11\_VehicleStartup (512u) | |
| **Description** | |
| VehicleStartup mask | |

### KU16\_ERH\_SC12\_ProductionMode

|  |  |
| --- | --- |
| Name | Value |
| KU16\_ERH\_SC12\_ProductionMode | (1024u) |
| **Definition** | |
| #define KU16\_ERH\_SC12\_ProductionMode (1024u) | |
| **Description** | |
| ProductionMode mask | |

### KU16\_ERH\_SC1\_ControlDTC

|  |  |
| --- | --- |
| Name | Value |
| KU16\_ERH\_SC1\_ControlDTC | (1u) |
| **Definition** | |
| #define KU16\_ERH\_SC1\_ControlDTC (1u) | |
| **Description** | |
| ControlDTC mask | |

### KU16\_ERH\_SC2\_IgnitonStatus

|  |  |
| --- | --- |
| Name | Value |
| KU16\_ERH\_SC2\_IgnitonStatus | (2u) |
| **Definition** | |
| #define KU16\_ERH\_SC2\_IgnitonStatus (2u) | |
| **Description** | |
| IgnitonStatus mask | |

### KU16\_ERH\_SC3\_LocalVoltage

|  |  |
| --- | --- |
| Name | Value |
| KU16\_ERH\_SC3\_LocalVoltage | (4u) |
| **Definition** | |
| #define KU16\_ERH\_SC3\_LocalVoltage (4u) | |
| **Description** | |
| LocalVoltage mask | |

### KU16\_ERH\_SC4\_SystemVoltage

|  |  |
| --- | --- |
| Name | Value |
| KU16\_ERH\_SC4\_SystemVoltage | (8u) |
| **Definition** | |
| #define KU16\_ERH\_SC4\_SystemVoltage (8u) | |
| **Description** | |
| SystemVoltage mask | |

### KU16\_ERH\_SC5\_TransportationMode

|  |  |
| --- | --- |
| Name | Value |
| KU16\_ERH\_SC5\_TransportationMode | (16u) |
| **Definition** | |
| #define KU16\_ERH\_SC5\_TransportationMode (16u) | |
| **Description** | |
| TransportationMode mask | |

### KU16\_ERH\_SC8\_CommunicationStatus

|  |  |
| --- | --- |
| Name | Value |
| KU16\_ERH\_SC8\_CommunicationStatus | (32u) |
| **Definition** | |
| #define KU16\_ERH\_SC8\_CommunicationStatus (32u) | |
| **Description** | |
| CommunicationStatus mask | |

### KU16\_ERH\_TIMER\_TRESHOLD

|  |  |
| --- | --- |
| Name | Value |
| KU16\_ERH\_TIMER\_TRESHOLD | (500u) |
| **Definition** | |
| #define KU16\_ERH\_TIMER\_TRESHOLD (500u) | |
| **Description** | |
| Counter used for SC10 debounce timer | |

# EEPROM

The EEPROM parameters are all specified in [Doc3 = SBE\_4G\_NVP\_layout.xls].

Refer to this document for more details.

# Configuration

No special configuration for ERH software component.

# Compilation Options

No compilation options for ERH software component.