Software Low Level Requirements for CMU Module of Engine Data Acquisition Unit of Airbus Helicopter Generic Vehicle Monitoring System and CMU+

Document Number: H698-003-011-CMU

Version No: 2.3

|  |
| --- |
| Proprietary Notice  This document and the information disclosed herein are proprietary data of Howell Instruments, Inc. Neither this document nor the information contained herein shall be reproduced, used, or disclosed to others without the written authorization of Howell Instruments, Inc. |

|  |  |
| --- | --- |
| Written By: | MUPPINENI SRAVANTHI (Senior Software Engineer, ALTEN Global Technologies **Private Limited**) |
| Reviewed By: | Riyaaz Basha.J (Senior Software Engineer, ALTEN Global Technologies **Private Limited**)    Pavitra D (Engineer-Quality – Quality Assurance, ALTEN Global Technologies **Private Limited**) |
| Approved By: | Shyamala B (Senior Director– Quality Assurance and Control , ALTEN Global Technologies **Private Limited** ) |
| Released By: | Madhu Chandra Srinivas (Associate Director – Software, ALTEN Global Technologies **Private Limited**) |

A logo with a triangle and a triangle in a red yellow and blue rectangle

Description automatically generated

**ALTEN Global Technologies Pvt Ltd.**

**Table of Contents**

[1 Amendment Record 41](#_Toc158905411)

[2 Objective 50](#_Toc158905412)

[3 Acronyms and Definitions 50](#_Toc158905413)

[4 Scope 51](#_Toc158905414)

[5 References 51](#_Toc158905415)

[6 Document Control 52](#_Toc158905416)

[7 Distributions 52](#_Toc158905417)

[8 Traceability 52](#_Toc158905418)

[9 Responsibilities 53](#_Toc158905419)

[10 Software Low Level Requirements - Flight Module 53](#_Toc158905420)

[10.1 daucmfa825comm 53](#_Toc158905421)

[10.1.1 VSendPartNumberData 53](#_Toc158905422)

[10.1.1.1 Brief Description 53](#_Toc158905423)

[10.1.1.2 List of HLRs allocated 53](#_Toc158905424)

[10.1.1.3 List of global variables accessed and modified 54](#_Toc158905425)

[10.1.1.4 Parameter list (Input/Output) 54](#_Toc158905426)

[10.1.1.5 Return Value 54](#_Toc158905427)

[10.1.1.6 Other CSUs called by this CSU 54](#_Toc158905428)

[10.1.1.7 Description of list of LLRs allocated 54](#_Toc158905429)

[10.1.1.7.1 daucmfa825comm-VSendPartNumberData-LLR-001 54](#_Toc158905430)

[10.1.1.7.2 daucmfa825comm-VSendPartNumberData-LLR-002 55](#_Toc158905431)

[10.1.1.7.3 daucmfa825comm-VSendPartNumberData-LLR-003 55](#_Toc158905432)

[10.1.2 VSendNvmData 55](#_Toc158905433)

[10.1.2.1 Brief Description 55](#_Toc158905434)

[10.1.2.2 List of HLRs allocated 55](#_Toc158905435)

[10.1.2.3 List of global variables accessed and modified 55](#_Toc158905436)

[10.1.2.4 Parameter list (Input/Output) 55](#_Toc158905437)

[10.1.2.5 Return Value 56](#_Toc158905438)

[10.1.2.6 Other CSUs called by this CSU 56](#_Toc158905439)

[10.1.2.7 Description of list of LLRs allocated 56](#_Toc158905440)

[10.1.2.7.1 daucmfa825comm-VSendNvmData-LLR-001 56](#_Toc158905441)

[10.1.2.7.2 daucmfa825comm-VSendNvmData-LLR-002 56](#_Toc158905442)

[10.1.2.7.3 daucmfa825comm-VSendNvmData-LLR-003 57](#_Toc158905443)

[10.1.2.7.4 daucmfa825comm-VSendNvmData-LLR-004 57](#_Toc158905444)

[10.1.2.7.5 daucmfa825comm-VSendNvmData-LLR-005 57](#_Toc158905445)

[10.1.3 VProcessNOC 57](#_Toc158905446)

[10.1.3.1 Brief Description 57](#_Toc158905447)

[10.1.3.2 List of HLRs allocated 57](#_Toc158905448)

[10.1.3.3 List of global variables accessed and modified 58](#_Toc158905449)

[10.1.3.4 Parameter list (Input/Output) 58](#_Toc158905450)

[10.1.3.5 Return Value 58](#_Toc158905451)

[10.1.3.6 Other CSUs called by this CSU 58](#_Toc158905452)

[10.1.3.7 Description of list of LLRs allocated 58](#_Toc158905453)

[10.1.3.7.1 daucmfa825comm-VProcessNOC-LLR-001 58](#_Toc158905454)

[10.1.4 VProcessNSC 58](#_Toc158905455)

[10.1.4.1 Brief Description 58](#_Toc158905456)

[10.1.4.2 List of HLRs allocated 59](#_Toc158905457)

[10.1.4.3 List of global variables accessed and modified 59](#_Toc158905458)

[10.1.4.4 Parameter list (Input/Output) 59](#_Toc158905459)

[10.1.4.5 Return Value 59](#_Toc158905460)

[10.1.4.6 Other CSUs called by this CSU 59](#_Toc158905461)

[10.1.4.7 Description of list of LLRs allocated 60](#_Toc158905462)

[10.1.4.7.1 daucmfa825comm-VProcessNSC-LLR-001 60](#_Toc158905463)

[10.1.4.7.2 daucmfa825comm-VProcessNSC-LLR-002 60](#_Toc158905464)

[10.1.4.7.3 daucmfa825comm-VProcessNSC-LLR-003 60](#_Toc158905465)

[10.1.4.7.4 daucmfa825comm-VProcessNSC-LLR-004 61](#_Toc158905466)

[10.1.4.7.5 daucmfa825comm-VProcessNSC-LLR-005 61](#_Toc158905467)

[10.1.4.7.6 daucmfa825comm-VProcessNSC-LLR-006 62](#_Toc158905468)

[10.1.4.7.7 daucmfa825comm-VProcessNSC-LLR-007 62](#_Toc158905469)

[10.1.4.7.8 daucmfa825comm-VProcessNSC-LLR-008 62](#_Toc158905470)

[10.1.5 A825CommTask 63](#_Toc158905471)

[10.1.5.1 Brief Description 63](#_Toc158905472)

[10.1.5.2 List of HLRs allocated 63](#_Toc158905473)

[10.1.5.3 List of global variables accessed and modified 63](#_Toc158905474)

[10.1.5.4 Parameter list (Input/Output) 63](#_Toc158905475)

[10.1.5.5 Return Value 63](#_Toc158905476)

[10.1.5.6 Other CSUs called by this CSU 63](#_Toc158905477)

[10.1.5.7 Description of list of LLRs allocated 64](#_Toc158905478)

[10.1.5.7.1 daucmfa825comm-A825CommTask-LLR-001 64](#_Toc158905479)

[10.1.5.7.2 daucmfa825comm-A825CommTask-LLR-002 64](#_Toc158905480)

[10.1.5.7.3 daucmfa825comm-A825CommTask-LLR-003 64](#_Toc158905481)

[10.1.5.7.4 daucmfa825comm-A825CommTask-LLR-004 64](#_Toc158905482)

[10.1.5.7.5 daucmfa825comm-A825CommTask-LLR-005 64](#_Toc158905483)

[10.1.5.7.6 daucmfa825comm-A825CommTask-LLR-006 65](#_Toc158905484)

[10.1.5.7.7 daucmfa825comm-A825CommTask-LLR-007 65](#_Toc158905485)

[10.1.5.7.8 daucmfa825comm-A825CommTask-LLR-008 65](#_Toc158905486)

[10.1.5.7.9 daucmfa825comm-A825CommTask-LLR-009 65](#_Toc158905487)

[10.1.5.7.10 daucmfa825comm-A825CommTask-LLR-010 66](#_Toc158905488)

[10.1.5.7.11 daucmfa825comm-A825CommTask-LLR-011 66](#_Toc158905489)

[10.1.5.7.12 daucmfa825comm-A825CommTask-LLR-012 66](#_Toc158905490)

[10.1.5.7.13 daucmfa825comm-A825CommTask-LLR-013 66](#_Toc158905491)

[10.1.5.7.14 daucmfa825comm-A825CommTask-LLR-014 67](#_Toc158905492)

[10.1.5.7.15 daucmfa825comm-A825CommTask-LLR-015 67](#_Toc158905493)

[10.1.5.7.16 daucmfa825comm-A825CommTask-LLR-016 67](#_Toc158905494)

[10.1.5.7.17 daucmfa825comm-A825CommTask-LLR-017 67](#_Toc158905495)

[10.1.5.7.18 daucmfa825comm-A825CommTask-LLR-018 67](#_Toc158905496)

[10.1.5.7.19 daucmfa825comm-A825CommTask-LLR-019 68](#_Toc158905497)

[10.1.6 A825CommInit 68](#_Toc158905498)

[10.1.6.1 Brief Description 68](#_Toc158905499)

[10.1.6.2 List of HLRs allocated 68](#_Toc158905500)

[10.1.6.3 List of global variables accessed and modified 68](#_Toc158905501)

[10.1.6.4 Parameter list (Input/Output) 68](#_Toc158905502)

[10.1.6.5 Return Value 68](#_Toc158905503)

[10.1.6.6 Other CSUs called by this CSU 69](#_Toc158905504)

[10.1.6.7 Description of list of LLRs allocated 69](#_Toc158905505)

[10.1.6.7.1 daucmfa825comm-A825CommInit-LLR-002 69](#_Toc158905506)

[10.1.6.7.2 daucmfa825comm-A825CommInit-LLR-003 69](#_Toc158905507)

[10.1.6.7.3 daucmfa825comm-A825CommInit-LLR-004 69](#_Toc158905508)

[10.1.7 ResetHardware 70](#_Toc158905509)

[10.1.7.1 Brief Description 70](#_Toc158905510)

[10.1.7.2 List of HLRs allocated 70](#_Toc158905511)

[10.1.7.3 List of global variables accessed and modified 70](#_Toc158905512)

[10.1.7.4 Parameter list (Input/Output) 70](#_Toc158905513)

[10.1.7.5 Return Value 70](#_Toc158905514)

[10.1.7.6 Other CSUs called by this CSU 70](#_Toc158905515)

[10.1.7.7 Description of list of LLRs allocated 70](#_Toc158905516)

[10.1.7.7.1 daucmfa825comm-ResetHardware-LLR-001 71](#_Toc158905517)

[10.1.7.7.2 daucmfa825comm-ResetHardware-LLR-002 71](#_Toc158905518)

[10.1.7.7.3 daucmfa825comm-ResetHardware-LLR-003 71](#_Toc158905519)

[10.1.7.7.4 daucmfa825comm-ResetHardware-LLR-004 71](#_Toc158905520)

[10.1.7.7.5 daucmfa825comm-ResetHardware-LLR-005 71](#_Toc158905521)

[10.1.7.7.6 daucmfa825comm-ResetHardware-LLR-006 71](#_Toc158905522)

[10.1.7.7.7 daucmfa825comm-ResetHardware-LLR-007 72](#_Toc158905523)

[10.1.7.7.8 daucmfa825comm-ResetHardware-LLR-008 72](#_Toc158905524)

[10.1.8 SetStack 72](#_Toc158905525)

[10.1.8.1 Brief Description 72](#_Toc158905526)

[10.1.8.2 List of HLRs allocated 72](#_Toc158905527)

[10.1.8.3 List of global variables accessed and modified 72](#_Toc158905528)

[10.1.8.4 Parameter list (Input/Output) 72](#_Toc158905529)

[10.1.8.5 Return Value 73](#_Toc158905530)

[10.1.8.6 Other CSUs called by this CSU 73](#_Toc158905531)

[10.1.8.7 Description of list of LLRs allocated 73](#_Toc158905532)

[10.1.8.7.1 daucmfa825comm-SetStack-LLR-001 73](#_Toc158905533)

[10.1.9 StartLogger 73](#_Toc158905534)

[10.1.9.1 Brief Description 73](#_Toc158905535)

[10.1.9.2 List of HLRs allocated 73](#_Toc158905536)

[10.1.9.3 List of global variables accessed and modified 73](#_Toc158905537)

[10.1.9.4 Parameter list (Input/Output) 74](#_Toc158905538)

[10.1.9.5 Return Value 74](#_Toc158905539)

[10.1.9.6 Other CSUs called by this CSU 74](#_Toc158905540)

[10.1.9.7 Description of list of LLRs allocated 74](#_Toc158905541)

[10.1.9.7.1 daucmfa825comm-StartLogger-LLR-001 74](#_Toc158905542)

[10.1.9.7.2 daucmfa825comm-StartLogger-LLR-002 74](#_Toc158905543)

[10.1.9.7.3 daucmfa825comm-StartLogger-LLR-003 75](#_Toc158905544)

[10.1.9.7.4 daucmfa825comm-StartLogger-LLR-004 75](#_Toc158905545)

[10.2 daucmfarinc825 75](#_Toc158905546)

[10.2.1 RxQueueRemove 75](#_Toc158905547)

[10.2.1.1 Brief Description 75](#_Toc158905548)

[10.2.1.2 List of HLRs allocated 75](#_Toc158905549)

[10.2.1.3 List of global variables accessed and modified 75](#_Toc158905550)

[10.2.1.4 Parameter list (Input/Output) 75](#_Toc158905551)

[10.2.1.5 Return Value 76](#_Toc158905552)

[10.2.1.6 Other CSUs called by this CSU 76](#_Toc158905553)

[10.2.1.7 Description of list of LLRs allocated 76](#_Toc158905554)

[10.2.1.7.1 daucmfarinc825-RxQueueRemove-LLR-001 76](#_Toc158905555)

[10.2.1.7.2 daucmfarinc825-RxQueueRemove-LLR-002 76](#_Toc158905556)

[10.2.1.7.3 daucmfarinc825-RxQueueRemove-LLR-003 76](#_Toc158905557)

[10.2.1.7.4 daucmfarinc825-RxQueueRemove-LLR-004 76](#_Toc158905558)

[10.2.1.7.5 daucmfarinc825-RxQueueRemove-LLR-005 77](#_Toc158905559)

[10.2.1.7.6 daucmfarinc825-RxQueueRemove-LLR-006 77](#_Toc158905560)

[10.2.1.7.7 daucmfarinc825-RxQueueRemove-LLR-007 77](#_Toc158905561)

[10.2.1.7.8 daucmfarinc825-RxQueueRemove-LLR-008 77](#_Toc158905562)

[10.2.1.7.9 daucmfarinc825-RxQueueRemove-LLR-009 78](#_Toc158905563)

[10.2.1.7.10 daucmfarinc825-RxQueueRemove-LLR-010 78](#_Toc158905564)

[10.2.1.7.11 daucmfarinc825-RxQueueRemove-LLR-011 78](#_Toc158905565)

[10.2.1.7.12 daucmfarinc825-RxQueueRemove-LLR-012 78](#_Toc158905566)

[10.2.1.7.13 daucmfarinc825-RxQueueRemove-LLR-013 79](#_Toc158905567)

[10.2.2 A825GetMessage 79](#_Toc158905568)

[10.2.2.1 Brief Description 79](#_Toc158905569)

[10.2.2.2 List of HLRs allocated 79](#_Toc158905570)

[10.2.2.3 List of global variables accessed and modified 79](#_Toc158905571)

[10.2.2.4 Parameter list (Input/Output) 79](#_Toc158905572)

[10.2.2.5 Return Value 79](#_Toc158905573)

[10.2.2.6 Other CSUs called by this CSU 79](#_Toc158905574)

[10.2.2.7 Description of list of LLRs allocated 80](#_Toc158905575)

[10.2.2.7.1 daucmfarinc825-A825GetMessage-LLR-001 80](#_Toc158905576)

[10.2.2.7.2 daucmfarinc825-A825GetMessage-LLR-002 80](#_Toc158905577)

[10.2.2.7.3 daucmfarinc825-A825GetMessage-LLR-003 80](#_Toc158905578)

[10.2.2.7.4 daucmfarinc825-A825GetMessage-LLR-004 80](#_Toc158905579)

[10.2.3 RxQueueInsert 80](#_Toc158905580)

[10.2.3.1 Brief Description 80](#_Toc158905581)

[10.2.3.2 List of HLRs allocated 81](#_Toc158905582)

[10.2.3.3 List of global variables accessed and modified 81](#_Toc158905583)

[10.2.3.4 Parameter list (Input/Output) 81](#_Toc158905584)

[10.2.3.5 Return Value 81](#_Toc158905585)

[10.2.3.6 Other CSUs called by this CSU 81](#_Toc158905586)

[10.2.3.7 Description of list of LLRs allocated 81](#_Toc158905587)

[10.2.3.7.1 daucmfarinc825-RxQueueInsert-LLR-001 81](#_Toc158905588)

[10.2.3.7.2 daucmfarinc825-RxQueueInsert-LLR-002 82](#_Toc158905589)

[10.2.3.7.3 daucmfarinc825-RxQueueInsert-LLR-003 82](#_Toc158905590)

[10.2.3.7.4 daucmfarinc825-RxQueueInsert-LLR-004 82](#_Toc158905591)

[10.2.3.7.5 daucmfarinc825-RxQueueInsert-LLR-005 82](#_Toc158905592)

[10.2.4 A825ReceiveMessage 82](#_Toc158905593)

[10.2.4.1 Brief Description 82](#_Toc158905594)

[10.2.4.2 List of HLRs allocated 82](#_Toc158905595)

[10.2.4.3 List of global variables accessed and modified 83](#_Toc158905596)

[10.2.4.4 Parameter list (Input/Output) 83](#_Toc158905597)

[10.2.4.5 Return Value 83](#_Toc158905598)

[10.2.4.6 Other CSUs called by this CSU 83](#_Toc158905599)

[10.2.4.7 Description of list of LLRs allocated 83](#_Toc158905600)

[10.2.4.7.1 daucmfarinc825-A825ReceiveMessage-LLR-001 83](#_Toc158905601)

[10.2.4.7.2 daucmfarinc825-A825ReceiveMessage-LLR-002 83](#_Toc158905602)

[10.2.4.7.3 daucmfarinc825-A825ReceiveMessage-LLR-003 84](#_Toc158905603)

[10.2.4.7.4 daucmfarinc825-A825ReceiveMessage-LLR-004 84](#_Toc158905604)

[10.2.4.7.5 daucmfarinc825-A825ReceiveMessage-LLR-005 84](#_Toc158905605)

[10.2.4.7.6 daucmfarinc825-A825ReceiveMessage-LLR-006 84](#_Toc158905606)

[10.2.4.7.7 daucmfarinc825-A825ReceiveMessage-LLR-007 84](#_Toc158905607)

[10.2.4.7.8 daucmfarinc825-A825ReceiveMessage-LLR-008 85](#_Toc158905608)

[10.2.4.7.9 daucmfarinc825-A825ReceiveMessage-LLR-009 85](#_Toc158905609)

[10.2.5 TxQueueInsert 85](#_Toc158905610)

[10.2.5.1 Brief Description 85](#_Toc158905611)

[10.2.5.2 List of HLRs allocated 85](#_Toc158905612)

[10.2.5.3 List of global variables accessed and modified 85](#_Toc158905613)

[10.2.5.4 Parameter list (Input/Output) 85](#_Toc158905614)

[10.2.5.5 Return Value 86](#_Toc158905615)

[10.2.5.6 Other CSUs called by this CSU 86](#_Toc158905616)

[10.2.5.7 Description of list of LLRs allocated 86](#_Toc158905617)

[10.2.5.7.1 daucmfarinc825-TxQueueInsert-LLR-001 86](#_Toc158905618)

[10.2.5.7.2 daucmfarinc825-TxQueueInsert-LLR-002 86](#_Toc158905619)

[10.2.5.7.3 daucmfarinc825-TxQueueInsert-LLR-003 86](#_Toc158905620)

[10.2.5.7.4 daucmfarinc825-TxQueueInsert-LLR-004 86](#_Toc158905621)

[10.2.5.7.5 daucmfarinc825-TxQueueInsert-LLR-005 87](#_Toc158905622)

[10.2.5.7.6 daucmfarinc825-TxQueueInsert-LLR-006 87](#_Toc158905623)

[10.2.5.7.7 daucmfarinc825-TxQueueInsert-LLR-007 87](#_Toc158905624)

[10.2.6 BuildBroadcastExtId 87](#_Toc158905625)

[10.2.6.1 Brief Description 87](#_Toc158905626)

[10.2.6.2 List of HLRs allocated 87](#_Toc158905627)

[10.2.6.3 List of global variables accessed and modified 87](#_Toc158905628)

[10.2.6.4 Parameter list (Input/Output) 88](#_Toc158905629)

[10.2.6.5 Return Value 88](#_Toc158905630)

[10.2.6.6 Other CSUs called by this CSU 88](#_Toc158905631)

[10.2.6.7 Description of list of LLRs allocated 88](#_Toc158905632)

[10.2.6.7.1 daucmfarinc825-BuildBroadcastExtId-LLR-001 88](#_Toc158905633)

[10.2.6.7.2 daucmfarinc825-BuildBroadcastExtId-LLR-002 89](#_Toc158905634)

[10.2.7 BuildPeerToPeerExtId 89](#_Toc158905635)

[10.2.7.1 Brief Description 89](#_Toc158905636)

[10.2.7.2 List of HLRs allocated 89](#_Toc158905637)

[10.2.7.3 List of global variables accessed and modified 89](#_Toc158905638)

[10.2.7.4 Parameter list (Input/Output) 89](#_Toc158905639)

[10.2.7.5 Return Value 89](#_Toc158905640)

[10.2.7.6 Other CSUs called by this CSU 89](#_Toc158905641)

[10.2.7.7 Description of list of LLRs allocated 90](#_Toc158905642)

[10.2.7.7.1 daucmfarinc825-BuildPeerToPeerExtId-LLR-001 90](#_Toc158905643)

[10.2.7.7.2 daucmfarinc825-BuildPeerToPeerExtId-LLR-002 90](#_Toc158905644)

[10.2.8 A825Xmit 90](#_Toc158905645)

[10.2.8.1 Brief Description 90](#_Toc158905646)

[10.2.8.2 List of HLRs allocated 91](#_Toc158905647)

[10.2.8.3 List of global variables accessed and modified 91](#_Toc158905648)

[10.2.8.4 Parameter list (Input/Output) 91](#_Toc158905649)

[10.2.8.5 Return Value 91](#_Toc158905650)

[10.2.8.6 Other CSUs called by this CSU 91](#_Toc158905651)

[10.2.8.7 Description of list of LLRs allocated 91](#_Toc158905652)

[10.2.8.7.1 daucmfarinc825-A825Xmit-LLR-001 91](#_Toc158905653)

[10.2.8.7.2 daucmfarinc825-A825Xmit-LLR-002 92](#_Toc158905654)

[10.2.8.7.3 daucmfarinc825-A825Xmit-LLR-003 92](#_Toc158905655)

[10.2.8.7.4 daucmfarinc825-A825Xmit-LLR-004 92](#_Toc158905656)

[10.2.8.7.5 daucmfarinc825-A825Xmit-LLR-005 92](#_Toc158905657)

[10.2.8.7.6 daucmfarinc825-A825Xmit-LLR-006 92](#_Toc158905658)

[10.2.8.7.7 daucmfarinc825-A825Xmit-LLR-007 92](#_Toc158905659)

[10.2.9 XmitFromBuff 93](#_Toc158905660)

[10.2.9.1 Brief Description 93](#_Toc158905661)

[10.2.9.2 List of HLRs allocated 93](#_Toc158905662)

[10.2.9.3 List of global variables accessed and modified 93](#_Toc158905663)

[10.2.9.4 Parameter list (Input/Output) 93](#_Toc158905664)

[10.2.9.5 Return Value 93](#_Toc158905665)

[10.2.9.6 Other CSUs called by this CSU 93](#_Toc158905666)

[10.2.9.7 Description of list of LLRs allocated 94](#_Toc158905667)

[10.2.9.7.1 daucmfarinc825-XmitFromBuff-LLR-001 94](#_Toc158905668)

[10.2.9.7.2 daucmfarinc825-XmitFromBuff-LLR-002 94](#_Toc158905669)

[10.2.9.7.3 daucmfarinc825-XmitFromBuff-LLR-003 94](#_Toc158905670)

[10.2.9.7.4 daucmfarinc825-XmitFromBuff-LLR-004 94](#_Toc158905671)

[10.2.9.7.5 daucmfarinc825-XmitFromBuff-LLR-005 94](#_Toc158905672)

[10.2.10 A825BufferCheck 95](#_Toc158905673)

[10.2.10.1 Brief Description 95](#_Toc158905674)

[10.2.10.2 List of HLRs allocated 95](#_Toc158905675)

[10.2.10.3 List of global variables accessed and modified 95](#_Toc158905676)

[10.2.10.4 Parameter list (Input/Output) 95](#_Toc158905677)

[10.2.10.5 Return Value 95](#_Toc158905678)

[10.2.10.6 Other CSUs called by this CSU 95](#_Toc158905679)

[10.2.10.7 Description of list of LLRs allocated 95](#_Toc158905680)

[10.2.10.7.1 daucmfarinc825-A825BufferCheck-LLR-001 96](#_Toc158905681)

[10.2.10.7.2 daucmfarinc825-A825BufferCheck-LLR-002 96](#_Toc158905682)

[10.2.10.7.3 daucmfarinc825-A825BufferCheck-LLR-003 96](#_Toc158905683)

[10.2.11 CanConfig 96](#_Toc158905684)

[10.2.11.1 Brief Description 96](#_Toc158905685)

[10.2.11.2 List of HLRs allocated 96](#_Toc158905686)

[10.2.11.3 List of global variables accessed and modified 96](#_Toc158905687)

[10.2.11.4 Parameter list (Input/Output) 97](#_Toc158905688)

[10.2.11.5 Return Value 97](#_Toc158905689)

[10.2.11.6 Other CSUs called by this CSU 97](#_Toc158905690)

[10.2.11.7 Description of list of LLRs allocated 97](#_Toc158905691)

[10.2.11.7.1 daucmfarinc825-CanConfig-LLR-001 97](#_Toc158905692)

[10.2.11.7.2 daucmfarinc825-CanConfig-LLR-002 97](#_Toc158905693)

[10.2.11.7.3 daucmfarinc825-CanConfig-LLR-003 98](#_Toc158905694)

[10.2.11.7.4 daucmfarinc825-CanConfig-LLR-004 98](#_Toc158905695)

[10.2.11.7.5 daucmfarinc825-CanConfig-LLR-005 98](#_Toc158905696)

[10.2.11.7.6 daucmfarinc825-CanConfig-LLR-006 98](#_Toc158905697)

[10.2.11.7.7 daucmfarinc825-CanConfig-LLR-007 98](#_Toc158905698)

[10.2.11.7.8 daucmfarinc825-CanConfig-LLR-008 99](#_Toc158905699)

[10.2.11.7.9 daucmfarinc825-CanConfig-LLR-009 99](#_Toc158905700)

[10.2.12 NvicConfig 99](#_Toc158905701)

[10.2.12.1 Brief Description 99](#_Toc158905702)

[10.2.12.2 List of HLRs allocated 99](#_Toc158905703)

[10.2.12.3 List of global variables accessed and modified 100](#_Toc158905704)

[10.2.12.4 Parameter list (Input/Output) 100](#_Toc158905705)

[10.2.12.5 Return Value 100](#_Toc158905706)

[10.2.12.6 Other CSUs called by this CSU 100](#_Toc158905707)

[10.2.12.7 Description of list of LLRs allocated 100](#_Toc158905708)

[10.2.12.7.1 daucmfarinc825-NvicConfig-LLR-001 100](#_Toc158905709)

[10.2.12.7.2 daucmfarinc825-NvicConfig-LLR-002 100](#_Toc158905710)

[10.2.12.7.3 daucmfarinc825-NvicConfig-LLR-003 100](#_Toc158905711)

[10.2.12.7.4 daucmfarinc825-NvicConfig-LLR-004 101](#_Toc158905712)

[10.2.13 A825Init 101](#_Toc158905713)

[10.2.13.1 Brief Description 101](#_Toc158905714)

[10.2.13.2 List of HLRs allocated 101](#_Toc158905715)

[10.2.13.3 List of global variables accessed and modified 101](#_Toc158905716)

[10.2.13.4 Parameter list (Input/Output) 102](#_Toc158905717)

[10.2.13.5 Return Value 102](#_Toc158905718)

[10.2.13.6 Other CSUs called by this CSU 102](#_Toc158905719)

[10.2.13.7 Description of list of LLRs allocated 102](#_Toc158905720)

[10.2.13.7.1 daucmfarinc825-A825Init-LLR-001 102](#_Toc158905721)

[10.2.13.7.2 daucmfarinc825-A825Init-LLR-002 102](#_Toc158905722)

[10.3 daucmfcbit 102](#_Toc158905723)

[10.3.1 StackTest 102](#_Toc158905724)

[10.3.1.1 Brief Description 103](#_Toc158905725)

[10.3.1.2 List of HLRs allocated 103](#_Toc158905726)

[10.3.1.3 List of global variables accessed and modified 103](#_Toc158905727)

[10.3.1.4 Parameter list (Input/Output) 103](#_Toc158905728)

[10.3.1.5 Return Value 103](#_Toc158905729)

[10.3.1.6 Other CSUs called by this CSU 103](#_Toc158905730)

[10.3.1.7 Description of list of LLRs allocated 103](#_Toc158905731)

[10.3.1.7.1 daucmfcbit-StackTest-LLR-001 103](#_Toc158905732)

[10.3.2 ContinuousBitTask 104](#_Toc158905733)

[10.3.2.1 Brief Description 104](#_Toc158905734)

[10.3.2.2 List of HLRs allocated 104](#_Toc158905735)

[10.3.2.3 List of global variables accessed and modified 104](#_Toc158905736)

[10.3.2.4 Parameter list (Input/Output) 104](#_Toc158905737)

[10.3.2.5 Return Value 104](#_Toc158905738)

[10.3.2.6 Other CSUs called by this CSU 105](#_Toc158905739)

[10.3.2.7 Description of list of LLRs allocated 105](#_Toc158905740)

[10.3.2.7.1 daucmfcbit-ContinuousBitTask-LLR-001 105](#_Toc158905741)

[10.3.2.7.2 daucmfcbit-ContinuousBitTask-LLR-002 105](#_Toc158905742)

[10.3.3 ContinuousBitTaskInit 105](#_Toc158905743)

[10.3.3.1 Brief Description 105](#_Toc158905744)

[10.3.3.2 List of HLRs allocated 105](#_Toc158905745)

[10.3.3.3 List of global variables accessed and modified 106](#_Toc158905746)

[10.3.3.4 Parameter list (Input/Output) 106](#_Toc158905747)

[10.3.3.5 Return Value 106](#_Toc158905748)

[10.3.3.6 Other CSUs called by this CSU 106](#_Toc158905749)

[10.3.3.7 Description of list of LLRs allocated 106](#_Toc158905750)

[10.3.3.7.1 daucmfcbit-ContinuousBitTaskInit-LLR-001 106](#_Toc158905751)

[10.3.3.7.2 daucmfcbit-ContinuousBitTaskInit-LLR-002 107](#_Toc158905752)

[10.3.3.7.3 daucmfcbit-ContinuousBitTaskInit-LLR-003 107](#_Toc158905753)

[10.4 daucmfcputest 107](#_Toc158905754)

[10.4.1 AluTest 107](#_Toc158905755)

[10.4.1.1 Brief Description 107](#_Toc158905756)

[10.4.1.2 List of HLRs allocated 107](#_Toc158905757)

[10.4.1.3 List of global variables accessed and modified 107](#_Toc158905758)

[10.4.1.4 Parameter list (Input/Output) 108](#_Toc158905759)

[10.4.1.5 Return Value 108](#_Toc158905760)

[10.4.1.6 Other CSUs called by this CSU 108](#_Toc158905761)

[10.4.1.7 Description of list of LLRs allocated 108](#_Toc158905762)

[10.4.1.7.1 daucmfcputest-AluTest-LLR-001 108](#_Toc158905763)

[10.4.1.7.2 daucmfcputest-AluTest-LLR-002 108](#_Toc158905764)

[10.4.1.7.3 daucmfcputest-AluTest-LLR-003 108](#_Toc158905765)

[10.4.1.7.4 daucmfcputest-AluTest-LLR-004 109](#_Toc158905766)

[10.4.1.7.5 daucmfcputest-AluTest-LLR-005 109](#_Toc158905767)

[10.4.1.7.6 daucmfcputest-AluTest-LLR-006 109](#_Toc158905768)

[10.4.1.7.7 daucmfcputest-AluTest-LLR-007 109](#_Toc158905769)

[10.4.1.7.8 daucmfcputest-AluTest-LLR-008 109](#_Toc158905770)

[10.4.1.7.9 daucmfcputest-AluTest-LLR-009 110](#_Toc158905771)

[10.4.1.7.10 daucmfcputest-AluTest-LLR-010 110](#_Toc158905772)

[10.4.1.7.11 daucmfcputest-AluTest-LLR-011 110](#_Toc158905773)

[10.5 daucmfcrt0 110](#_Toc158905774)

[10.5.1 Crt0TransferData 110](#_Toc158905775)

[10.5.1.1 Brief Description 110](#_Toc158905776)

[10.5.1.2 List of HLRs allocated 110](#_Toc158905777)

[10.5.1.3 List of global variables accessed and modified 111](#_Toc158905778)

[10.5.1.4 Parameter list (Input/Output) 111](#_Toc158905779)

[10.5.1.5 Return Value 111](#_Toc158905780)

[10.5.1.6 Other CSUs called by this CSU 111](#_Toc158905781)

[10.5.1.7 Description of list of LLRs allocated 111](#_Toc158905782)

[10.5.1.7.1 daucmfcrt0-Crt0TransferData-LLR-001 111](#_Toc158905783)

[10.5.1.7.2 daucmfcrt0-Crt0TransferData-LLR-002 111](#_Toc158905784)

[10.5.1.7.3 daucmfcrt0-Crt0TransferData-LLR-004 112](#_Toc158905785)

[10.5.2 PowerOnRamTest 112](#_Toc158905786)

[10.5.2.1 Brief Description 112](#_Toc158905787)

[10.5.2.2 List of HLRs allocated 112](#_Toc158905788)

[10.5.2.3 List of global variables accessed and modified 112](#_Toc158905789)

[10.5.2.4 Parameter list (Input/Output) 112](#_Toc158905790)

[10.5.2.5 Return Value 113](#_Toc158905791)

[10.5.2.6 Other CSUs called by this CSU 113](#_Toc158905792)

[10.5.2.7 Description of list of LLRs allocated 113](#_Toc158905793)

[10.5.2.7.1 daucmfcrt0-PowerOnRamTest-LLR-001 113](#_Toc158905794)

[10.5.2.7.2 daucmfcrt0-PowerOnRamTest-LLR-002 113](#_Toc158905795)

[10.6 daucmfhw 114](#_Toc158905796)

[10.6.1 HwInit 114](#_Toc158905797)

[10.6.1.1 Brief Description 114](#_Toc158905798)

[10.6.1.2 List of HLRs allocated 114](#_Toc158905799)

[10.6.1.3 List of global variables accessed and modified 114](#_Toc158905800)

[10.6.1.4 Parameter list (Input/Output) 114](#_Toc158905801)

[10.6.1.5 Return Value 114](#_Toc158905802)

[10.6.1.6 Other CSUs called by this CSU 114](#_Toc158905803)

[10.6.1.7 Description of list of LLRs allocated 115](#_Toc158905804)

[10.6.1.7.1 daucmfhw-HwInit-LLR-001 115](#_Toc158905805)

[10.6.1.7.2 daucmfhw-HwInit-LLR-002 115](#_Toc158905806)

[10.6.1.7.3 daucmfhw-HwInit-LLR-003 116](#_Toc158905807)

[10.6.1.7.4 daucmfhw-HwInit-LLR-004 116](#_Toc158905808)

[10.6.1.7.5 daucmfhw-HwInit-LLR-005 116](#_Toc158905809)

[10.6.1.7.6 daucmfhw-HwInit-LLR-006 116](#_Toc158905810)

[10.6.1.7.7 daucmfhw-HwInit-LLR-007 116](#_Toc158905811)

[10.6.1.7.8 daucmfhw-HwInit-LLR-008 116](#_Toc158905812)

[10.6.1.7.9 daucmfhw-HwInit-LLR-009 116](#_Toc158905813)

[10.6.1.7.10 daucmfhw-HwInit-LLR-010 117](#_Toc158905814)

[10.6.1.7.11 daucmfhw-HwInit-LLR-011 117](#_Toc158905815)

[10.6.1.7.12 daucmfhw-HwInit-LLR-012 117](#_Toc158905816)

[10.6.1.7.13 daucmfhw-HwInit-LLR-013 117](#_Toc158905817)

[10.6.1.7.14 daucmfhw-HwInit-LLR-014 117](#_Toc158905818)

[10.6.1.7.15 daucmfhw-HwInit-LLR-015 117](#_Toc158905819)

[10.6.1.7.16 daucmfhw-HwInit-LLR-016 118](#_Toc158905820)

[10.6.1.7.17 daucmfhw-HwInit-LLR-017 118](#_Toc158905821)

[10.6.1.7.18 daucmfhw-HwInit-LLR-018 118](#_Toc158905822)

[10.6.1.7.19 daucmfhw-HwInit-LLR-019 118](#_Toc158905823)

[10.6.1.7.20 daucmfhw-HwInit-LLR-020 118](#_Toc158905824)

[10.6.1.7.21 daucmfhw-HwInit-LLR-021 118](#_Toc158905825)

[10.6.1.7.22 daucmfhw-HwInit-LLR-022 118](#_Toc158905826)

[10.6.1.7.23 daucmfhw-HwInit-LLR-023 119](#_Toc158905827)

[10.6.1.7.24 daucmfhw-HwInit-LLR-024 119](#_Toc158905828)

[10.6.1.7.25 daucmfhw-HwInit-LLR-025 119](#_Toc158905829)

[10.6.1.7.26 daucmfhw-HwInit-LLR-026 119](#_Toc158905830)

[10.6.1.7.27 daucmfhw-HwInit-LLR-027 119](#_Toc158905831)

[10.6.1.7.28 daucmfhw-HwInit-LLR-028 119](#_Toc158905832)

[10.6.1.7.29 daucmfhw-HwInit-LLR-029 120](#_Toc158905833)

[10.6.1.7.30 daucmfhw-HwInit-LLR-030 120](#_Toc158905834)

[10.6.1.7.31 daucmfhw-HwInit-LLR-034 120](#_Toc158905835)

[10.6.2 HwCopy 120](#_Toc158905836)

[10.6.2.1 Brief Description 120](#_Toc158905837)

[10.6.2.2 List of HLRs allocated 121](#_Toc158905838)

[10.6.2.3 List of global variables accessed and modified 121](#_Toc158905839)

[10.6.2.4 Parameter list (Input/Output) 121](#_Toc158905840)

[10.6.2.5 Return Value 121](#_Toc158905841)

[10.6.2.6 Other CSUs called by this CSU 121](#_Toc158905842)

[10.6.2.7 Description of list of LLRs allocated 121](#_Toc158905843)

[10.6.2.7.1 daucmfhw-HwCopy-LLR-001 121](#_Toc158905844)

[10.7 daucmfinit 121](#_Toc158905845)

[10.7.1 InitTask 122](#_Toc158905846)

[10.7.1.1 Brief Description 122](#_Toc158905847)

[10.7.1.2 List of HLRs allocated 122](#_Toc158905848)

[10.7.1.3 List of global variables accessed and modified 122](#_Toc158905849)

[10.7.1.4 Parameter list (Input/Output) 122](#_Toc158905850)

[10.7.1.5 Return Value 122](#_Toc158905851)

[10.7.1.6 Other CSUs called by this CSU 122](#_Toc158905852)

[10.7.1.7 Description of list of LLRs allocated 123](#_Toc158905853)

[10.7.1.7.1 daucmfinit-InitTask-LLR-001 123](#_Toc158905854)

[10.7.1.7.2 daucmfinit-InitTask-LLR-002 123](#_Toc158905855)

[10.7.1.7.3 daucmfinit-InitTask-LLR-003 123](#_Toc158905856)

[10.7.1.7.4 daucmfinit-InitTask-LLR-004 123](#_Toc158905857)

[10.7.1.7.5 daucmfinit-InitTask-LLR-005 123](#_Toc158905858)

[10.7.1.7.6 daucmfinit-InitTask-LLR-011 124](#_Toc158905859)

[10.7.1.7.7 daucmfinit-InitTask-LLR-012 124](#_Toc158905860)

[10.7.1.7.8 daucmfinit-InitTask-LLR-013 124](#_Toc158905861)

[10.7.2 HeartBeatInit 124](#_Toc158905862)

[10.7.2.1 Brief Description 124](#_Toc158905863)

[10.7.2.2 List of HLRs allocated 125](#_Toc158905864)

[10.7.2.3 List of global variables accessed and modified 125](#_Toc158905865)

[10.7.2.4 Parameter list (Input/Output) 125](#_Toc158905866)

[10.7.2.5 Return Value 125](#_Toc158905867)

[10.7.2.6 Other CSUs called by this CSU 125](#_Toc158905868)

[10.7.2.7 Description of list of LLRs allocated 125](#_Toc158905869)

[10.7.2.7.1 daucmfinit-HeartBeatInit-LLR-001 125](#_Toc158905870)

[10.7.3 InitInit 126](#_Toc158905871)

[10.7.3.1 Brief Description 126](#_Toc158905872)

[10.7.3.2 List of HLRs allocated 126](#_Toc158905873)

[10.7.3.3 List of global variables accessed and modified 126](#_Toc158905874)

[10.7.3.4 Parameter list (Input/Output) 126](#_Toc158905875)

[10.7.3.5 Return Value 126](#_Toc158905876)

[10.7.3.6 Other CSUs called by this CSU 126](#_Toc158905877)

[10.7.3.7 Description of list of LLRs allocated 127](#_Toc158905878)

[10.7.3.7.1 daucmfinit-InitInit-LLR-001 127](#_Toc158905879)

[10.7.3.7.2 daucmfinit-InitInit-LLR-002 127](#_Toc158905880)

[10.8 daucmfintr 127](#_Toc158905881)

[10.8.1 IntrInit 127](#_Toc158905882)

[10.8.1.1 Brief Description 127](#_Toc158905883)

[10.8.1.2 List of HLRs allocated 128](#_Toc158905884)

[10.8.1.3 List of global variables accessed and modified 128](#_Toc158905885)

[10.8.1.4 Parameter list (Input/Output) 128](#_Toc158905886)

[10.8.1.5 Return Value 128](#_Toc158905887)

[10.8.1.6 Other CSUs called by this CSU 128](#_Toc158905888)

[10.8.1.7 Description of list of LLRs allocated 128](#_Toc158905889)

[10.8.1.7.1 daucmfintr-IntrInit-LLR-001 128](#_Toc158905890)

[10.8.1.7.2 daucmfintr-IntrInit-LLR-002 129](#_Toc158905891)

[10.8.2 IntrInstall 129](#_Toc158905892)

[10.8.2.1 Brief Description 129](#_Toc158905893)

[10.8.2.2 List of HLRs allocated 129](#_Toc158905894)

[10.8.2.3 List of global variables accessed and modified 129](#_Toc158905895)

[10.8.2.4 Parameter list (Input/Output) 129](#_Toc158905896)

[10.8.2.5 Return Value 129](#_Toc158905897)

[10.8.2.6 Other CSUs called by this CSU 129](#_Toc158905898)

[10.8.2.7 Description of list of LLRs allocated 130](#_Toc158905899)

[10.8.2.7.1 daucmfintr-IntrInstall-LLR-001 130](#_Toc158905900)

[10.8.2.7.2 daucmfintr-IntrInstall-LLR-002 130](#_Toc158905901)

[10.8.2.7.3 daucmfintr-IntrInstall-LLR-003 130](#_Toc158905902)

[10.8.3 ResetIsr 130](#_Toc158905903)

[10.8.3.1 Brief Description 130](#_Toc158905904)

[10.8.3.2 List of HLRs allocated 130](#_Toc158905905)

[10.8.3.3 List of global variables accessed and modified 131](#_Toc158905906)

[10.8.3.4 Parameter list (Input/Output) 131](#_Toc158905907)

[10.8.3.5 Return Value 131](#_Toc158905908)

[10.8.3.6 Other CSUs called by this CSU 131](#_Toc158905909)

[10.8.3.7 Description of list of LLRs allocated 131](#_Toc158905910)

[10.8.3.7.1 daucmfintr-ResetIsr-LLR-001 131](#_Toc158905911)

[10.8.4 NonMaskable 131](#_Toc158905912)

[10.8.4.1 Brief Description 131](#_Toc158905913)

[10.8.4.2 List of HLRs allocated 132](#_Toc158905914)

[10.8.4.3 List of global variables accessed and modified 132](#_Toc158905915)

[10.8.4.4 Parameter list (Input/Output) 132](#_Toc158905916)

[10.8.4.5 Return Value 132](#_Toc158905917)

[10.8.4.6 Other CSUs called by this CSU 132](#_Toc158905918)

[10.8.4.7 Description of list of LLRs allocated 132](#_Toc158905919)

[10.8.4.7.1 daucmfintr-NonMaskable-LLR-001 132](#_Toc158905920)

[10.8.5 HardFault 132](#_Toc158905921)

[10.8.5.1 Brief Description 133](#_Toc158905922)

[10.8.5.2 List of HLRs allocated 133](#_Toc158905923)

[10.8.5.3 List of global variables accessed and modified 133](#_Toc158905924)

[10.8.5.4 Parameter list (Input/Output) 133](#_Toc158905925)

[10.8.5.5 Return Value 133](#_Toc158905926)

[10.8.5.6 Other CSUs called by this CSU 133](#_Toc158905927)

[10.8.5.7 Description of list of LLRs allocated 133](#_Toc158905928)

[10.8.5.7.1 daucmfintr-HardFault-LLR-001 133](#_Toc158905929)

[10.8.6 MemManage 134](#_Toc158905930)

[10.8.6.1 Brief Description 134](#_Toc158905931)

[10.8.6.2 List of HLRs allocated 134](#_Toc158905932)

[10.8.6.3 List of global variables accessed and modified 134](#_Toc158905933)

[10.8.6.4 Parameter list (Input/Output) 134](#_Toc158905934)

[10.8.6.5 Return Value 134](#_Toc158905935)

[10.8.6.6 Other CSUs called by this CSU 134](#_Toc158905936)

[10.8.6.7 Description of list of LLRs allocated 135](#_Toc158905937)

[10.8.6.7.1 daucmfintr-MemManage-LLR-001 135](#_Toc158905938)

[10.8.7 BusFault 135](#_Toc158905939)

[10.8.7.1 Brief Description 135](#_Toc158905940)

[10.8.7.2 List of HLRs allocated 135](#_Toc158905941)

[10.8.7.3 List of global variables accessed and modified 135](#_Toc158905942)

[10.8.7.4 Parameter list (Input/Output) 135](#_Toc158905943)

[10.8.7.5 Return Value 135](#_Toc158905944)

[10.8.7.6 12.14.7.6 Other CSUs called by this CSU 136](#_Toc158905945)

[10.8.7.7 Description of list of LLRs allocated 136](#_Toc158905946)

[10.8.7.7.1 daucmfintr-BusFault-LLR-001 136](#_Toc158905947)

[10.8.8 UsageFault 136](#_Toc158905948)

[10.8.8.1 Brief Description 136](#_Toc158905949)

[10.8.8.2 List of HLRs allocated 136](#_Toc158905950)

[10.8.8.3 List of global variables accessed and modified 136](#_Toc158905951)

[10.8.8.4 Parameter list (Input/Output) 136](#_Toc158905952)

[10.8.8.5 Return Value 137](#_Toc158905953)

[10.8.8.6 Other CSUs called by this CSU 137](#_Toc158905954)

[10.8.8.7 Description of list of LLRs allocated 137](#_Toc158905955)

[10.8.8.7.1 daucmfintr-UsageFault-LLR-001 137](#_Toc158905956)

[10.8.9 SpuriousInterrupt 137](#_Toc158905957)

[10.8.9.1 Brief Description 137](#_Toc158905958)

[10.8.9.2 List of HLRs allocated 137](#_Toc158905959)

[10.8.9.3 List of global variables accessed and modified 137](#_Toc158905960)

[10.8.9.4 Parameter list (Input/Output) 138](#_Toc158905961)

[10.8.9.5 Return Value 138](#_Toc158905962)

[10.8.9.6 Other CSUs called by this CSU 138](#_Toc158905963)

[10.8.9.7 Description of list of LLRs allocated 138](#_Toc158905964)

[10.8.9.7.1 daucmfintr-SpuriousInterrupt-LLR-001 138](#_Toc158905965)

[10.9 daucmfmain 138](#_Toc158905966)

[10.9.1 MainFunc 138](#_Toc158905967)

[10.9.1.1 Brief Description 139](#_Toc158905968)

[10.9.1.2 List of HLRs allocated 139](#_Toc158905969)

[10.9.1.3 List of global variables accessed and modified 139](#_Toc158905970)

[10.9.1.4 Parameter list (Input/Output) 139](#_Toc158905971)

[10.9.1.5 Return Value 139](#_Toc158905972)

[10.9.1.6 Other CSUs called by this CSU 139](#_Toc158905973)

[10.9.1.7 Description of list of LLRs allocated 139](#_Toc158905974)

[10.9.1.7.1 daucmfmain-MainFunc-LLR-001 139](#_Toc158905975)

[10.9.1.7.2 daucmfmain-MainFunc-LLR-002 140](#_Toc158905976)

[10.9.1.7.3 daucmfmain-MainFunc-LLR-003 140](#_Toc158905977)

[10.9.1.7.4 daucmfmain-MainFunc-LLR-004 140](#_Toc158905978)

[10.9.1.7.5 daucmfmain-MainFunc-LLR-005 140](#_Toc158905979)

[10.9.1.7.6 daucmfmain-MainFunc-LLR-006 140](#_Toc158905980)

[10.10 daucmfoscpu 140](#_Toc158905981)

[10.10.1 OsTaskCreate 140](#_Toc158905982)

[10.10.1.1 Brief Description 141](#_Toc158905983)

[10.10.1.2 List of HLRs allocated 141](#_Toc158905984)

[10.10.1.3 List of global variables accessed and modified 141](#_Toc158905985)

[10.10.1.4 Parameter list (Input/Output) 141](#_Toc158905986)

[10.10.1.5 Return Value 141](#_Toc158905987)

[10.10.1.6 Other CSUs called by this CSU 141](#_Toc158905988)

[10.10.1.7 Description of list of LLRs allocated 142](#_Toc158905989)

[10.10.1.7.1 daucmfoscpu-OsTaskCreate-LLR-001 142](#_Toc158905990)

[10.10.1.7.2 daucmfoscpu-OsTaskCreate-LLR-002 142](#_Toc158905991)

[10.10.1.7.3 daucmfoscpu-OsTaskCreate-LLR-003 142](#_Toc158905992)

[10.10.1.7.4 daucmfoscpu-OsTaskCreate-LLR-004 143](#_Toc158905993)

[10.11 daucmfoscpua 143](#_Toc158905994)

[10.11.1 12.17.1 OsCtxSw 143](#_Toc158905995)

[10.11.1.1 Brief Description 143](#_Toc158905996)

[10.11.1.2 List of HLRs allocated 143](#_Toc158905997)

[10.11.1.3 List of global variables accessed and modified 143](#_Toc158905998)

[10.11.1.4 Parameter list (Input/Output) 143](#_Toc158905999)

[10.11.1.5 Return Value 144](#_Toc158906000)

[10.11.1.6 Other CSUs called by this CSU 144](#_Toc158906001)

[10.11.1.7 Description of list of LLRs allocated 144](#_Toc158906002)

[10.11.1.7.1 daucmfoscpua-OsCtxSw-LLR-001 144](#_Toc158906003)

[10.11.2 OsIntCtxSw 144](#_Toc158906004)

[10.11.2.1 Brief Description 144](#_Toc158906005)

[10.11.2.2 List of HLRs allocated 144](#_Toc158906006)

[10.11.2.3 List of global variables accessed and modified 145](#_Toc158906007)

[10.11.2.4 Parameter list (Input/Output) 145](#_Toc158906008)

[10.11.2.5 Return Value 145](#_Toc158906009)

[10.11.2.6 Other CSUs called by this CSU 145](#_Toc158906010)

[10.11.2.7 Description of list of LLRs allocated 145](#_Toc158906011)

[10.11.2.7.1 daucmfoscpua-OsIntCtxSw-LLR-001 145](#_Toc158906012)

[10.11.3 SaveStatusReg 145](#_Toc158906013)

[10.11.3.1 Brief Description 145](#_Toc158906014)

[10.11.3.2 List of HLRs allocated 146](#_Toc158906015)

[10.11.3.3 List of global variables accessed and modified 146](#_Toc158906016)

[10.11.3.4 Parameter list (Input/Output) 146](#_Toc158906017)

[10.11.3.5 Return Value 146](#_Toc158906018)

[10.11.3.6 Other CSUs called by this CSU 146](#_Toc158906019)

[10.11.3.7 Description of list of LLRs allocated 146](#_Toc158906020)

[10.11.3.7.1 daucmfoscpua-SaveStatusReg-LLR-001 146](#_Toc158906021)

[10.11.4 RestoreStatusReg 147](#_Toc158906022)

[10.11.4.1 Brief Description 147](#_Toc158906023)

[10.11.4.2 List of HLRs allocated 147](#_Toc158906024)

[10.11.4.3 List of global variables accessed and modified 147](#_Toc158906025)

[10.11.4.4 Parameter list (Input/Output) 147](#_Toc158906026)

[10.11.4.5 Return Value 147](#_Toc158906027)

[10.11.4.6 Other CSUs called by this CSU 147](#_Toc158906028)

[10.11.4.7 Description of list of LLRs allocated 147](#_Toc158906029)

[10.11.4.7.1 daucmfoscpua-RestoreStatusReg-LLR-001 147](#_Toc158906030)

[10.11.5 OSStartHighRdy 148](#_Toc158906031)

[10.11.5.1 Brief Description 148](#_Toc158906032)

[10.11.5.2 List of HLRs allocated 148](#_Toc158906033)

[10.11.5.3 List of global variables accessed and modified 148](#_Toc158906034)

[10.11.5.4 Parameter list (Input/Output) 148](#_Toc158906035)

[10.11.5.5 Return Value 148](#_Toc158906036)

[10.11.5.6 Other CSUs called by this CSU 148](#_Toc158906037)

[10.11.5.7 Description of list of LLRs allocated 149](#_Toc158906038)

[10.11.5.7.1 daucmfoscpua-OsStartHighRdy-LLR-001 149](#_Toc158906039)

[10.11.5.7.2 daucmfoscpua-OsStartHighRdy-LLR-002 149](#_Toc158906040)

[10.11.5.7.3 daucmfoscpua-OsStartHighRdy-LLR-003 149](#_Toc158906041)

[10.11.5.7.4 daucmfoscpua-OsStartHighRdy-LLR-004 149](#_Toc158906042)

[10.11.5.7.5 daucmfoscpua-OsStartHighRdy-LLR-005 149](#_Toc158906043)

[10.11.5.7.6 daucmfoscpua-OsStartHighRdy-LLR-006 150](#_Toc158906044)

[10.11.6 PendSvHandler 150](#_Toc158906045)

[10.11.6.1 Brief Description 150](#_Toc158906046)

[10.11.6.2 List of HLRs allocated 150](#_Toc158906047)

[10.11.6.3 List of global variables accessed and modified 150](#_Toc158906048)

[10.11.6.4 Parameter list (Input/Output) 150](#_Toc158906049)

[10.11.6.5 Return Value 150](#_Toc158906050)

[10.11.6.6 Other CSUs called by this CSU 150](#_Toc158906051)

[10.11.6.7 Description of list of LLRs allocated 151](#_Toc158906052)

[10.11.6.7.1 daucmfoscpua-PendSvHandler-LLR-001 151](#_Toc158906053)

[10.11.6.7.2 daucmfoscpua-PendSvHandler-LLR-002 151](#_Toc158906054)

[10.11.6.7.3 daucmfoscpua-PendSvHandler-LLR-003 151](#_Toc158906055)

[10.11.6.7.4 daucmfoscpua-PendSvHandler-LLR-004 151](#_Toc158906056)

[10.11.6.7.5 daucmfoscpua-PendSvHandler-LLR-005 151](#_Toc158906057)

[10.11.6.7.6 daucmfoscpua-PendSvHandler-LLR-006 151](#_Toc158906058)

[10.11.6.7.7 daucmfoscpua-PendSvHandler-LLR-007 152](#_Toc158906059)

[10.12 daucmfpbit 152](#_Toc158906060)

[10.12.1 ConfigDataCheckData 152](#_Toc158906061)

[10.12.1.1 Brief Description 152](#_Toc158906062)

[10.12.1.2 List of HLRs allocated 152](#_Toc158906063)

[10.12.1.3 List of global variables accessed and modified 153](#_Toc158906064)

[10.12.1.4 Parameter list (Input/Output) 153](#_Toc158906065)

[10.12.1.5 Return Value 153](#_Toc158906066)

[10.12.1.6 Other CSUs called by this CSU 153](#_Toc158906067)

[10.12.1.7 Description of list of LLRs allocated 153](#_Toc158906068)

[10.12.1.7.1 daucmfpbit-ConfigDataCheckData-LLR-001 153](#_Toc158906069)

[10.12.1.7.2 daucmfpbit-ConfigDataCheckData-LLR-002 153](#_Toc158906070)

[10.12.1.7.3 daucmfpbit-ConfigDataCheckData-LLR-003 153](#_Toc158906071)

[10.12.2 CpuTest 154](#_Toc158906072)

[10.12.2.1 Brief Description 154](#_Toc158906073)

[10.12.2.2 List of HLRs allocated 154](#_Toc158906074)

[10.12.2.3 List of global variables accessed and modified 154](#_Toc158906075)

[10.12.2.4 Parameter list (Input/Output) 154](#_Toc158906076)

[10.12.2.5 Return Value 154](#_Toc158906077)

[10.12.2.6 Other CSUs called by this CSU 154](#_Toc158906078)

[10.12.2.7 Description of list of LLRs allocated 155](#_Toc158906079)

[10.12.2.7.1 daucmfpbit-CpuTest-LLR-001 155](#_Toc158906080)

[10.12.2.7.2 daucmfpbit-CpuTest-LLR-002 155](#_Toc158906081)

[10.12.2.7.3 daucmfpbit-CpuTest-LLR-003 155](#_Toc158906082)

[10.12.3 BitErrorHandler 155](#_Toc158906083)

[10.12.3.1 Brief Description 155](#_Toc158906084)

[10.12.3.2 List of HLRs allocated 155](#_Toc158906085)

[10.12.3.3 List of global variables accessed and modified 155](#_Toc158906086)

[10.12.3.4 Parameter list (Input/Output) 156](#_Toc158906087)

[10.12.3.5 Return Value 156](#_Toc158906088)

[10.12.3.6 Other CSUs called by this CSU 156](#_Toc158906089)

[10.12.3.7 Description of list of LLRs allocated 156](#_Toc158906090)

[10.12.3.7.1 daucmfpbit-BitErrorHandler-LLR-001 156](#_Toc158906091)

[10.12.3.7.2 daucmfpbit-BitErrorHandler-LLR-002 156](#_Toc158906092)

[10.12.4 PbitNvmReadWriteTest 157](#_Toc158906093)

[10.12.4.1 Brief Description 157](#_Toc158906094)

[10.12.4.2 List of HLRs allocated 157](#_Toc158906095)

[10.12.4.3 List of global variables accessed and modifed 157](#_Toc158906096)

[10.12.4.4 Parameters List (Input/Output 157](#_Toc158906097)

[10.12.4.5 Return value 157](#_Toc158906098)

[10.12.4.6 Others CSUs called by this CSU 157](#_Toc158906099)

[10.12.4.7 Description of list of LLRs allocated 157](#_Toc158906100)

[10.12.4.7.1 daucmfpbit-PbitNvmReadWriteTest-LLR-002 158](#_Toc158906101)

[10.12.4.7.2 daucmfpbit-PbitNvmReadWriteTest-LLR-003 158](#_Toc158906102)

[10.12.5 PbitCheck 158](#_Toc158906103)

[10.12.5.1 Brief Description 158](#_Toc158906104)

[10.12.5.2 List of HLRs allocated 158](#_Toc158906105)

[10.12.5.3 List of global variables accessed and modified 158](#_Toc158906106)

[10.12.5.4 Parameter list (Input/Output) 158](#_Toc158906107)

[10.12.5.5 Return Value 159](#_Toc158906108)

[10.12.5.6 Other CSUs called by this CSU 159](#_Toc158906109)

[10.12.5.7 Description of list of LLRs allocated 159](#_Toc158906110)

[10.12.5.7.1 daucmfpbit-PbitCheck-LLR-001 159](#_Toc158906111)

[10.12.5.7.2 daucmfpbit-PbitCheck-LLR-002 159](#_Toc158906112)

[10.12.5.7.3 daucmfpbit-PbitCheck-LLR-005 159](#_Toc158906113)

[10.13 daucmfrevno 159](#_Toc158906114)

[10.13.1 Brief Description 160](#_Toc158906115)

[10.13.2 List of HLRs allocated 160](#_Toc158906116)

[10.13.3 List of global variables accessed and modified 160](#_Toc158906117)

[10.13.4 Parameter list (Input/Output) 160](#_Toc158906118)

[10.13.5 Return Value 160](#_Toc158906119)

[10.13.6 Other CSUs called by this CSU 160](#_Toc158906120)

[10.13.7 Description of list of LLRs allocated 160](#_Toc158906121)

[10.13.7.1 daucmfrevno-LLR-001 160](#_Toc158906122)

[10.14 daucmfrtc 161](#_Toc158906123)

[10.14.1 BBatteryLevelIsGood 161](#_Toc158906124)

[10.14.1.1 Brief Description 161](#_Toc158906125)

[10.14.1.2 List of HLRs allocated 161](#_Toc158906126)

[10.14.1.3 List of global variables accessed and modified 161](#_Toc158906127)

[10.14.1.4 Parameter list (Input/Output) 161](#_Toc158906128)

[10.14.1.5 Return Value 161](#_Toc158906129)

[10.14.1.6 Other CSUs called by this CSU 161](#_Toc158906130)

[10.14.1.7 Description of list of LLRs allocated 162](#_Toc158906131)

[10.14.1.7.1 daucmfrtc-BBatteryLevelIsGood-LLR-001 162](#_Toc158906132)

[10.14.1.7.2 daucmfrtc-BBatteryLevelIsGood-LLR-002 162](#_Toc158906133)

[10.14.2 BIntegrityIsGood 162](#_Toc158906134)

[10.14.2.1 Brief Description 162](#_Toc158906135)

[10.14.2.2 List of HLRs allocated 163](#_Toc158906136)

[10.14.2.3 List of global variables accessed and modified 163](#_Toc158906137)

[10.14.2.4 Parameter list (Input/Output) 163](#_Toc158906138)

[10.14.2.5 Return Value 163](#_Toc158906139)

[10.14.2.6 Other CSUs called by this CSU 163](#_Toc158906140)

[10.14.2.7 Description of list of LLRs allocated 163](#_Toc158906141)

[10.14.2.7.1 daucmfrtc-BIntegrityIsGood-LLR-001 163](#_Toc158906142)

[10.14.2.7.2 daucmfrtc-BIntegrityIsGood-LLR-002 164](#_Toc158906143)

[10.14.3 RTCHWInit 164](#_Toc158906144)

[10.14.3.1 Brief Description 164](#_Toc158906145)

[10.14.3.2 List of HLRs allocated 164](#_Toc158906146)

[10.14.3.3 List of global variables accessed and modified 164](#_Toc158906147)

[10.14.3.4 Parameter list (Input/Output) 165](#_Toc158906148)

[10.14.3.5 Return Value 165](#_Toc158906149)

[10.14.3.6 Other CSUs called by this CSU 165](#_Toc158906150)

[10.14.3.7 Description of list of LLRs allocated 165](#_Toc158906151)

[10.14.3.7.1 daucmfrtc-RTCHWInit-LLR-001 165](#_Toc158906152)

[10.15 daucmfsi2c 165](#_Toc158906153)

[10.15.1 SI2CTransfer 166](#_Toc158906154)

[10.15.1.1 Brief Description 166](#_Toc158906155)

[10.15.1.2 List of HLRs allocated 166](#_Toc158906156)

[10.15.1.3 List of global variables accessed and modified 166](#_Toc158906157)

[10.15.1.4 Parameter list (Input/Output) 166](#_Toc158906158)

[10.15.1.5 Return Value 166](#_Toc158906159)

[10.15.1.6 Other CSUs called by this CSU 166](#_Toc158906160)

[10.15.1.7 Description of list of LLRs allocated 167](#_Toc158906161)

[10.15.1.7.1 daucmfsi2c-SI2CTransfer-LLR-001 167](#_Toc158906162)

[10.15.1.7.2 daucmfsi2c-SI2CTransfer-LLR-002 167](#_Toc158906163)

[10.15.1.7.3 daucmfsi2c-SI2CTransfer-LLR-003 167](#_Toc158906164)

[10.15.1.7.4 daucmfsi2c-SI2CTransfer-LLR-004 167](#_Toc158906165)

[10.15.2 Si2cInit 168](#_Toc158906166)

[10.15.2.1 Brief Description 168](#_Toc158906167)

[10.15.2.2 List of HLRs allocated 168](#_Toc158906168)

[10.15.2.3 List of global variables accessed and modified 168](#_Toc158906169)

[10.15.2.4 Parameter list (Input/Output) 168](#_Toc158906170)

[10.15.2.5 Return Value 168](#_Toc158906171)

[10.15.2.6 Other CSUs called by this CSU 168](#_Toc158906172)

[10.15.2.7 Description of list of LLRs allocated 169](#_Toc158906173)

[10.15.2.7.1 daucmfsi2c-Si2cInit-LLR-001 169](#_Toc158906174)

[10.15.2.7.2 daucmfsi2c-Si2cInit-LLR-002 169](#_Toc158906175)

[10.15.2.7.3 daucmfsi2c-Si2cInit-LLR-003 170](#_Toc158906176)

[10.15.2.7.4 daucmfsi2c-Si2cInit-LLR-004 170](#_Toc158906177)

[10.15.2.7.5 daucmfsi2c-Si2cInit-LLR-005 170](#_Toc158906178)

[10.15.3 TransferFinish 171](#_Toc158906179)

[10.15.3.1 Brief Description 171](#_Toc158906180)

[10.15.3.2 List of HLRs allocated 171](#_Toc158906181)

[10.15.3.3 List of global variables accessed and modified 171](#_Toc158906182)

[10.15.3.4 Parameter list (Input/Output) 171](#_Toc158906183)

[10.15.3.5 Return Value 171](#_Toc158906184)

[10.15.3.6 Other CSUs called by this CSU 171](#_Toc158906185)

[10.15.3.7 Description of list of LLRs allocated 171](#_Toc158906186)

[10.15.3.7.1 daucmfsi2c-TransferFinish-LLR-001 172](#_Toc158906187)

[10.15.4 Si2c2Intr 172](#_Toc158906188)

[10.15.4.1 Brief Description 172](#_Toc158906189)

[10.15.4.2 List of HLRs allocated 172](#_Toc158906190)

[10.15.4.3 List of global variables accessed and modified 172](#_Toc158906191)

[10.15.4.4 Parameter list (Input/Output) 172](#_Toc158906192)

[10.15.4.5 Return Value 172](#_Toc158906193)

[10.15.4.6 Other CSUs called by this CSU 172](#_Toc158906194)

[10.15.4.7 Description of list of LLRs allocated 173](#_Toc158906195)

[10.15.4.7.1 daucmfsi2c-Si2c2Intr-LLR-017 173](#_Toc158906196)

[10.15.4.7.2 daucmfsi2c-Si2c2Intr-LLR-001 173](#_Toc158906197)

[10.15.4.7.3 daucmfsi2c-Si2c2Intr-LLR-002 173](#_Toc158906198)

[10.15.4.7.4 daucmfsi2c-Si2c2Intr-LLR-003 173](#_Toc158906199)

[10.15.4.7.5 daucmfsi2c-Si2c2Intr-LLR-004 174](#_Toc158906200)

[10.15.4.7.6 daucmfsi2c-Si2c2Intr-LLR-005 174](#_Toc158906201)

[10.15.4.7.7 daucmfsi2c-Si2c2Intr-LLR-006 174](#_Toc158906202)

[10.15.4.7.8 daucmfsi2c-Si2c2Intr-LLR-007 174](#_Toc158906203)

[10.15.4.7.9 daucmfsi2c-Si2c2Intr-LLR-008 175](#_Toc158906204)

[10.15.4.7.10 daucmfsi2c-Si2c2Intr-LLR-009 175](#_Toc158906205)

[10.15.4.7.11 daucmfsi2c-Si2c2Intr-LLR-010 175](#_Toc158906206)

[10.15.4.7.12 daucmfsi2c-Si2c2Intr-LLR-011 176](#_Toc158906207)

[10.15.4.7.13 daucmfsi2c-Si2c2Intr-LLR-012 176](#_Toc158906208)

[10.15.4.7.14 daucmfsi2c-Si2c2Intr-LLR-013 176](#_Toc158906209)

[10.15.4.7.15 daucmfsi2c-Si2c2Intr-LLR-014 176](#_Toc158906210)

[10.15.4.7.16 daucmfsi2c-Si2c2Intr-LLR-015 176](#_Toc158906211)

[10.15.4.7.17 daucmfsi2c-Si2c2Intr-LLR-016 177](#_Toc158906212)

[10.15.5 Si2c2ErrIntr 177](#_Toc158906213)

[10.15.5.1 Brief Description 177](#_Toc158906214)

[10.15.5.2 List of HLRs allocated 177](#_Toc158906215)

[10.15.5.3 List of global variables accessed and modified 177](#_Toc158906216)

[10.15.5.4 Parameter list (Input/Output) 177](#_Toc158906217)

[10.15.5.5 Return Value 178](#_Toc158906218)

[10.15.5.6 Other CSUs called by this CSU 178](#_Toc158906219)

[10.15.5.7 Description of list of LLRs allocated 178](#_Toc158906220)

[10.15.5.7.1 daucmfsi2c-Si2c2ErrIntr-LLR-001 178](#_Toc158906221)

[10.15.5.7.2 daucmfsi2c-Si2c2ErrIntr-LLR-002 178](#_Toc158906222)

[10.15.5.7.3 daucmfsi2c-Si2c2ErrIntr-LLR-003 178](#_Toc158906223)

[10.16 daucmfrterr 179](#_Toc158906224)

[10.16.1 InitializeLeds 179](#_Toc158906225)

[10.16.1.1 Brief Description 179](#_Toc158906226)

[10.16.1.2 List of HLRs allocated 179](#_Toc158906227)

[10.16.1.3 List of global variables accessed and modified 179](#_Toc158906228)

[10.16.1.4 Parameter list (Input/Output) 179](#_Toc158906229)

[10.16.1.5 Return Value 180](#_Toc158906230)

[10.16.1.6 Other CSUs called by this CSU 180](#_Toc158906231)

[10.16.1.7 Description of list of LLRs allocated 180](#_Toc158906232)

[10.16.1.7.1 daucmfrterr-InitializeLeds-LLR-002 180](#_Toc158906233)

[10.16.1.7.2 daucmfrterr-InitializeLeds-LLR-003 180](#_Toc158906234)

[10.16.1.7.3 daucmfrterr-InitializeLeds-LLR-004 180](#_Toc158906235)

[10.16.1.7.4 daucmfrterr-InitializeLeds-LLR-005 181](#_Toc158906236)

[10.16.1.7.5 daucmfrterr-InitializeLeds-LLR-006 181](#_Toc158906237)

[10.16.2 RterrDisplayErrorId 181](#_Toc158906238)

[10.16.2.1 Brief Description 181](#_Toc158906239)

[10.16.2.2 List of HLRs allocated 181](#_Toc158906240)

[10.16.2.3 List of global variables accessed and modified 181](#_Toc158906241)

[10.16.2.4 Parameter list (Input/Output) 182](#_Toc158906242)

[10.16.2.5 Return Value 182](#_Toc158906243)

[10.16.2.6 Other CSUs called by this CSU 182](#_Toc158906244)

[10.16.2.7 Description of list of LLRs allocated 182](#_Toc158906245)

[10.16.2.7.1 daucmfrterr-RterrDisplayErrorId-LLR-001 182](#_Toc158906246)

[10.16.2.7.2 daucmfrterr-RterrDisplayErrorId-LLR-007 182](#_Toc158906247)

[10.16.2.7.3 daucmfrterr-RterrDisplayErrorId-LLR-008 182](#_Toc158906248)

[10.16.2.7.4 daucmfrterr-RterrDisplayErrorId-LLR-009 183](#_Toc158906249)

[10.16.2.7.5 daucmfrterr-RterrDisplayErrorId-LLR-010 183](#_Toc158906250)

[10.16.3 RterrForever 183](#_Toc158906251)

[10.16.3.1 Brief Description 183](#_Toc158906252)

[10.16.3.2 List of HLRs allocated 183](#_Toc158906253)

[10.16.3.3 List of global variables accessed and modified 183](#_Toc158906254)

[10.16.3.4 Parameter list (Input/Output) 184](#_Toc158906255)

[10.16.3.5 Return Value 184](#_Toc158906256)

[10.16.3.6 Other CSUs called by this CSU 184](#_Toc158906257)

[10.16.3.7 Description of list of LLRs allocated 184](#_Toc158906258)

[10.16.3.7.1 daucmfrterr-RterrForever-LLR-001 184](#_Toc158906259)

[10.16.4 CheckValidErrorFlash 184](#_Toc158906260)

[10.16.4.1 Brief Description 184](#_Toc158906261)

[10.16.4.2 List of HLRs allocated 184](#_Toc158906262)

[10.16.4.3 List of global variables accessed and modified 185](#_Toc158906263)

[10.16.4.4 Parameter list (Input/Output) 185](#_Toc158906264)

[10.16.4.5 Return Value 185](#_Toc158906265)

[10.16.4.6 Other CSUs called by this CSU 185](#_Toc158906266)

[10.16.4.7 Description of list of LLRs allocated 185](#_Toc158906267)

[10.16.4.7.1 daucmfrterr-CheckValidErrorFlash-LLR-001 185](#_Toc158906268)

[10.16.4.7.2 daucmfrterr-CheckValidErrorFlash-LLR-002 185](#_Toc158906269)

[10.16.4.7.3 daucmfrterr-CheckValidErrorFlash-LLR-003 185](#_Toc158906270)

[10.16.4.7.4 daucmfrterr-CheckValidErrorFlash-LLR-004 186](#_Toc158906271)

[10.16.4.7.5 daucmfrterr-CheckValidErrorFlash-LLR-005 186](#_Toc158906272)

[10.16.5 WriteErrorToFlash 186](#_Toc158906273)

[10.16.5.1 Brief Description 186](#_Toc158906274)

[10.16.5.2 List of HLRs allocated 186](#_Toc158906275)

[10.16.5.3 List of global variables accessed and modified 186](#_Toc158906276)

[10.16.5.4 Parameter list (Input/Output) 186](#_Toc158906277)

[10.16.5.5 Return Value 187](#_Toc158906278)

[10.16.5.6 Other CSUs called by this CSU 187](#_Toc158906279)

[10.16.5.7 Description of list of LLRs allocated 187](#_Toc158906280)

[10.16.5.7.1 daucmfrterr-WriteErrorToFlash-LLR-001 187](#_Toc158906281)

[10.16.5.7.2 daucmfrterr-WriteErrorToFlash-LLR-002 187](#_Toc158906282)

[10.16.5.7.3 daucmfrterr-WriteErrorToFlash-LLR-003 187](#_Toc158906283)

[10.16.5.7.4 daucmfrterr-WriteErrorToFlash-LLR-004 188](#_Toc158906284)

[10.16.5.7.5 daucmfrterr-WriteErrorToFlash-LLR-005 188](#_Toc158906285)

[10.16.5.7.6 daucmfrterr-WriteErrorToFlash-LLR-006 188](#_Toc158906286)

[10.16.5.7.7 daucmfrterr-WriteErrorToFlash-LLR-007 188](#_Toc158906287)

[10.16.5.7.8 daucmfrterr-WriteErrorToFlash-LLR-008 188](#_Toc158906288)

[10.16.5.7.9 daucmfrterr-WriteErrorToFlash-LLR-009 188](#_Toc158906289)

[10.16.5.7.10 daucmfrterr-WriteErrorToFlash-LLR-010 189](#_Toc158906290)

[10.16.5.7.11 daucmfrterr-WriteErrorToFlash-LLR-011 189](#_Toc158906291)

[10.16.5.7.12 daucmfrterr-WriteErrorToFlash-LLR-012 189](#_Toc158906292)

[10.16.5.7.13 daucmfrterr-WriteErrorToFlash-LLR-013 189](#_Toc158906293)

[10.16.5.7.14 daucmfrterr-WriteErrorToFlash-LLR-014 189](#_Toc158906294)

[10.16.5.7.15 daucmfrterr-WriteErrorToFlash-LLR-015 189](#_Toc158906295)

[10.16.5.7.16 daucmfrterr-WriteErrorToFlash-LLR-016 190](#_Toc158906296)

[10.16.5.7.17 daucmfrterr-WriteErrorToFlash-LLR-017 190](#_Toc158906297)

[10.16.5.7.18 daucmfrterr-WriteErrorToFlash-LLR-018 190](#_Toc158906298)

[10.16.5.7.19 daucmfrterr-WriteErrorToFlash-LLR-019 190](#_Toc158906299)

[10.16.5.7.20 daucmfrterr-WriteErrorToFlash-LLR-020 190](#_Toc158906300)

[10.16.6 EraseErrSection 190](#_Toc158906301)

[10.16.6.1 Brief Description 191](#_Toc158906302)

[10.16.6.2 List of HLRs allocated 191](#_Toc158906303)

[10.16.6.3 List of global variables accessed and modified 191](#_Toc158906304)

[10.16.6.4 Parameter list (Input/Output) 191](#_Toc158906305)

[10.16.6.5 Return Value 191](#_Toc158906306)

[10.16.6.6 Other CSUs called by this CSU 191](#_Toc158906307)

[10.16.6.7 Description of list of LLRs allocated 191](#_Toc158906308)

[10.16.6.7.1 daucmfrterr-EraseErrSection-LLR-001 191](#_Toc158906309)

[10.16.6.7.2 daucmfrterr-EraseErrSection-LLR-002 192](#_Toc158906310)

[10.16.6.7.3 daucmfrterr-EraseErrSection-LLR-003 192](#_Toc158906311)

[10.16.6.7.4 daucmfrterr-EraseErrSection-LLR-004 192](#_Toc158906312)

[10.16.6.7.5 daucmfrterr-EraseErrSection-LLR-005 192](#_Toc158906313)

[10.16.7 ProgramWord 192](#_Toc158906314)

[10.16.7.1 Brief Description 193](#_Toc158906315)

[10.16.7.2 List of HLRs allocated 193](#_Toc158906316)

[10.16.7.3 List of global variables accessed and modified 193](#_Toc158906317)

[10.16.7.4 Parameter list (Input/Output) 193](#_Toc158906318)

[10.16.7.5 Return Value 193](#_Toc158906319)

[10.16.7.6 Other CSUs called by this CSU 193](#_Toc158906320)

[10.16.7.7 Description of list of LLRs allocated 193](#_Toc158906321)

[10.16.7.7.1 daucmfrterr-ProgramWord-LLR-001 193](#_Toc158906322)

[10.16.7.7.2 daucmfrterr-ProgramWord-LLR-002 194](#_Toc158906323)

[10.16.7.7.3 daucmfrterr-ProgramWord-LLR-003 194](#_Toc158906324)

[10.16.7.7.4 daucmfrterr-ProgramWord-LLR-004 194](#_Toc158906325)

[10.16.7.7.5 daucmfrterr-ProgramWord-LLR-005 194](#_Toc158906326)

[10.17 daucmftbase 194](#_Toc158906327)

[10.17.1 TbaseTaskSignaling 194](#_Toc158906328)

[10.17.1.1 Brief Description 195](#_Toc158906329)

[10.17.1.2 List of HLRs allocated 195](#_Toc158906330)

[10.17.1.3 List of global variables accessed and modified 195](#_Toc158906331)

[10.17.1.4 Parameter list (Input/Output) 195](#_Toc158906332)

[10.17.1.5 Return Value 195](#_Toc158906333)

[10.17.1.6 Other CSUs called by this CSU 195](#_Toc158906334)

[10.17.1.7 Description of list of LLRs allocated 195](#_Toc158906335)

[10.17.1.7.1 daucmftbase-TbaseTaskSignaling-LLR-001 196](#_Toc158906336)

[10.17.1.7.2 daucmftbase-TbaseTaskSignaling-LLR-002 196](#_Toc158906337)

[10.17.1.7.3 daucmftbase-TbaseTaskSignaling-LLR-003 196](#_Toc158906338)

[10.17.1.7.4 daucmftbase-TbaseTaskSignaling-LLR-004 196](#_Toc158906339)

[10.17.1.7.5 daucmftbase-TbaseTaskSignaling-LLR-005 196](#_Toc158906340)

[10.17.2 TbaseIntrHandler 196](#_Toc158906341)

[10.17.2.1 Brief Description 196](#_Toc158906342)

[10.17.2.2 List of HLRs allocated 196](#_Toc158906343)

[10.17.2.3 List of global variables accessed and modified 197](#_Toc158906344)

[10.17.2.4 Parameter list (Input/Output) 197](#_Toc158906345)

[10.17.2.5 Return Value 197](#_Toc158906346)

[10.17.2.6 Other CSUs called by this CSU 197](#_Toc158906347)

[10.17.2.7 Description of list of LLRs allocated 197](#_Toc158906348)

[10.17.2.7.1 daucmftbase-TbaseIntrHandler-LLR-001 197](#_Toc158906349)

[10.17.2.7.2 daucmftbase-TbaseIntrHandler-LLR-002 197](#_Toc158906350)

[10.17.2.7.3 daucmftbase-TbaseIntrHandler-LLR-003 198](#_Toc158906351)

[10.17.2.7.4 daucmftbase-TbaseIntrHandler-LLR-004 198](#_Toc158906352)

[10.17.3 TbaseInit 198](#_Toc158906353)

[10.17.3.1 Brief Description 198](#_Toc158906354)

[10.17.3.2 List of HLRs allocated 198](#_Toc158906355)

[10.17.3.3 List of global variables accessed and modified 198](#_Toc158906356)

[10.17.3.4 Parameter list (Input/Output) 199](#_Toc158906357)

[10.17.3.5 Return Value 199](#_Toc158906358)

[10.17.3.6 Other CSUs called by this CSU 199](#_Toc158906359)

[10.17.3.7 Description of list of LLRs allocated 199](#_Toc158906360)

[10.17.3.7.1 daucmftbase-TbaseInit-LLR-001 199](#_Toc158906361)

[10.18 daucmfucos 199](#_Toc158906362)

[10.18.1 OsInit 199](#_Toc158906363)

[10.18.1.1 Brief Description 199](#_Toc158906364)

[10.18.1.2 List of HLRs allocated 200](#_Toc158906365)

[10.18.1.3 List of global variables accessed and modified 200](#_Toc158906366)

[10.18.1.4 Parameter list (Input/Output) 200](#_Toc158906367)

[10.18.1.5 Return Value 200](#_Toc158906368)

[10.18.1.6 Other CSUs called by this CSU 200](#_Toc158906369)

[10.18.1.7 Description of list of LLRs allocated 200](#_Toc158906370)

[10.18.1.7.1 daucmfucos-OsInit-LLR-001 200](#_Toc158906371)

[10.18.1.7.2 daucmfucos-OsInit-LLR-002 201](#_Toc158906372)

[10.18.2 OsTaskIdle 201](#_Toc158906373)

[10.18.2.1 Brief Description 201](#_Toc158906374)

[10.18.2.2 List of HLRs allocated 201](#_Toc158906375)

[10.18.2.3 List of global variables accessed and modified 202](#_Toc158906376)

[10.18.2.4 Parameter list (Input/Output) 202](#_Toc158906377)

[10.18.2.5 Return Value 202](#_Toc158906378)

[10.18.2.6 Other CSUs called by this CSU 202](#_Toc158906379)

[10.18.2.7 Description of list of LLRs allocated 202](#_Toc158906380)

[10.18.2.7.1 daucmfucos-OsTaskIdle-LLR-001 202](#_Toc158906381)

[10.18.3 OsStart 202](#_Toc158906382)

[10.18.3.1 Brief Description 202](#_Toc158906383)

[10.18.3.2 List of HLRs allocated 203](#_Toc158906384)

[10.18.3.3 List of global variables accessed and modified 203](#_Toc158906385)

[10.18.3.4 Parameter list (Input/Output) 203](#_Toc158906386)

[10.18.3.5 Return Value 203](#_Toc158906387)

[10.18.3.6 Other CSUs called by this CSU 203](#_Toc158906388)

[10.18.3.7 Description of list of LLRs allocated 203](#_Toc158906389)

[10.18.3.7.1 daucmfucos-OsStart-LLR-001 203](#_Toc158906390)

[10.18.3.7.2 daucmfucos-OsStart-LLR-002 204](#_Toc158906391)

[10.18.3.7.3 daucmfucos-OsStart-LLR-003 204](#_Toc158906392)

[10.18.3.7.4 daucmfucos-OsStart-LLR-004 204](#_Toc158906393)

[10.18.4 OsSched 204](#_Toc158906394)

[10.18.4.1 Brief Description 204](#_Toc158906395)

[10.18.4.2 List of HLRs allocated 204](#_Toc158906396)

[10.18.4.3 List of global variables accessed and modified 205](#_Toc158906397)

[10.18.4.4 Parameter list (Input/Output) 205](#_Toc158906398)

[10.18.4.5 Return Value 205](#_Toc158906399)

[10.18.4.6 Other CSUs called by this CSU 205](#_Toc158906400)

[10.18.4.7 Description of list of LLRs allocated 205](#_Toc158906401)

[10.18.4.7.1 daucmfucos-OsSched-LLR-001 205](#_Toc158906402)

[10.18.4.7.2 daucmfucos-OsSched-LLR-002 205](#_Toc158906403)

[10.18.4.7.3 daucmfucos-OsSched-LLR-003 206](#_Toc158906404)

[10.18.4.7.4 daucmfucos-OsSched-LLR-004 206](#_Toc158906405)

[10.18.5 OsTcbInit 206](#_Toc158906406)

[10.18.5.1 Brief Description 206](#_Toc158906407)

[10.18.5.2 List of HLRs allocated 206](#_Toc158906408)

[10.18.5.3 List of global variables accessed and modified 206](#_Toc158906409)

[10.18.5.4 Parameter list (Input/Output) 207](#_Toc158906410)

[10.18.5.5 Return Value 207](#_Toc158906411)

[10.18.5.6 Other CSUs called by this CSU 207](#_Toc158906412)

[10.18.5.7 Description of list of LLRs allocated 207](#_Toc158906413)

[10.18.5.7.1 daucmfucos-OsTcbInit-LLR-001 207](#_Toc158906414)

[10.18.5.7.2 daucmfucos-OsTcbInit-LLR-002 207](#_Toc158906415)

[10.18.5.7.3 daucmfucos-OsTcbInit-LLR-003 207](#_Toc158906416)

[10.18.5.7.4 daucmfucos-OsTcbInit-LLR-004 208](#_Toc158906417)

[10.18.6 OsIntEnter 208](#_Toc158906418)

[10.18.6.1 Brief Description 208](#_Toc158906419)

[10.18.6.2 List of HLRs allocated 209](#_Toc158906420)

[10.18.6.3 List of global variables accessed and modified 209](#_Toc158906421)

[10.18.6.4 Parameter list (Input/Output) 209](#_Toc158906422)

[10.18.6.5 Return Value 209](#_Toc158906423)

[10.18.6.6 Other CSUs called by this CSU 209](#_Toc158906424)

[10.18.6.7 Description of list of LLRs allocated 209](#_Toc158906425)

[10.18.6.7.1 daucmfucos-OsIntEnter-LLR-001 209](#_Toc158906426)

[10.18.7 OsIntExit 210](#_Toc158906427)

[10.18.7.1 Brief Description 210](#_Toc158906428)

[10.18.7.2 List of HLRs allocated 210](#_Toc158906429)

[10.18.7.3 List of global variables accessed and modified 210](#_Toc158906430)

[10.18.7.4 Parameter list (Input/Output) 210](#_Toc158906431)

[10.18.7.5 Return Value 210](#_Toc158906432)

[10.18.7.6 Other CSUs called by this CSU 210](#_Toc158906433)

[10.18.7.7 Description of list of LLRs allocated 211](#_Toc158906434)

[10.18.7.7.1 daucmfucos-OsIntExit-LLR-001 211](#_Toc158906435)

[10.18.7.7.2 daucmfucos-OsIntExit-LLR-002 211](#_Toc158906436)

[10.18.7.7.3 daucmfucos-OsIntExit-LLR-003 211](#_Toc158906437)

[10.18.7.7.4 daucmfucos-OsIntExit-LLR-004 211](#_Toc158906438)

[10.18.8 OsTimeDly 211](#_Toc158906439)

[10.18.8.1 Brief Description 212](#_Toc158906440)

[10.18.8.2 List of HLRs allocated 212](#_Toc158906441)

[10.18.8.3 List of global variables accessed and modified 212](#_Toc158906442)

[10.18.8.4 Parameter list (Input/Output) 212](#_Toc158906443)

[10.18.8.5 Return Value 212](#_Toc158906444)

[10.18.8.6 Other CSUs called by this CSU 212](#_Toc158906445)

[10.18.8.7 Description of list of LLRs allocated 212](#_Toc158906446)

[10.18.8.7.1 daucmfucos-OsTimeDly-LLR-001 213](#_Toc158906447)

[10.18.8.7.2 daucmfucos-OsTimeDly-LLR-002 213](#_Toc158906448)

[10.18.9 OsTimeTick 213](#_Toc158906449)

[10.18.9.1 Brief Description 213](#_Toc158906450)

[10.18.9.2 List of HLRs allocated 213](#_Toc158906451)

[10.18.9.3 List of global variables accessed and modified 213](#_Toc158906452)

[10.18.9.4 Parameter list (Input/Output) 214](#_Toc158906453)

[10.18.9.5 Return Value 214](#_Toc158906454)

[10.18.9.6 Other CSUs called by this CSU 214](#_Toc158906455)

[10.18.9.7 Description of list of LLRs allocated 214](#_Toc158906456)

[10.18.9.7.1 daucmfucos-OsTimeTick-LLR-001 214](#_Toc158906457)

[10.18.9.7.2 daucmfucos-OsTimeTick-LLR-002 214](#_Toc158906458)

[10.18.9.7.3 daucmfucos-OsTimeTick-LLR-003 215](#_Toc158906459)

[10.18.9.7.4 daucmfucos-OsTimeTick-LLR-004 215](#_Toc158906460)

[10.18.10 OsSemCreate 215](#_Toc158906461)

[10.18.10.1 Brief Description 215](#_Toc158906462)

[10.18.10.2 List of HLRs allocated 215](#_Toc158906463)

[10.18.10.3 List of global variables accessed and modified 215](#_Toc158906464)

[10.18.10.4 Parameter list (Input/Output) 215](#_Toc158906465)

[10.18.10.5 Return Value 216](#_Toc158906466)

[10.18.10.6 Other CSUs called by this CSU 216](#_Toc158906467)

[10.18.10.7 Description of list of LLRs allocated 216](#_Toc158906468)

[10.18.10.7.1 daucmfucos-OsSemCreate-LLR-001 216](#_Toc158906469)

[10.18.10.7.2 daucmfucos-OsSemCreate-LLR-002 216](#_Toc158906470)

[10.18.10.7.3 daucmfucos-OsSemCreate-LLR-003 216](#_Toc158906471)

[10.18.10.7.4 daucmfucos-OsSemCreate-LLR-004 216](#_Toc158906472)

[10.18.10.7.5 daucmfucos-OsSemCreate-LLR-005 216](#_Toc158906473)

[10.18.11 OsSemPend 217](#_Toc158906474)

[10.18.11.1 Brief Description 217](#_Toc158906475)

[10.18.11.2 List of HLRs allocated 217](#_Toc158906476)

[10.18.11.3 List of global variables accessed and modified 217](#_Toc158906477)

[10.18.11.4 Parameter list (Input/Output) 217](#_Toc158906478)

[10.18.11.5 Return Value 218](#_Toc158906479)

[10.18.11.6 Other CSUs called by this CSU 218](#_Toc158906480)

[10.18.11.7 Description of list of LLRs allocated 218](#_Toc158906481)

[10.18.11.7.1 daucmfucos-OsSemPend-LLR-001 218](#_Toc158906482)

[10.18.11.7.2 daucmfucos-OsSemPend-LLR-002 218](#_Toc158906483)

[10.18.11.7.3 daucmfucos-OsSemPend-LLR-003 218](#_Toc158906484)

[10.18.12 OsSemPost 219](#_Toc158906485)

[10.18.12.1 Brief Description 219](#_Toc158906486)

[10.18.12.2 List of HLRs allocated 219](#_Toc158906487)

[10.18.12.3 List of global variables accessed and modified 219](#_Toc158906488)

[10.18.12.4 Parameter list (Input/Output) 220](#_Toc158906489)

[10.18.12.5 Return Value 220](#_Toc158906490)

[10.18.12.6 Other CSUs called by this CSU 220](#_Toc158906491)

[10.18.12.7 Description of list of LLRs allocated 220](#_Toc158906492)

[10.18.12.7.1 daucmfucos-OsSemPost-LLR-001 220](#_Toc158906493)

[10.18.12.7.2 daucmfucos-OsSemPost-LLR-002 220](#_Toc158906494)

[10.18.12.7.3 daucmfucos-OsSemPost-LLR-003 221](#_Toc158906495)

[10.18.12.7.4 daucmfucos-OsSemPost-LLR-004 221](#_Toc158906496)

[10.19 daucmfwdog 222](#_Toc158906497)

[10.19.1 WdogInit 222](#_Toc158906498)

[10.19.1.1 Brief Description 222](#_Toc158906499)

[10.19.1.2 List of HLRs allocated 222](#_Toc158906500)

[10.19.1.3 List of global variables accessed and modified 222](#_Toc158906501)

[10.19.1.4 Parameter list (Input/Output) 222](#_Toc158906502)

[10.19.1.5 Return Value 222](#_Toc158906503)

[10.19.1.6 Other CSUs called by this CSU 222](#_Toc158906504)

[10.19.1.7 Description of list of LLRs allocated 223](#_Toc158906505)

[10.19.1.7.1 daucmfwdog-WdogInit-LLR-001 223](#_Toc158906506)

[10.19.1.7.2 daucmfwdog-WdogInit-LLR-002 223](#_Toc158906507)

[10.19.1.7.3 daucmfwdog-WdogInit-LLR-003 223](#_Toc158906508)

[10.19.1.7.4 daucmfwdog-WdogInit-LLR-004 223](#_Toc158906509)

[10.19.1.7.5 daucmfwdog-WdogInit-LLR-005 223](#_Toc158906510)

[10.19.1.7.6 daucmfwdog-WdogInit-LLR-006 224](#_Toc158906511)

[10.19.1.7.7 daucmfwdog-WdogInit-LLR-007 224](#_Toc158906512)

[10.19.1.7.8 daucmfwdog-WdogInit-LLR-008 224](#_Toc158906513)

[10.19.2 WdogKickWatchDog 224](#_Toc158906514)

[10.19.2.1 Brief Description 224](#_Toc158906515)

[10.19.2.2 List of HLRs allocated 224](#_Toc158906516)

[10.19.2.3 List of global variables accessed and modified 224](#_Toc158906517)

[10.19.2.4 Parameter list (Input/Output) 224](#_Toc158906518)

[10.19.2.5 Return Value 225](#_Toc158906519)

[10.19.2.6 Other CSUs called by this CSU 225](#_Toc158906520)

[10.19.2.7 Description of list of LLRs allocated 225](#_Toc158906521)

[10.19.2.7.1 daucmfwdog-WdogKickWatchDog-LLR-001 225](#_Toc158906522)

[10.20 daucmfxram 225](#_Toc158906523)

[10.20.1 XramInit 225](#_Toc158906524)

[10.20.1.1 Brief Description 225](#_Toc158906525)

[10.20.1.2 List of HLRs allocated 225](#_Toc158906526)

[10.20.1.3 List of global variables accessed and modified 226](#_Toc158906527)

[10.20.1.4 Parameter list (Input/Output) 226](#_Toc158906528)

[10.20.1.5 Return Value 226](#_Toc158906529)

[10.20.1.6 Other CSUs called by this CSU 226](#_Toc158906530)

[10.20.1.7 Description of list of LLRs allocated 226](#_Toc158906531)

[10.20.1.7.1 daucmfxram-XramInit-LLR-001 226](#_Toc158906532)

[10.20.1.7.2 daucmfxram-XramInit-LLR-002 227](#_Toc158906533)

[10.20.1.7.3 daucmfxram-XramInit-LLR-003 227](#_Toc158906534)

[10.20.1.7.4 daucmfxram-XramInit-LLR-004 228](#_Toc158906535)

[10.20.1.7.5 daucmfxram-XramInit-LLR-005 228](#_Toc158906536)

[10.20.1.7.6 daucmfxram-XramInit-LLR-006 228](#_Toc158906537)

[10.20.1.7.7 daucmfxram-XramInit-LLR-007 228](#_Toc158906538)

[10.20.1.7.8 daucmfxram-XramInit-LLR-008 229](#_Toc158906539)

[10.20.1.7.9 daucmfxram-XramInit-LLR-009 229](#_Toc158906540)

[10.20.1.7.10 daucmfxram-XramInit-LLR-010 230](#_Toc158906541)

[10.20.1.7.11 daucmfxram-XramInit-LLR-011 230](#_Toc158906542)

[10.20.1.7.12 daucmfxram-XramInit-LLR-012 230](#_Toc158906543)

[10.20.1.7.13 daucmfxram-XramInit-LLR-013 231](#_Toc158906544)

[10.20.1.7.14 daucmfxram-XramInit-LLR-014 231](#_Toc158906545)

[10.20.1.7.15 daucmfxram-XramInit-LLR-015 231](#_Toc158906546)

[10.20.1.7.16 daucmfxram-XramInit-LLR-016 231](#_Toc158906547)

[10.20.1.7.17 daucmfxram-XramInit-LLR-017 232](#_Toc158906548)

[10.20.1.7.18 daucmfxram-XramInit-LLR-018 233](#_Toc158906549)

[10.20.1.7.19 daucmfxram-XramInit-LLR-019 233](#_Toc158906550)

[10.20.1.7.20 IsOlder 233](#_Toc158906551)

[10.20.1.7.20.1 Brief Description 233](#_Toc158906552)

[10.20.1.7.20.2 List of HLRs allocated 233](#_Toc158906553)

[10.20.1.7.20.3 List of global variables accessed and modified 233](#_Toc158906554)

[10.20.1.7.20.4 Parameter list (Input/Output) 234](#_Toc158906555)

[10.20.1.7.20.5 Return Value 234](#_Toc158906556)

[10.20.1.7.20.6 Other CSUs called by this CSU 234](#_Toc158906557)

[10.20.1.7.20.7 Description of list of LLRs allocated 234](#_Toc158906558)

[10.20.1.7.20.7.1 daucmfxram-IsOlder-LLR-001 234](#_Toc158906559)

[10.20.1.7.20.7.2 daucmfxram-IsOlder-LLR-002 234](#_Toc158906560)

[10.20.1.7.20.7.3 daucmfxram-IsOlder-LLR-003 235](#_Toc158906561)

[10.20.1.7.20.7.4 daucmfxram-IsOlder-LLR-004 235](#_Toc158906562)

[10.20.1.7.20.7.5 daucmfxram-IsOlder-LLR-005 235](#_Toc158906563)

[10.20.1.7.20.7.6 daucmfxram-IsOlder-LLR-006 235](#_Toc158906564)

[10.20.1.7.20.7.7 daucmfxram-IsOlder-LLR-007 236](#_Toc158906565)

[10.20.1.7.21 InitNvram 236](#_Toc158906566)

[10.20.1.7.21.1 Brief Description 236](#_Toc158906567)

[10.20.1.7.21.2 List of HLRs allocated 236](#_Toc158906568)

[10.20.1.7.21.3 List of global variables accessed and modified 236](#_Toc158906569)

[10.20.1.7.21.4 Parameter list (Input/Output) 236](#_Toc158906570)

[10.20.1.7.21.5 Return Value 237](#_Toc158906571)

[10.20.1.7.21.6 Other CSUs called by this CSU 237](#_Toc158906572)

[10.20.1.7.21.7 Description of list of LLRs allocated 237](#_Toc158906573)

[10.20.1.7.21.7.1 daucmfxram-InitNvram-LLR-001 237](#_Toc158906574)

[10.20.1.7.21.7.2 daucmfxram-InitNvram-LLR-002 237](#_Toc158906575)

[10.20.1.7.21.7.3 daucmfxram-InitNvram-LLR-003 238](#_Toc158906576)

[10.20.1.7.21.7.4 daucmfxram-InitNvram-LLR-004 238](#_Toc158906577)

[10.20.1.7.21.7.5 daucmfxram-InitNvram-LLR-005 239](#_Toc158906578)

[11 Software Low Level Requirements-CMU Flight Module Library 239](#_Toc158906579)

[11.1 daulibmisc 239](#_Toc158906580)

[11.1.1 NvicPriorityGroupConfig 239](#_Toc158906581)

[11.1.1.1 Brief Description 239](#_Toc158906582)

[11.1.1.2 List of HLRs allocated 239](#_Toc158906583)

[11.1.1.3 List of global variables accessed and modified 240](#_Toc158906584)

[11.1.1.4 Parameter list (Input/Output) 240](#_Toc158906585)

[11.1.1.5 Return Value 240](#_Toc158906586)

[11.1.1.6 Other CSUs called by this CSU 240](#_Toc158906587)

[11.1.1.7 Description of list of LLRs allocated 240](#_Toc158906588)

[11.1.1.7.1 daulibmisc-NvicPriorityGroupConfig-LLR-001 240](#_Toc158906589)

[11.1.2 NvicInit 240](#_Toc158906590)

[11.1.2.1 Brief Description 240](#_Toc158906591)

[11.1.2.2 List of HLRs allocated 241](#_Toc158906592)

[11.1.2.3 List of global variables accessed and modified 241](#_Toc158906593)

[11.1.2.4 Parameter list (Input/Output) 241](#_Toc158906594)

[11.1.2.5 Return Value 241](#_Toc158906595)

[11.1.2.6 Other CSUs called by this CSU 241](#_Toc158906596)

[11.1.2.7 Description of list of LLRs allocated 241](#_Toc158906597)

[11.1.2.7.1 daulibmisc-NvicInit-LLR-001 241](#_Toc158906598)

[11.1.2.7.2 daulibmisc-NvicInit-LLR-002 242](#_Toc158906599)

[11.1.2.7.3 daulibmisc-NvicInit-LLR-003 242](#_Toc158906600)

[11.1.3 NvicSetVectorTable 242](#_Toc158906601)

[11.1.3.1 Brief Description 242](#_Toc158906602)

[11.1.3.2 List of HLRs allocated 242](#_Toc158906603)

[11.1.3.3 List of global variables accessed and modified 243](#_Toc158906604)

[11.1.3.4 Parameter list (Input/Output) 243](#_Toc158906605)

[11.1.3.5 Return Value 243](#_Toc158906606)

[11.1.3.6 Other CSUs called by this CSU 243](#_Toc158906607)

[11.1.3.7 Description of list of LLRs allocated 243](#_Toc158906608)

[11.1.3.7.1 daulibmisc-NvicSetVectorTable-LLR-001 243](#_Toc158906609)

[11.2 daulibstm32f4xxcan 243](#_Toc158906610)

[11.2.1 CanDeInit 244](#_Toc158906611)

[11.2.1.1 Brief Description 244](#_Toc158906612)

[11.2.1.2 List of HLRs allocated 244](#_Toc158906613)

[11.2.1.3 List of global variables accessed and modified 244](#_Toc158906614)

[11.2.1.4 Parameter list (Input/Output) 244](#_Toc158906615)

[11.2.1.5 Return Value 244](#_Toc158906616)

[11.2.1.6 Other CSUs called by this CSU 244](#_Toc158906617)

[11.2.1.7 Description of list of LLRs allocated 244](#_Toc158906618)

[11.2.1.7.1 daulibstm32f4xxcan-CanDeInit-LLR-001 245](#_Toc158906619)

[11.2.1.7.2 daulibstm32f4xxcan-CanDeInit-LLR-002 245](#_Toc158906620)

[11.2.2 CanInit 245](#_Toc158906621)

[11.2.2.1 Brief Description 245](#_Toc158906622)

[11.2.2.2 List of HLRs allocated 245](#_Toc158906623)

[11.2.2.3 List of global variables accessed and modified 245](#_Toc158906624)

[11.2.2.4 Parameter list (Input/Output) 245](#_Toc158906625)

[11.2.2.5 Return Value 246](#_Toc158906626)

[11.2.2.6 Other CSUs called by this CSU 246](#_Toc158906627)

[11.2.2.7 Description of list of LLRs allocated 246](#_Toc158906628)

[11.2.2.7.1 daulibstm32f4xxcan-CanInit-LLR-001 246](#_Toc158906629)

[11.2.2.7.2 daulibstm32f4xxcan-CanInit-LLR-002 246](#_Toc158906630)

[11.2.2.7.3 daulibstm32f4xxcan-CanInit-LLR-003 246](#_Toc158906631)

[11.2.2.7.4 daulibstm32f4xxcan-CanInit-LLR-004 247](#_Toc158906632)

[11.2.2.7.5 daulibstm32f4xxcan-CanInit-LLR-005 247](#_Toc158906633)

[11.2.2.7.6 daulibstm32f4xxcan-CanInit-LLR-006 247](#_Toc158906634)

[11.2.2.7.7 daulibstm32f4xxcan-CanInit-LLR-007 247](#_Toc158906635)

[11.2.2.7.8 daulibstm32f4xxcan-CanInit-LLR-008 247](#_Toc158906636)

[11.2.2.7.9 daulibstm32f4xxcan-CanInit-LLR-009 247](#_Toc158906637)

[11.2.2.7.10 daulibstm32f4xxcan-CanInit-LLR-010 248](#_Toc158906638)

[11.2.2.7.11 daulibstm32f4xxcan-CanInit-LLR-011 248](#_Toc158906639)

[11.2.2.7.12 daulibstm32f4xxcan-CanInit-LLR-012 248](#_Toc158906640)

[11.2.2.7.13 daulibstm32f4xxcan-CanInit-LLR-013 248](#_Toc158906641)

[11.2.2.7.14 daulibstm32f4xxcan-CanInit-LLR-014 248](#_Toc158906642)

[11.2.2.7.15 daulibstm32f4xxcan-CanInit-LLR-015 249](#_Toc158906643)

[11.2.2.7.16 daulibstm32f4xxcan-CanInit-LLR-016 249](#_Toc158906644)

[11.2.2.7.17 daulibstm32f4xxcan-CanInit-LLR-017 249](#_Toc158906645)

[11.2.2.7.18 daulibstm32f4xxcan-CanInit-LLR-018 249](#_Toc158906646)

[11.2.2.7.19 daulibstm32f4xxcan-CanInit-LLR-019 250](#_Toc158906647)

[11.2.2.7.20 daulibstm32f4xxcan-CanInit-LLR-020 250](#_Toc158906648)

[11.2.3 CanFilterInit 250](#_Toc158906649)

[11.2.3.1 Brief Description 250](#_Toc158906650)

[11.2.3.2 List of HLRs allocated 250](#_Toc158906651)

[11.2.3.3 List of global variables accessed and modified 251](#_Toc158906652)

[11.2.3.4 Parameter list (Input/Output) 251](#_Toc158906653)

[11.2.3.5 Return Value 251](#_Toc158906654)

[11.2.3.6 Other CSUs called by this CSU 251](#_Toc158906655)

[11.2.3.7 Description of list of LLRs allocated 251](#_Toc158906656)

[11.2.3.7.1 daulibstm32f4xxcan-CanFilterInit-LLR-001 251](#_Toc158906657)

[11.2.3.7.2 daulibstm32f4xxcan-CanFilterInit-LLR-002 251](#_Toc158906658)

[11.2.3.7.3 daulibstm32f4xxcan-CanFilterInit-LLR-003 251](#_Toc158906659)

[11.2.3.7.4 daulibstm32f4xxcan-CanFilterInit-LLR-004 252](#_Toc158906660)

[11.2.3.7.5 daulibstm32f4xxcan-CanFilterInit-LLR-005 252](#_Toc158906661)

[11.2.3.7.6 daulibstm32f4xxcan-CanFilterInit-LLR-006 253](#_Toc158906662)

[11.2.3.7.7 daulibstm32f4xxcan-CanFilterInit-LLR-007 253](#_Toc158906663)

[11.2.3.7.8 daulibstm32f4xxcan-CanFilterInit-LLR-008 253](#_Toc158906664)

[11.2.3.7.9 daulibstm32f4xxcan-CanFilterInit-LLR-009 253](#_Toc158906665)

[11.2.3.7.10 daulibstm32f4xxcan-CanFilterInit-LLR-010 253](#_Toc158906666)

[11.2.4 CanTransmit 253](#_Toc158906667)

[11.2.4.1 Brief Description 254](#_Toc158906668)

[11.2.4.2 List of HLRs allocated 254](#_Toc158906669)

[11.2.4.3 List of global variables accessed and modified 254](#_Toc158906670)

[11.2.4.4 Parameter list (Input/Output) 254](#_Toc158906671)

[11.2.4.5 Return Value 254](#_Toc158906672)

[11.2.4.6 Other CSUs called by this CSU 254](#_Toc158906673)

[11.2.4.7 Description of list of LLRs allocated 254](#_Toc158906674)

[11.2.4.7.1 daulibstm32f4xxcan-CanTransmit-LLR-001 254](#_Toc158906675)

[11.2.4.7.2 daulibstm32f4xxcan-CanTransmit-LLR-002 255](#_Toc158906676)

[11.2.4.7.3 daulibstm32f4xxcan-CanTransmit-LLR-003 255](#_Toc158906677)

[11.2.4.7.4 daulibstm32f4xxcan-CanTransmit-LLR-004 255](#_Toc158906678)

[11.2.4.7.5 daulibstm32f4xxcan-CanTransmit-LLR-005 255](#_Toc158906679)

[11.2.4.7.6 daulibstm32f4xxcan-CanTransmit-LLR-006 255](#_Toc158906680)

[11.2.4.7.7 daulibstm32f4xxcan-CanTransmit-LLR-007 256](#_Toc158906681)

[11.2.4.7.8 daulibstm32f4xxcan-CanTransmit-LLR-008 256](#_Toc158906682)

[11.2.4.7.9 daulibstm32f4xxcan-CanTransmit-LLR-009 256](#_Toc158906683)

[11.2.4.7.10 daulibstm32f4xxcan-CanTransmit-LLR-010 257](#_Toc158906684)

[11.2.4.7.11 daulibstm32f4xxcan-CanTransmit-LLR-011 257](#_Toc158906685)

[11.2.5 CanReceive 257](#_Toc158906686)

[11.2.5.1 Brief Description 257](#_Toc158906687)

[11.2.5.2 List of HLRs allocated 257](#_Toc158906688)

[11.2.5.3 List of global variables accessed and modified 257](#_Toc158906689)

[11.2.5.4 Parameter list (Input/Output) 258](#_Toc158906690)

[11.2.5.5 Return Value 258](#_Toc158906691)

[11.2.5.6 Other CSUs called by this CSU 258](#_Toc158906692)

[11.2.5.7 Description of list of LLRs allocated 258](#_Toc158906693)

[11.2.5.7.1 daulibstm32f4xxcan-CanReceive-LLR-001 258](#_Toc158906694)

[11.2.5.7.2 daulibstm32f4xxcan-CanReceive-LLR-002 258](#_Toc158906695)

[11.2.5.7.3 daulibstm32f4xxcan-CanReceive-LLR-003 258](#_Toc158906696)

[11.2.5.7.4 daulibstm32f4xxcan-CanReceive-LLR-004 259](#_Toc158906697)

[11.2.5.7.5 daulibstm32f4xxcan-CanReceive-LLR-005 259](#_Toc158906698)

[11.2.5.7.6 daulibstm32f4xxcan-CanReceive-LLR-006 259](#_Toc158906699)

[11.2.5.7.7 daulibstm32f4xxcan-CanReceive-LLR-007 259](#_Toc158906700)

[11.2.5.7.8 daulibstm32f4xxcan-CanReceive-LLR-008 260](#_Toc158906701)

[11.2.5.7.9 daulibstm32f4xxcan-CanReceive-LLR-009 260](#_Toc158906702)

[11.2.6 CanItConfig 260](#_Toc158906703)

[11.2.6.1 Brief Description 260](#_Toc158906704)

[11.2.6.2 List of HLRs allocated 260](#_Toc158906705)

[11.2.6.3 List of global variables accessed and modified 260](#_Toc158906706)

[11.2.6.4 Parameter list (Input/Output) 260](#_Toc158906707)

[11.2.6.5 Return Value 261](#_Toc158906708)

[11.2.6.6 Other CSUs called by this CSU 261](#_Toc158906709)

[11.2.6.7 Description of list of LLRs allocated 261](#_Toc158906710)

[11.2.6.7.1 daulibstm32f4xxcan-CanItConfig-LLR-001 261](#_Toc158906711)

[11.2.6.7.2 daulibstm32f4xxcan-CanItConfig-LLR-002 261](#_Toc158906712)

[11.3 daulibstm32f4xxcrc 261](#_Toc158906713)

[11.3.1 CrcResetDr 261](#_Toc158906714)

[11.3.1.1 Brief Description 262](#_Toc158906715)

[11.3.1.2 List of HLRs allocated 262](#_Toc158906716)

[11.3.1.3 List of global variables accessed and modified 262](#_Toc158906717)

[11.3.1.4 Parameter list (Input/Output) 262](#_Toc158906718)

[11.3.1.5 Return Value 262](#_Toc158906719)

[11.3.1.6 Other CSUs called by this CSU 262](#_Toc158906720)

[11.3.1.7 Description of list of LLRs allocated 262](#_Toc158906721)

[11.3.1.7.1 daulibstm32f4xxcrc-CrcResetDr-LLR-001 262](#_Toc158906722)

[11.3.2 CrcCalcBlockCrc 263](#_Toc158906723)

[11.3.2.1 Brief Description 263](#_Toc158906724)

[11.3.2.2 List of HLRs allocated 263](#_Toc158906725)

[11.3.2.3 List of global variables accessed and modified 263](#_Toc158906726)

[11.3.2.4 Parameter list (Input/Output) 263](#_Toc158906727)

[11.3.2.5 Return Value 263](#_Toc158906728)

[11.3.2.6 Other CSUs called by this CSU 263](#_Toc158906729)

[11.3.2.7 Description of list of LLRs allocated 263](#_Toc158906730)

[11.3.2.7.1 daulibstm32f4xxcrc-CrcCalcBlockCrc-LLR-001 264](#_Toc158906731)

[11.4 daulibstm32f4xxflash 264](#_Toc158906732)

[11.4.1 FlashSetLatency 264](#_Toc158906733)

[11.4.1.1 Brief Description 264](#_Toc158906734)

[11.4.1.2 List of HLRs allocated 264](#_Toc158906735)

[11.4.1.3 List of global variables accessed and modified 264](#_Toc158906736)

[11.4.1.4 Parameter list (Input/Output) 264](#_Toc158906737)

[11.4.1.5 Return Value 265](#_Toc158906738)

[11.4.1.6 Other CSUs called by this CSU 265](#_Toc158906739)

[11.4.1.7 Description of list of LLRs allocated 265](#_Toc158906740)

[11.4.1.7.1 daulibstm32f4xxflash-FlashSetLatency-LLR-001 265](#_Toc158906741)

[11.4.2 FlashPrefetchBufferCmd 265](#_Toc158906742)

[11.4.2.1 Brief Description 265](#_Toc158906743)

[11.4.2.2 List of HLRs allocated 265](#_Toc158906744)

[11.4.2.3 List of global variables accessed and modified 265](#_Toc158906745)

[11.4.2.4 Parameter list (Input/Output) 266](#_Toc158906746)

[11.4.2.5 Return Value 266](#_Toc158906747)

[11.4.2.6 Other CSUs called by this CSU 266](#_Toc158906748)

[11.4.2.7 Description of list of LLRs allocated 266](#_Toc158906749)

[11.4.2.7.1 daulibstm32f4xxflash-FlashPrefetchBufferCmd-LLR-001 266](#_Toc158906750)

[11.4.2.7.2 daulibstm32f4xxflash-FlashPrefetchBufferCmd-LLR-002 266](#_Toc158906751)

[11.4.3 FlashInstructionCacheCmd 266](#_Toc158906752)

[11.4.3.1 Brief Description 266](#_Toc158906753)

[11.4.3.2 List of HLRs allocated 267](#_Toc158906754)

[11.4.3.3 List of global variables accessed and modified 267](#_Toc158906755)

[11.4.3.4 Parameter list (Input/Output) 267](#_Toc158906756)

[11.4.3.5 Return Value 267](#_Toc158906757)

[11.4.3.6 Other CSUs called by this CSU 267](#_Toc158906758)

[11.4.3.7 Description of list of LLRs allocated 267](#_Toc158906759)

[11.4.3.7.1 daulibstm32f4xxflash-FlashInstructionCacheCmd-LLR-001 267](#_Toc158906760)

[11.4.3.7.2 daulibstm32f4xxflash-FlashInstructionCacheCmd-LLR-002 267](#_Toc158906761)

[11.4.4 FlashDataCacheCmd 268](#_Toc158906762)

[11.4.4.1 Brief Description 268](#_Toc158906763)

[11.4.4.2 List of HLRs allocated 268](#_Toc158906764)

[11.4.4.3 List of global variables accessed and modified 268](#_Toc158906765)

[11.4.4.4 Parameter list (Input/Output) 268](#_Toc158906766)

[11.4.4.5 Return Value 268](#_Toc158906767)

[11.4.4.6 Other CSUs called by this CSU 268](#_Toc158906768)

[11.4.4.7 Description of list of LLRs allocated 269](#_Toc158906769)

[11.4.4.7.1 daulibstm32f4xxflash-FlashDataCacheCmd-LLR-001 269](#_Toc158906770)

[11.4.4.7.2 daulibstm32f4xxflash-FlashDataCacheCmd-LLR-002 269](#_Toc158906771)

[11.5 daulibstm32f4xxfsmc 269](#_Toc158906772)

[11.5.1 FsmcNorSramInit 269](#_Toc158906773)

[11.5.1.1 Brief Description 269](#_Toc158906774)

[11.5.1.2 List of HLRs allocated 269](#_Toc158906775)

[11.5.1.3 List of global variables accessed and modified 269](#_Toc158906776)

[11.5.1.4 Parameter list (Input/Output) 270](#_Toc158906777)

[11.5.1.5 Return Value 270](#_Toc158906778)

[11.5.1.6 Other CSUs called by this CSU 270](#_Toc158906779)

[11.5.1.7 Description of list of LLRs allocated 270](#_Toc158906780)

[11.5.1.7.1 daulibstm32f4xxfsmc-FsmcNorSramInit-LLR-001 270](#_Toc158906781)

[11.5.1.7.2 daulibstm32f4xxfsmc-FsmcNorSramInit-LLR-002 271](#_Toc158906782)

[11.5.1.7.3 daulibstm32f4xxfsmc-FsmcNorSramInit-LLR-003 271](#_Toc158906783)

[11.5.2 FsmcNorSramCmd 271](#_Toc158906784)

[11.5.2.1 Brief Description 271](#_Toc158906785)

[11.5.2.2 List of HLRs allocated 271](#_Toc158906786)

[11.5.2.3 List of global variables accessed and modified 272](#_Toc158906787)

[11.5.2.4 Parameter list (Input/Output) 272](#_Toc158906788)

[11.5.2.5 Return Value 272](#_Toc158906789)

[11.5.2.6 Other CSUs called by this CSU 272](#_Toc158906790)

[11.5.2.7 Description of list of LLRs allocated 272](#_Toc158906791)

[11.5.2.7.1 daulibstm32f4xxfsmc-FsmcNorSramCmd-LLR-001 272](#_Toc158906792)

[11.5.2.7.2 daulibstm32f4xxfsmc-FsmcNorSramCmd-LLR-002 272](#_Toc158906793)

[11.6 daulibstm32f4xxgpio 273](#_Toc158906794)

[11.6.1 GpioInit 273](#_Toc158906795)

[11.6.1.1 Brief Description 273](#_Toc158906796)

[11.6.1.2 List of HLRs allocated 273](#_Toc158906797)

[11.6.1.3 List of global variables accessed and modified 273](#_Toc158906798)

[11.6.1.4 Parameter list (Input/Output) 273](#_Toc158906799)

[11.6.1.5 Return Value 273](#_Toc158906800)

[11.6.1.6 Other CSUs called by this CSU 274](#_Toc158906801)

[11.6.1.7 Description of list of LLRs allocated 274](#_Toc158906802)

[11.6.1.7.1 daulibstm32f4xxgpio-GpioInit-LLR-001 274](#_Toc158906803)

[11.6.2 GpioSetBits 275](#_Toc158906804)

[11.6.2.1 Brief Description 275](#_Toc158906805)

[11.6.2.2 List of HLRs allocated 275](#_Toc158906806)

[11.6.2.3 List of global variables accessed and modified 275](#_Toc158906807)

[11.6.2.4 Parameter list (Input/Output) 275](#_Toc158906808)

[11.6.2.5 Return Value 275](#_Toc158906809)

[11.6.2.6 Other CSUs called by this CSU 275](#_Toc158906810)

[11.6.2.7 Description of list of LLRs allocated 275](#_Toc158906811)

[11.6.2.7.1 daulibstm32f4xxgpio-GpioSetBits-LLR-001 276](#_Toc158906812)

[11.6.3 GpioResetBits 276](#_Toc158906813)

[11.6.3.1 Brief Description 276](#_Toc158906814)

[11.6.3.2 List of HLRs allocated 276](#_Toc158906815)

[11.6.3.3 List of global variables accessed and modified 276](#_Toc158906816)

[11.6.3.4 Parameter list (Input/Output) 276](#_Toc158906817)

[11.6.3.5 Return Value 276](#_Toc158906818)

[11.6.3.6 Other CSUs called by this CSU 276](#_Toc158906819)

[11.6.3.7 Description of list of LLRs allocated 277](#_Toc158906820)

[11.6.3.7.1 daulibstm32f4xxgpio-GpioResetBits-LLR-001 277](#_Toc158906821)

[11.6.4 GpioReadInputDataBit 277](#_Toc158906822)

[11.6.4.1 Brief Description 277](#_Toc158906823)

[11.6.4.2 List of HLRs allocated 277](#_Toc158906824)

[11.6.4.3 List of global variables accessed and modified 277](#_Toc158906825)

[11.6.4.4 Parameter list (Input/Output) 277](#_Toc158906826)

[11.6.4.5 Return Value 277](#_Toc158906827)

[11.6.4.6 Other CSUs called by this CSU 278](#_Toc158906828)

[11.6.4.7 Description of list of LLRs allocated 278](#_Toc158906829)

[11.6.4.7.1 daulibstm32f4xxgpio-GpioReadInputDataBit-LLR-001 278](#_Toc158906830)

[11.6.4.7.2 daulibstm32f4xxgpio-GpioReadInputDataBit-LLR-002 278](#_Toc158906831)

[11.6.5 GpioToggleBits 278](#_Toc158906832)

[11.6.5.1 Brief Description 278](#_Toc158906833)

[11.6.5.2 List of HLRs allocated 278](#_Toc158906834)

[11.6.5.3 List of global variables accessed and modified 278](#_Toc158906835)

[11.6.5.4 Parameter list (Input/Output) 279](#_Toc158906836)

[11.6.5.5 Return Value 279](#_Toc158906837)

[11.6.5.6 Other CSUs called by this CSU 279](#_Toc158906838)

[11.6.5.7 Description of list of LLRs allocated 279](#_Toc158906839)

[11.6.5.7.1 daulibstm32f4xxgpio-GpioToggleBits-LLR-001 279](#_Toc158906840)

[11.6.6 GpioPinAFConfig 279](#_Toc158906841)

[11.6.6.1 Brief Description 279](#_Toc158906842)

[11.6.6.2 List of HLRs allocated 279](#_Toc158906843)

[11.6.6.3 List of global variables accessed and modified 280](#_Toc158906844)

[11.6.6.4 Parameter list (Input/Output) 280](#_Toc158906845)

[11.6.6.5 Return Value 280](#_Toc158906846)

[11.6.6.6 Other CSUs called by this CSU 280](#_Toc158906847)

[11.6.6.7 Description of list of LLRs allocated 280](#_Toc158906848)

[11.6.6.7.1 daulibstm32f4xxgpio-GpioPinAFConfig-LLR-001 280](#_Toc158906849)

[11.7 daulibstm32f4xxi2c 281](#_Toc158906850)

[11.7.1 I2cInit 281](#_Toc158906851)

[11.7.1.1 Brief Description 281](#_Toc158906852)

[11.7.1.2 List of HLRs allocated 281](#_Toc158906853)

[11.7.1.3 List of global variables accessed and modified 281](#_Toc158906854)

[11.7.1.4 Parameter list (Input/Output) 281](#_Toc158906855)

[11.7.1.5 Return Value 281](#_Toc158906856)

[11.7.1.6 Other CSUs called by this CSU 282](#_Toc158906857)

[11.7.1.7 Description of list of LLRs allocated 282](#_Toc158906858)

[11.7.1.7.1 daulibstm32f4xxi2c-I2cInit-LLR-001 282](#_Toc158906859)

[11.7.1.7.2 daulibstm32f4xxi2c-I2cInit-LLR-002 282](#_Toc158906860)

[11.7.1.7.3 daulibstm32f4xxi2c-I2cInit-LLR-003 282](#_Toc158906861)

[11.7.1.7.4 daulibstm32f4xxi2c-I2cInit-LLR-004 283](#_Toc158906862)

[11.7.1.7.5 daulibstm32f4xxi2c-I2cInit-LLR-005 283](#_Toc158906863)

[11.7.1.7.6 daulibstm32f4xxi2c-I2cInit-LLR-006 283](#_Toc158906864)

[11.7.1.7.7 daulibstm32f4xxi2c-I2cInit-LLR-007 283](#_Toc158906865)

[11.7.1.7.8 daulibstm32f4xxi2c-I2cInit-LLR-008 284](#_Toc158906866)

[11.7.1.7.9 daulibstm32f4xxi2c-I2cInit-LLR-009 284](#_Toc158906867)

[11.7.1.7.10 daulibstm32f4xxi2c-I2cInit-LLR-010 284](#_Toc158906868)

[11.7.2 I2cCmd 284](#_Toc158906869)

[11.7.2.1 Brief Description 284](#_Toc158906870)

[11.7.2.2 List of HLRs allocated 284](#_Toc158906871)

[11.7.2.3 List of global variables accessed and modified 285](#_Toc158906872)

[11.7.2.4 Parameter list (Input/Output) 285](#_Toc158906873)

[11.7.2.5 Return Value 285](#_Toc158906874)

[11.7.2.6 Other CSUs called by this CSU 285](#_Toc158906875)

[11.7.2.7 Description of list of LLRs allocated 285](#_Toc158906876)

[11.7.2.7.1 daulibstm32f4xxi2c-I2cCmd-LLR-001 285](#_Toc158906877)

[11.7.2.7.2 daulibstm32f4xxi2c-I2cCmd-LLR-002 285](#_Toc158906878)

[11.7.3 I2cGenerateStart 286](#_Toc158906879)

[11.7.3.1 Brief Description 286](#_Toc158906880)

[11.7.3.2 List of HLRs allocated 286](#_Toc158906881)

[11.7.3.3 List of global variables accessed and modified 286](#_Toc158906882)

[11.7.3.4 Parameter list (Input/Output) 286](#_Toc158906883)

[11.7.3.5 Return Value 286](#_Toc158906884)

[11.7.3.6 Other CSUs called by this CSU 287](#_Toc158906885)

[11.7.3.7 Description of list of LLRs allocated 287](#_Toc158906886)

[11.7.3.7.1 daulibstm32f4xxi2c-I2cGenerateStart-LLR-001 287](#_Toc158906887)

[11.7.3.7.2 daulibstm32f4xxi2c-I2cGenerateStart-LLR-002 287](#_Toc158906888)

[11.7.4 I2cGenerateStop 287](#_Toc158906889)

[11.7.4.1 Brief Description 287](#_Toc158906890)

[11.7.4.2 List of HLRs allocated 287](#_Toc158906891)

[11.7.4.3 List of global variables accessed and modified 288](#_Toc158906892)

[11.7.4.4 Parameter list (Input/Output) 288](#_Toc158906893)

[11.7.4.5 Return Value 288](#_Toc158906894)

[11.7.4.6 Other CSUs called by this CSU 288](#_Toc158906895)

[11.7.4.7 Description of list of LLRs allocated 288](#_Toc158906896)

[11.7.4.7.1 daulibstm32f4xxi2c-I2cGenerateStop-LLR-001 288](#_Toc158906897)

[11.7.4.7.2 daulibstm32f4xxi2c-I2cGenerateStop-LLR-002 289](#_Toc158906898)

[11.7.5 I2cSend7bitAddress 289](#_Toc158906899)

[11.7.5.1 Brief Description 289](#_Toc158906900)

[11.7.5.2 List of HLRs allocated 289](#_Toc158906901)

[11.7.5.3 List of global variables accessed and modified 289](#_Toc158906902)

[11.7.5.4 Parameter list (Input/Output) 289](#_Toc158906903)

[11.7.5.5 Return Value 290](#_Toc158906904)

[11.7.5.6 Other CSUs called by this CSU 290](#_Toc158906905)

[11.7.5.7 Description of list of LLRs allocated 290](#_Toc158906906)

[11.7.5.7.1 daulibstm32f4xxi2c-I2cSend7bitAddress-LLR-001 290](#_Toc158906907)

[11.7.5.7.2 daulibstm32f4xxi2c-I2cSend7bitAddress-LLR-002 290](#_Toc158906908)

[11.7.5.7.3 daulibstm32f4xxi2c-I2cSend7bitAddress-LLR-003 290](#_Toc158906909)

[11.7.6 I2cAcknowledgeConfig 290](#_Toc158906910)

[11.7.6.1 Brief Description 291](#_Toc158906911)

[11.7.6.2 List of HLRs allocated 291](#_Toc158906912)

[11.7.6.3 List of global variables accessed and modified 291](#_Toc158906913)

[11.7.6.4 Parameter list (Input/Output) 291](#_Toc158906914)

[11.7.6.5 Return Value 291](#_Toc158906915)

[11.7.6.6 Other CSUs called by this CSU 291](#_Toc158906916)

[11.7.6.7 Description of list of LLRs allocated 291](#_Toc158906917)

[11.7.6.7.1 daulibstm32f4xxi2c-I2cAcknowledgeConfig-LLR-001 292](#_Toc158906918)

[11.7.6.7.2 daulibstm32f4xxi2c-I2cAcknowledgeConfig-LLR-002 292](#_Toc158906919)

[11.7.7 I2cSendData 292](#_Toc158906920)

[11.7.7.1 Brief Description 292](#_Toc158906921)

[11.7.7.2 List of HLRs allocated 292](#_Toc158906922)

[11.7.7.3 List of global variables accessed and modified 292](#_Toc158906923)

[11.7.7.4 Parameter list (Input/Output) 292](#_Toc158906924)

[11.7.7.5 Return Value 293](#_Toc158906925)

[11.7.7.6 Other CSUs called by this CSU 293](#_Toc158906926)

[11.7.7.7 Description of list of LLRs allocated 293](#_Toc158906927)

[11.7.7.7.1 daulibstm32f4xxi2c-I2cSendData-LLR-001 293](#_Toc158906928)

[11.7.8 I2cReceiveData 293](#_Toc158906929)

[11.7.8.1 Brief Description 293](#_Toc158906930)

[11.7.8.2 List of HLRs allocated 294](#_Toc158906931)

[11.7.8.3 List of global variables accessed and modified 294](#_Toc158906932)

[11.7.8.4 Parameter list (Input/Output) 294](#_Toc158906933)

[11.7.8.5 Return Value 294](#_Toc158906934)

[11.7.8.6 Other CSUs called by this CSU 294](#_Toc158906935)

[11.7.8.7 Description of list of LLRs allocated 294](#_Toc158906936)

[11.7.8.7.1 daulibstm32f4xxi2c-I2cReceiveData-LLR-001 294](#_Toc158906937)

[11.7.9 I2cITConfig 295](#_Toc158906938)

[11.7.9.1 Brief Description 295](#_Toc158906939)

[11.7.9.2 List of HLRs allocated 295](#_Toc158906940)

[11.7.9.3 List of global variables accessed and modified 295](#_Toc158906941)

[11.7.9.4 Parameter list (Input/Output) 295](#_Toc158906942)

[11.7.9.5 Return Value 295](#_Toc158906943)

[11.7.9.6 Other CSUs called by this CSU 296](#_Toc158906944)

[11.7.9.7 Description of list of LLRs allocated 296](#_Toc158906945)

[11.7.9.7.1 daulibstm32f4xxi2c-I2cITConfig-LLR-001 296](#_Toc158906946)

[11.7.9.7.2 daulibstm32f4xxi2c-I2cITConfig-LLR-002 296](#_Toc158906947)

[11.7.10 I2cGetLastEvent 296](#_Toc158906948)

[11.7.10.1 Brief Description 296](#_Toc158906949)

[11.7.10.2 List of HLRs allocated 296](#_Toc158906950)

[11.7.10.3 List of global variables accessed and modified 296](#_Toc158906951)

[11.7.10.4 Parameter list (Input/Output) 297](#_Toc158906952)

[11.7.10.5 Return Value 297](#_Toc158906953)

[11.7.10.6 Other CSUs called by this CSU 297](#_Toc158906954)

[11.7.10.7 Description of list of LLRs allocated 297](#_Toc158906955)

[11.7.10.7.1 daulibstm32f4xxi2c-I2cGetLastEvent-LLR-001 297](#_Toc158906956)

[11.7.10.7.2 daulibstm32f4xxi2c-I2cGetLastEvent-LLR-002 297](#_Toc158906957)

[11.7.10.7.3 daulibstm32f4xxi2c-I2cGetLastEvent-LLR-003 298](#_Toc158906958)

[11.7.11 I2cGetITStatus 298](#_Toc158906959)

[11.7.11.1 Brief Description 298](#_Toc158906960)

[11.7.11.2 List of HLRs allocated 298](#_Toc158906961)

[11.7.11.3 List of global variables accessed and modified 298](#_Toc158906962)

[11.7.11.4 Parameter list (Input/Output) 298](#_Toc158906963)

[11.7.11.5 Return Value 298](#_Toc158906964)

[11.7.11.6 Other CSUs called by this CSU 299](#_Toc158906965)

[11.7.11.7 Description of list of LLRs allocated 299](#_Toc158906966)

[11.7.11.7.1 daulibstm32f4xxi2c-I2cGetITStatus-LLR-001 299](#_Toc158906967)

[11.7.11.7.2 daulibstm32f4xxi2c-I2cGetITStatus-LLR-002 299](#_Toc158906968)

[11.7.11.7.3 daulibstm32f4xxi2c-I2cGetITStatus-LLR-003 299](#_Toc158906969)

[11.7.11.7.4 daulibstm32f4xxi2c-I2cGetITStatus-LLR-004 299](#_Toc158906970)

[11.7.11.7.5 daulibstm32f4xxi2c-I2cGetITStatus-LLR-005 299](#_Toc158906971)

[11.7.12 I2cClearITPendingBit 300](#_Toc158906972)

[11.7.12.1 Brief Description 300](#_Toc158906973)

[11.7.12.2 List of HLRs allocated 300](#_Toc158906974)

[11.7.12.3 List of global variables accessed and modified 300](#_Toc158906975)

[11.7.12.4 Parameter list (Input/Output) 300](#_Toc158906976)

[11.7.12.5 Return Value 300](#_Toc158906977)

[11.7.12.6 Other CSUs called by this CSU 301](#_Toc158906978)

[11.7.12.7 Description of list of LLRs allocated 301](#_Toc158906979)

[11.7.12.7.1 daulibstm32f4xxi2c-I2cClearITPendingBit-LLR-001 301](#_Toc158906980)

[11.7.12.7.2 daulibstm32f4xxi2c-I2cClearITPendingBit-LLR-002 301](#_Toc158906981)

[11.8 daulibstm32f4xxiwdg 301](#_Toc158906982)

[11.8.1 IwdgWriteAccessCmd 301](#_Toc158906983)

[11.8.1.1 Brief Description 301](#_Toc158906984)

[11.8.1.2 List of HLRs allocated 302](#_Toc158906985)

[11.8.1.3 List of global variables accessed and modified 302](#_Toc158906986)

[11.8.1.4 Parameter list (Input/Output) 302](#_Toc158906987)

[11.8.1.5 Return Value 302](#_Toc158906988)

[11.8.1.6 Other CSUs called by this CSU 302](#_Toc158906989)

[11.8.1.7 Description of list of LLRs allocated 302](#_Toc158906990)

[11.8.1.7.1 daulibstm32f4xxiwdg-IwdgWriteAccessCmd-LLR-001 302](#_Toc158906991)

[11.8.2 IwdgSetPrescaler 302](#_Toc158906992)

[11.8.2.1 Brief Description 303](#_Toc158906993)

[11.8.2.2 List of HLRs allocated 303](#_Toc158906994)

[11.8.2.3 List of global variables accessed and modified 303](#_Toc158906995)

[11.8.2.4 Parameter list (Input/Output) 303](#_Toc158906996)

[11.8.2.5 Return Value 303](#_Toc158906997)

[11.8.2.6 Other CSUs called by this CSU 303](#_Toc158906998)

[11.8.2.7 Description of list of LLRs allocated 303](#_Toc158906999)

[11.8.2.7.1 daulibstm32f4xxiwdg-IwdgSetPrescaler-LLR-001 303](#_Toc158907000)

[11.8.3 IwdgSetReload 304](#_Toc158907001)

[11.8.3.1 Brief Description 304](#_Toc158907002)

[11.8.3.2 List of HLRs allocated 304](#_Toc158907003)

[11.8.3.3 List of global variables accessed and modified 304](#_Toc158907004)

[11.8.3.4 Parameter list (Input/Output) 304](#_Toc158907005)

[11.8.3.5 Return Value 304](#_Toc158907006)

[11.8.3.6 Other CSUs called by this CSU 304](#_Toc158907007)

[11.8.3.7 Description of list of LLRs allocated 304](#_Toc158907008)

[11.8.3.7.1 daulibstm32f4xxiwdg-IwdgSetReload-LLR-001 305](#_Toc158907009)

[11.8.4 IwdgReloadCounter 305](#_Toc158907010)

[11.8.4.1 Brief Description 305](#_Toc158907011)

[11.8.4.2 List of HLRs allocated 305](#_Toc158907012)

[11.8.4.3 List of global variables accessed and modified 305](#_Toc158907013)

[11.8.4.4 Parameter list (Input/Output) 305](#_Toc158907014)

[11.8.4.5 Return Value 305](#_Toc158907015)

[11.8.4.6 Other CSUs called by this CSU 305](#_Toc158907016)

[11.8.4.7 Description of list of LLRs allocated 306](#_Toc158907017)

[11.8.4.7.1 daulibstm32f4xxiwdg-IwdgReloadCounter-LLR-001 306](#_Toc158907018)

[11.8.5 IwdgEnable 306](#_Toc158907019)

[11.8.5.1 Brief Description 306](#_Toc158907020)

[11.8.5.2 List of HLRs allocated 306](#_Toc158907021)

[11.8.5.3 List of global variables accessed and modified 306](#_Toc158907022)

[11.8.5.4 Parameter list (Input/Output) 306](#_Toc158907023)

[11.8.5.5 Return Value 306](#_Toc158907024)

[11.8.5.6 Other CSUs called by this CSU 307](#_Toc158907025)

[11.8.5.7 Description of list of LLRs allocated 307](#_Toc158907026)

[11.8.5.7.1 daulibstm32f4xxiwdg-IwdgEnable-LLR-001 307](#_Toc158907027)

[11.9 daulibstm32f4xxpwr 307](#_Toc158907028)

[11.9.1 PwrMainRegulatorModeConfig 307](#_Toc158907029)

[11.9.1.1 Brief Description 307](#_Toc158907030)

[11.9.1.2 List of HLRs allocated 307](#_Toc158907031)

[11.9.1.3 List of global variables accessed and modified 307](#_Toc158907032)

[11.9.1.4 Parameter list (Input/Output) 308](#_Toc158907033)

[11.9.1.5 Return Value 308](#_Toc158907034)

[11.9.1.6 Other CSUs called by this CSU 308](#_Toc158907035)

[11.9.1.7 Description of list of LLRs allocated 308](#_Toc158907036)

[11.9.1.7.1 daulibstm32f4xxpwr-PwrMainRegulatorModeConfig-LLR-001 308](#_Toc158907037)

[11.9.1.7.2 daulibstm32f4xxpwr-PwrMainRegulatorModeConfig-LLR-002 308](#_Toc158907038)

[11.10 daulibstm32f4xxrcc 308](#_Toc158907039)

[11.10.1 RccDeInit 309](#_Toc158907040)

[11.10.1.1 Brief Description 309](#_Toc158907041)

[11.10.1.2 List of HLRs allocated 309](#_Toc158907042)

[11.10.1.3 List of global variables accessed and modified 309](#_Toc158907043)

[11.10.1.4 Parameter list (Input/Output) 309](#_Toc158907044)

[11.10.1.5 Return Value 309](#_Toc158907045)

[11.10.1.6 Other CSUs called by this CSU 309](#_Toc158907046)

[11.10.1.7 Description of list of LLRs allocated 310](#_Toc158907047)

[11.10.1.7.1 daulibstm32f4xxrcc-RccDeInit-LLR-001 310](#_Toc158907048)

[11.10.1.7.2 daulibstm32f4xxrcc-RccDeInit-LLR-002 310](#_Toc158907049)

[11.10.1.7.3 daulibstm32f4xxrcc-RccDeInit-LLR-003 310](#_Toc158907050)

[11.10.1.7.4 daulibstm32f4xxrcc-RccDeInit-LLR-004 310](#_Toc158907051)

[11.10.1.7.5 daulibstm32f4xxrcc-RccDeInit-LLR-005 310](#_Toc158907052)

[11.10.1.7.6 daulibstm32f4xxrcc-RccDeInit-LLR-006 310](#_Toc158907053)

[11.10.2 RccHseConfig 311](#_Toc158907054)

[11.10.2.1 Brief Description 311](#_Toc158907055)

[11.10.2.2 List of HLRs allocated 311](#_Toc158907056)

[11.10.2.3 List of global variables accessed and modified 311](#_Toc158907057)

[11.10.2.4 Parameter list (Input/Output) 311](#_Toc158907058)

[11.10.2.5 Return Value 311](#_Toc158907059)

[11.10.2.6 Other CSUs called by this CSU 311](#_Toc158907060)

[11.10.2.7 Description of list of LLRs allocated 311](#_Toc158907061)

[11.10.2.7.1 daulibstm32f4xxrcc-RccHseConfig-LLR-001 311](#_Toc158907062)

[11.10.3 RccWaitForHseStartUp 312](#_Toc158907063)

[11.10.3.1 Brief Description 312](#_Toc158907064)

[11.10.3.2 List of HLRs allocated 312](#_Toc158907065)

[11.10.3.3 List of global variables accessed and modified 312](#_Toc158907066)

[11.10.3.4 Parameter list (Input/Output) 312](#_Toc158907067)

[11.10.3.5 Return Value 312](#_Toc158907068)

[11.10.3.6 Other CSUs called by this CSU 312](#_Toc158907069)

[11.10.3.7 Description of list of LLRs allocated 313](#_Toc158907070)

[11.10.3.7.1 daulibstm32f4xxrcc-RccWaitForHseStartUp-LLR-001 313](#_Toc158907071)

[11.10.3.7.2 daulibstm32f4xxrcc-RccWaitForHseStartUp-LLR-002 313](#_Toc158907072)

[11.10.3.7.3 daulibstm32f4xxrcc-RccWaitForHseStartUp-LLR-003 313](#_Toc158907073)

[11.10.4 RccPllConfig 313](#_Toc158907074)

[11.10.4.1 Brief Description 313](#_Toc158907075)

[11.10.4.2 List of HLRs allocated 313](#_Toc158907076)

[11.10.4.3 List of global variables accessed and modified 314](#_Toc158907077)

[11.10.4.4 Parameter list (Input/Output) 314](#_Toc158907078)

[11.10.4.5 Return Value 314](#_Toc158907079)

[11.10.4.6 Other CSUs called by this CSU 314](#_Toc158907080)

[11.10.4.7 Description of list of LLRs allocated 314](#_Toc158907081)

[11.10.4.7.1 daulibstm32f4xxrcc-RccPllConfig-LLR-001 314](#_Toc158907082)

[11.10.5 RccPllCmd 314](#_Toc158907083)

[11.10.5.1 Brief Description 315](#_Toc158907084)

[11.10.5.2 List of HLRs allocated 315](#_Toc158907085)

[11.10.5.3 List of global variables accessed and modified 315](#_Toc158907086)

[11.10.5.4 Parameter list (Input/Output) 315](#_Toc158907087)

[11.10.5.5 Return Value 315](#_Toc158907088)

[11.10.5.6 Other CSUs called by this CSU 315](#_Toc158907089)

[11.10.5.7 Description of list of LLRs allocated 315](#_Toc158907090)

[11.10.5.7.1 daulibstm32f4xxrcc-RccPllCmd-LLR-001 315](#_Toc158907091)

[11.10.6 RccSysClkConfig 316](#_Toc158907092)

[11.10.6.1 Brief Description 316](#_Toc158907093)

[11.10.6.2 List of HLRs allocated 316](#_Toc158907094)

[11.10.6.3 List of global variables accessed and modified 316](#_Toc158907095)

[11.10.6.4 Parameter list (Input/Output) 316](#_Toc158907096)

[11.10.6.5 Return Value 316](#_Toc158907097)

[11.10.6.6 Other CSUs called by this CSU 316](#_Toc158907098)

[11.10.6.7 Description of list of LLRs allocated 316](#_Toc158907099)

[11.10.6.7.1 daulibstm32f4xxrcc-RccSysClkConfig-LLR-001 316](#_Toc158907100)

[11.10.7 RccGetSysClkSource 317](#_Toc158907101)

[11.10.7.1 Brief Description 317](#_Toc158907102)

[11.10.7.2 List of HLRs allocated 317](#_Toc158907103)

[11.10.7.3 List of global variables accessed and modified 317](#_Toc158907104)

[11.10.7.4 Parameter list (Input/Output) 317](#_Toc158907105)

[11.10.7.5 Return Value 317](#_Toc158907106)

[11.10.7.6 Other CSUs called by this CSU 317](#_Toc158907107)

[11.10.7.7 Description of list of LLRs allocated 318](#_Toc158907108)

[11.10.7.7.1 daulibstm32f4xxrcc-RccGetSysClkSource-LLR-001 318](#_Toc158907109)

[11.10.8 RccHclkConfig 318](#_Toc158907110)

[11.10.8.1 Brief Description 318](#_Toc158907111)

[11.10.8.2 List of HLRs allocated 318](#_Toc158907112)

[11.10.8.3 List of global variables accessed and modified 318](#_Toc158907113)

[11.10.8.4 Parameter list (Input/Output) 318](#_Toc158907114)

[11.10.8.5 Return Value 318](#_Toc158907115)

[11.10.8.6 Other CSUs called by this CSU 319](#_Toc158907116)

[11.10.8.7 Description of list of LLRs allocated 319](#_Toc158907117)

[11.10.8.7.1 daulibstm32f4xxrcc-RccHclkConfig-LLR-001 319](#_Toc158907118)

[11.10.9 RccPclk1Config 319](#_Toc158907119)

[11.10.9.1 Brief Description 319](#_Toc158907120)

[11.10.9.2 List of HLRs allocated 319](#_Toc158907121)

[11.10.9.3 List of global variables accessed and modified 319](#_Toc158907122)

[11.10.9.4 Parameter list (Input/Output) 319](#_Toc158907123)

[11.10.9.5 Return Value 320](#_Toc158907124)

[11.10.9.6 Other CSUs called by this CSU 320](#_Toc158907125)

[11.10.9.7 Description of list of LLRs allocated 320](#_Toc158907126)

[11.10.9.7.1 daulibstm32f4xxrcc-RccPclk1Config-LLR-001 320](#_Toc158907127)

[11.10.10 RccPclk2Config 320](#_Toc158907128)

[11.10.10.1 Brief Description 320](#_Toc158907129)

[11.10.10.2 List of HLRs allocated 320](#_Toc158907130)

[11.10.10.3 List of global variables accessed and modified 321](#_Toc158907131)

[11.10.10.4 Parameter list (Input/Output) 321](#_Toc158907132)

[11.10.10.5 Return Value 321](#_Toc158907133)

[11.10.10.6 Other CSUs called by this CSU 321](#_Toc158907134)

[11.10.10.7 Description of list of LLRs allocated 321](#_Toc158907135)

[11.10.10.7.1 daulibstm32f4xxrcc-RccPclk2Config-LLR-001 321](#_Toc158907136)

[11.10.11 RccGetClocksFreq 321](#_Toc158907137)

[11.10.11.1 Brief Description 321](#_Toc158907138)

[11.10.11.2 List of HLRs allocated 322](#_Toc158907139)

[11.10.11.3 List of global variables accessed and modified 322](#_Toc158907140)

[11.10.11.4 Parameter list (Input/Output) 322](#_Toc158907141)

[11.10.11.5 Return Value 322](#_Toc158907142)

[11.10.11.6 Other CSUs called by this CSU 322](#_Toc158907143)

[11.10.11.7 Description of list of LLRs allocated 322](#_Toc158907144)

[11.10.11.7.1 daulibstm32f4xxrcc-RccGetClocksFreq-LLR-001 322](#_Toc158907145)

[11.10.11.7.2 daulibstm32f4xxrcc-RccGetClocksFreq-LLR-002 322](#_Toc158907146)

[11.10.11.7.3 daulibstm32f4xxrcc-RccGetClocksFreq-LLR-003 323](#_Toc158907147)

[11.10.11.7.4 daulibstm32f4xxrcc-RccGetClocksFreq-LLR-004 323](#_Toc158907148)

[11.10.11.7.5 daulibstm32f4xxrcc-RccGetClocksFreq-LLR-005 323](#_Toc158907149)

[11.10.11.7.6 daulibstm32f4xxrcc-RccGetClocksFreq-LLR-006 323](#_Toc158907150)

[11.10.11.7.7 daulibstm32f4xxrcc-RccGetClocksFreq-LLR-007 324](#_Toc158907151)

[11.10.12 RccAhb1PeriphClockCmd 324](#_Toc158907152)

[11.10.12.1 Brief Description 324](#_Toc158907153)

[11.10.12.2 List of HLRs allocated 324](#_Toc158907154)

[11.10.12.3 List of global variables accessed and modified 324](#_Toc158907155)

[11.10.12.4 Parameter list (Input/Output) 324](#_Toc158907156)

[11.10.12.5 Return Value 325](#_Toc158907157)

[11.10.12.6 Other CSUs called by this CSU 325](#_Toc158907158)

[11.10.12.7 Description of list of LLRs allocated 325](#_Toc158907159)

[11.10.12.7.1 daulibstm32f4xxrcc-RccAhb1PeriphClockCmd-LLR-001 325](#_Toc158907160)

[11.10.12.7.2 daulibstm32f4xxrcc-RccAhb1PeriphClockCmd-LLR-002 325](#_Toc158907161)

[11.10.13 RccAhb3PeriphClockCmd 325](#_Toc158907162)

[11.10.13.1 Brief Description 325](#_Toc158907163)

[11.10.13.2 List of HLRs allocated 326](#_Toc158907164)

[11.10.13.3 List of global variables accessed and modified 326](#_Toc158907165)

[11.10.13.4 Parameter list (Input/Output) 326](#_Toc158907166)

[11.10.13.5 Return Value 326](#_Toc158907167)

[11.10.13.6 Other CSUs called by this CSU 326](#_Toc158907168)

[11.10.13.7 Description of list of LLRs allocated 326](#_Toc158907169)

[11.10.13.7.1 daulibstm32f4xxrcc-RccAhb3PeriphClockCmd-LLR-001 326](#_Toc158907170)

[11.10.13.7.2 daulibstm32f4xxrcc-RccAhb3PeriphClockCmd-LLR-002 326](#_Toc158907171)

[11.10.14 RccApb1PeriphClockCmd 327](#_Toc158907172)

[11.10.14.1 Brief Description 327](#_Toc158907173)

[11.10.14.2 List of HLRs allocated 327](#_Toc158907174)

[11.10.14.3 List of global variables accessed and modified 327](#_Toc158907175)

[11.10.14.4 Parameter list (Input/Output) 327](#_Toc158907176)

[11.10.14.5 Return Value 327](#_Toc158907177)

[11.10.14.6 Other CSUs called by this CSU 327](#_Toc158907178)

[11.10.14.7 Description of list of LLRs allocated 328](#_Toc158907179)

[11.10.14.7.1 daulibstm32f4xxrcc-RccApb1PeriphClockCmd-LLR-001 328](#_Toc158907180)

[11.10.14.7.2 daulibstm32f4xxrcc-RccApb1PeriphClockCmd-LLR-002 328](#_Toc158907181)

[11.10.15 RccGetFlagStatus 328](#_Toc158907182)

[11.10.15.1 Brief Description 328](#_Toc158907183)

[11.10.15.2 List of HLRs allocated 328](#_Toc158907184)

[11.10.15.3 List of global variables accessed and modified 328](#_Toc158907185)

[11.10.15.4 Parameter list (Input/Output) 328](#_Toc158907186)

[11.10.15.5 Return Value 329](#_Toc158907187)

[11.10.15.6 Other CSUs called by this CSU 329](#_Toc158907188)

[11.10.15.7 Description of list of LLRs allocated 329](#_Toc158907189)

[11.10.15.7.1 daulibstm32f4xxrcc-RccGetFlagStatus-LLR-001 329](#_Toc158907190)

[11.10.16 RccApb1PeriphResetCmd 329](#_Toc158907191)

[11.10.16.1 Brief Description 329](#_Toc158907192)

[11.10.16.2 List of HLRs allocated 329](#_Toc158907193)

[11.10.16.3 List of global variables accessed and modified 330](#_Toc158907194)

[11.10.16.4 Parameter list (Input/Output) 330](#_Toc158907195)

[11.10.16.5 Return Value 330](#_Toc158907196)

[11.10.16.6 Other CSUs called by this CSU 330](#_Toc158907197)

[11.10.16.7 Description of list of LLRs allocated 330](#_Toc158907198)

[11.10.16.7.1 daulibstm32f4xxrcc-RccApb1PeriphResetCmd-LLR-001 330](#_Toc158907199)

[11.10.16.7.2 daulibstm32f4xxrcc-RccApb1PeriphResetCmd-LLR-002 330](#_Toc158907200)

[12 Appendix A : Data Dictionary 331](#_Toc158907201)

[13 Appendix B : Data Constants 331](#_Toc158907202)

**LIST OF TABLES**

[Table 1: Amendment Record 41](#_Toc158905407)

[Table 2: List of Acronyms 50](#_Toc158905408)

[Table 3: Reference document 51](#_Toc158905409)

[Table 4: Error parameters 156](#_Toc158905410)

# 1 Amendment Record

Table 1: Amendment Record

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version Number** | **Description of Amendment** | **Change**  **Request No.** | **Changed By** | **Release**  **Date** |
| 1.1 | Initial Release | NA | Mounika B | 26- Jan-2022 |
| 1.2 | Below sections are updated as per LLD review comments and self review  Section 10.1.13: updated the global variabels  Section 10.1.2.3: updated the global variables  Section 10.1.3.3: updated the global variables  Section 10.1.4.3: updated the global variables  Section 10.1.5.3: updated the global variables  Section 10.1.5.6: updated the other CSU’s  Section 10.6.2.6: updated the other CSU’s  Section 10.20.2.6: updated the other CSU’s  Section 10.1.5.6:updated the other CSU’s  Section 10.1.9.6:updated the other CSU’s  Below sections are updated as per the code review comments  Section 10.6.2.4:updated formal parameters  Section 10.6:updated the file description  Section 10.6.2.1:updated function description  Section 10.19.3.4:updated the formal parameter  Section 10.19.1.4:updated the formal parameters  Section 11.7.3.4:updated the parameters list  Section 10.6:updated the file name  Below Requirement ID’s are updated based on code Review comments:  H698-LLD-CMU-FNC-80,  H698-LLD-CMU-FNC-170,  H698-LLD-CMU-FNC-541,  H698-LLD-CMU-FNC-551,  H698-LLD-CMU-FNC-552,  H698-LLD-CMU-FNC-572,  H698-LLD-CMU-FNC-574 to H698-LLD-CMU-FNC-588,  H698-LLD-CMU-FNC-981,  H698-LLD-CMU-FNC-991,  H698-LLD-CMU-FNC-990,  H698-LLD-CMU-FNC-1051,  H698-LLD-CMU-FNC-1053,  H698-LLD-CMU-FNC-1173,  H698-LLD-CMU-FNC-1151,  H698-LLD-CMU-FNC-1152,  H698-LLD-CMU-FNC-1219,  H698-LLD-CMU-FNC-1220,  H698-LLD-CMU-FNC-657,  H698-LLD-CMU-FNC-658,  H698-LLD-CMU-FNC-659,  H698-LLD-CMU-FNC-20  H698-LLD-CMU-FNC-21,  H698-LLD-CMU-FNC-32,  H698-LLD-CMU-FNC-33,  H698-LLD-CMU-FNC-31,  H698-LLD-CMU-FNC-55,  H698-LLD-CMU-FNC-58,  H698-LLD-CMU-FNC-71,  H698-LLD-CMU-FNC-59,  H698-LLD-CMU-FNC-1451,  H698-LLD-CMU-FNC-1484,  H698-LLD-CMU-FNC-1935, H698-LLD-CMU-FNC-2057, H698-LLD-CMU-FNC-1934, H698-LLD-CMU-FNC-1182, H698-LLD-CMU-FNC-1979, H698-LLD-CMU-FNC-1980, H698-LLD-CMU-FNC-1184, H698-LLD-CMU-FNC-2058, H698-LLD-CMU-FNC-1205, H698-LLD-CMU-FNC-1206, H698-LLD-CMU-FNC-1182, H698-LLD-CMU-FNC-1183, H698-LLD-CMU-FNC-1184, H698-LLD-CMU-FNC-1185, H698-LLD-CMU-FNC-1196, H698-LLD-CMU-FNC-1186, H698-LLD-CMU-FNC-1187, H698-LLD-CMU-FNC-1188, H698-LLD-CMU-FNC-1189, H698-LLD-CMU-FNC-1190, H698-LLD-CMU-FNC-1191, H698-LLD-CMU-FNC-1192, H698-LLD-CMU-FNC-1193, H698-LLD-CMU-FNC-1194, H698-LLD-CMU-FNC-1195,  H698-LLD-CMU-FNC-76,  H698-LLD-CMU-FNC-70  Updated Data Dictionary and Data Constants as per the LLD review comments and code review comments | PR100028 ,PR100029 | Mounika B | 05-Feb-2022 |
| 1.3 | Below Requirement ID’s are updated based on QA Review comments:  H698-LLD-CMU-FNC-2057,  H698-LLD-CMU-FNC-80,  H698-LLD-CMU-FNC-2058,  H698-LLD-CMU-FNC-76,  H698-LLD-CMU-FNC-56,  H698-LLD-CMU-FNC-55,  H698-LLD-CMU-FNC-75,  H698-LLD-CMU-FNC-57,  H698-LLD-CMU-FNC-826,  H698-LLD-CMU-FNC-1485,  H698-LLD-CMU-FNC-361,  H698-LLD-CMU-FNC-893  Updated data constants and data dictionary | PR100028 ,PR100029 | Mounika B | 07-Feb-2022 |
| 1.4 | Below Requirement ID’s are updated based on Latest SRS  Updated section:  Section 4: Scope updated  Section 5: Reference updated  Section 10.2.5.5 – updated the return values  Section 10.3.2.3 – deleted A825\_comm\_task\_stk2 reference  Section 10.8.1.6 – deleted A825CommInit2 and A825Init2 reference  Section 10.7.1.3 – deleted Hw\_cmnum reference  Section 10.13.5.3 – deleted U32\_checksum\_app reference  Section 10.1.4.6 – updated other CSU’s  Section 11.2.5.4 – updated Parameter list (Input/Output)  Section 10.14.1 to section 10.14.1.7 – function InitializeLeds is newly added  Section 10.5.1 to section 10.5.1.7 – function PowerOnRamTest is newly added  Section 10.14 – daucmfrtc file is newly added  Section 10.15 – daucmfsi2c file is newly added  Section 11.7 – daulibstm32f4xxi2c file is newly added  Updated ID’s:  H698-LLD-CMU-FNC-32  H698-LLD-CMU-FNC-54  H698-LLD-CMU-FNC-76  H698-LLD-CMU-FNC-84  H698-LLD-CMU-FNC-85  H698-LLD-CMU-FNC-541  H698-LLD-CMU-FNC-938  H698-LLD-CMU-FNC-1185  H698-LLD-CMU-FNC-1217  H698-LLD-CMU-FNC-1232  H698-LLD-CMU-FNC-1253  H698-LLD-CMU-FNC-1298  H698-LLD-CMU-FNC-1376  H698-LLD-CMU-FNC-1417  H698-LLD-CMU-FNC-1419  H698-LLD-CMU-FNC-1422  H698-LLD-CMU-FNC-1425  H698-LLD-CMU-FNC-1484  H698-LLD-CMU-FNC-1486  H698-LLD-CMU-FNC-1487  H698-LLD-CMU-FNC-1488  H698-LLD-CMU-FNC-1489  H698-LLD-CMU-FNC-1490  H698-LLD-CMU-FNC-1491  H698-LLD-CMU-FNC-1492  H698-LLD-CMU-FNC-1493  H698-LLD-CMU-FNC-1494  H698-LLD-CMU-FNC-1495  H698-LLD-CMU-FNC-1496  H698-LLD-CMU-FNC-1500  H698-LLD-CMU-FNC-1535  H698-LLD-CMU-FNC-1536  H698-LLD-CMU-FNC-1549  H698-LLD-CMU-FNC-905  H698-LLD-CMU-FNC-618  H698-LLD-CMU-FNC-56  H698-LLD-CMU-FNC-57  H698-LLD-CMU-FNC-58  H698-LLD-CMU-FNC-80  H698-LLD-CMU-FNC-754  H698-LLD-CMU-FNC-763  H698-LLD-CMU-FNC-772  H698-LLD-CMU-FNC-781  H698-LLD-CMU-FNC-790  H698-LLD-CMU-FNC-799  H698-LLD-CMU-FNC-687  H698-LLD-CMU-FNC-713  H698-LLD-CMU-FNC-714  H698-LLD-CMU-FNC-633  H698-LLD-CMU-FNC-99  H698-LLD-CMU-FNC-531  H698-LLD-CMU-FNC-550  H698-LLD-CMU-FNC-552  H698-LLD-CMU-FNC-98  H698-LLD-CMU-FNC-827  Newly added ID’s:  H698-LLD-CMU-FNC-2065  H698-LLD-CMU-FNC-2066  H698-LLD-CMU-FNC-2067  H698-LLD-CMU-FNC-2068  H698-LLD-CMU-FNC-2069  H698-LLD-CMU-FNC-2078  H698-LLD-CMU-FNC-2079  H698-LLD-CMU-FNC-2089  H698-LLD-CMU-FNC-2090  H698-LLD-CMU-FNC-2099  H698-LLD-CMU-FNC-2100  H698-LLD-CMU-FNC-2109  H698-LLD-CMU-FNC-2327  H698-LLD-CMU-FNC-2336  H698-LLD-CMU-FNC-2337  H698-LLD-CMU-FNC-2338  H698-LLD-CMU-FNC-2339  H698-LLD-CMU-FNC-2340  H698-LLD-CMU-FNC-2341  H698-LLD-CMU-FNC-2342  H698-LLD-CMU-FNC-2359  H698-LLD-CMU-FNC-2360  H698-LLD-CMU-FNC-2361  H698-LLD-CMU-FNC-2362  H698-LLD-CMU-FNC-2363  Deleted :  H698-LLD-CMU-FNC-950  H698-LLD-CMU-FNC-661  H698-LLD-CMU-FNC-662  H698-LLD-CMU-FNC-663  H698-LLD-CMU-FNC-688  H698-LLD-CMU-FNC-689  H698-LLD-CMU-FNC-690  H698-LLD-CMU-FNC-691  H698-LLD-CMU-FNC-692  H698-LLD-CMU-FNC-1051  H698-LLD-CMU-FNC-1052  H698-LLD-CMU-FNC-1053  H698-LLD-CMU-FNC-1054  H698-LLD-CMU-FNC-1055  H698-LLD-CMU-FNC-1060  H698-LLD-CMU-FNC-1061  H698-LLD-CMU-FNC-1062  H698-LLD-CMU-FNC-1063  H698-LLD-CMU-FNC-1064  H698-LLD-CMU-FNC-906  H698-LLD-CMU-FNC-907  H698-LLD-CMU-FNC-908  H698-LLD-CMU-FNC-917  H698-LLD-CMU-FNC-949  H698-LLD-CMU-FNC-952  H698-LLD-CMU-FNC-620  H698-LLD-CMU-FNC-96  H698-LLD-CMU-FNC-1186  H698-LLD-CMU-FNC-1187  H698-LLD-CMU-FNC-1188  H698-LLD-CMU-FNC-1189  H698-LLD-CMU-FNC-1190  H698-LLD-CMU-FNC-1191  H698-LLD-CMU-FNC-1196  Section 10.1.10 – function A825Timestamp deleted.  Section 10.4 – daucmfarinc8252 file deleted.  Section 10.2 – daucmfa825comm2 file deleted  Section 10.4 – daucmfclk file deleted  Section 10.17 – daucmfsi2c file deleted  Section 11.10 – daulibstm32f4xxsi2c file deleted  Section 10.12.2 to 10.12.2.7 – RamTest file is deleted.  Updated data constants and data dictionary | PR100075 | P Afreen | 19-Dec-2022 |
| 1.5 | Updated sections/requirements as per review comment  H698-LLD-CMU-FNC-20  H698-LLD-CMU-FNC-21  H698-LLD-CMU-FNC-22  H698-LLD-CMU-FNC-31  H698-LLD-CMU-FNC-33  H698-LLD-CMU-FNC-44  H698-LLD-CMU-FNC-55  H698-LLD-CMU-FNC-77  H698-LLD-CMU-FNC-78  H698-LLD-CMU-FNC-81  H698-LLD-CMU-FNC-82  H698-LLD-CMU-FNC-83  H698-LLD-CMU-FNC-97  H698-LLD-CMU-FNC-133  H698-LLD-CMU-FNC-657  H698-LLD-CMU-FNC-658  H698-LLD-CMU-FNC-659  H698-LLD-CMU-FNC-1252  H698-LLD-CMU-FNC-1262  H698-LLD-CMU-FNC-1594  H698-LLD-CMU-FNC-2186  The following LLR ID’s/Sections are updated  H698-LLD-CMU-FNC-2359  H698-LLD-CMU-FNC-2336  H698-LLD-CMU-FNC-2337  H698-LLD-CMU-FNC-2338  H698-LLD-CMU-FNC-2339  H698-LLD-CMU-FNC-2340  H698-LLD-CMU-FNC-2341  Section 10.9.1.2 – List of HLRs allocated  Section 10.12.3.4- Parameter list (Input/Output)  Section 10.12.3.4 – Parameter list (Input/Output)  Section 10.20.1.7.21 – InitNvram function  Section 10.20.1.7.19 – daucmfxram-XramInit-LLR-019  Section 10.20.1.7.20 – IsOlder function  Section 10.3.1.4 – Parameter list (Input/Output)  Section 10.6 – daucmfclk deleted  Section 10.5.2 – PowerOnRamTest added  Section 10.6.1.3 – List of global variables accessed and modified  Section 10.7.1.6 – Other CSUs called by this CSU updated  Sections 10.11.5.7.2 – daucmfoscpua-OsStartHighRdy-LLR-002  Section 10.11.5.7.3 – daucmfoscpua-OsStartHighRdy-LLR-003  Section 10.11.5.7.4 – daucmfoscpua-OsStartHighRdy-LLR-004  Section 10.11.5.7.5 –daucmfoscpua-OsStartHighRdy-LLR-005  Section 10.11.5.7.6 – daucmfoscpua-OsStartHighRdy-LLR-006  Section 10.12.1.5 – Return Value  Section 10.15.2 – Si2cInit  Section 10.12.4 – PbitNvmReadWriteTest  Section 10.14.1 – BbatteryLevelIsGood  Section 10.14.2 – BintegrityIsGood  Section 10.6 – daucmfhw  Section 11.2.5.4 – Parameter list (Input/Output)  Section 11.5 daulibstm32f4xxfsmc is updated.  Section 10.14.3 is updated.  Deleted Id:  H698-LLD-CMU-FNC-673 | PR100075 | P Afreen | 25-Dec-22 |
| 1.6 | Updated as per QA comments  Modified functional requirement  H698-LLD-CMU-FNC-714  Modified General requirement  Section 11 Appendix:Data Dictionary  Section 13 Appendix: Data Constant | PR100075 | P Afreen | 25-Dec-2022 |
| 1.7 | Updated Amendment Record as per QA comments and self review. | PR100075 | P Afreen | 27-Dec-2022 |
| 1.8 | The below requirement and section is updated as per latest SRS and SAD  H698-LLD-CMU-FNC-1059  Appendix B : Data Constants | PR100134 | P Afreen | 06-Jan-2023 |
| 1.9 | The below section is updated for software part number  Appendix B : Data Constants | PR100145 | P Afreen | 19-Jan-2023 |
| 2.0 | The below sections are updated according to the review comments:  Section 2: Objective  Section 4:scope  Section 5:References  Section 8:Traceability | PR100171 | MUPPINENI SRAVANTHI | 12-Feb-2024 |
| 2.1 | The below sections are updated according to the review comments:  Section 5:References | PR100171 | MUPPINENI SRAVANTHI | 14-Feb-2024 |
| 2.2 | The below sections are updated as per review comments  Updated front page section 5  The below sections are updated as per self review  Section 13 | PR100224 | MUPPINENI SRAVANTHI | 02-08-2024 |
| 2.3 | The below section is updated as per review comments  Section13 : Appendix B : Data Constants | PR100224 | MUPPINENI SRAVANTHI | 17-08-2024 |

# 2 Objective

The document H698-003-011-CMU defines the Software Low Level Requirements for CMU Module of the Airbus Engine Data Acquisition Unit (EDAU) and Configuration Management Unit plus NVM (CMU+).

# 3 Acronyms and Definitions

Table 2: List of Acronyms

|  |  |
| --- | --- |
| **Acronym** | **Definition** |
| A825 | ARINC Specification 825 |
| Alten | Alten Global Technology Solutions Pvt. Ltd. |
| AHB | Advanced High-Performance Bus |
| ARINC | Aeronautical Radio Incorporated |
| CAN | Controller Area Network |
| CM | Configuration Module |
| CMS | Configuration Management System |
| CMSIS | Cortex microcontroller software interface standard |
| CMU+ | Configuration Management Unit plus NVM |
| CPU | Central Processing Unit |
| CRC | Cyclic Redundancy Check |
| CSC | Computer Software Component |
| CSU | Computer Software Unit |
| DAU | Data Acquisition Unit |
| DOC | Document Object Code |
| FPCCR | Floating-Point Context Control Register |
| FPSCR | Floating-Point Status Control Register |
| FPU | Floating Point Unit |
| HW | Hardware |
| ICD | Interface Control Document |
| ISR | Interrupt Service Routine |
| LED | Light Emitting Diode |
| NA | Not Applicable |
| NOC | Normal Operating Channel |
| NSC | Node Service Channel |
| QA | Quality Assurance |
| RAM | Random Access Memory |
| RCI | Redundancy Channel Identifier |
| ROM | Read Only Memory |
| SAD | Software Architectural Document |
| SCMP | Software Configuration Management Plan |
| SLL | Software Low Level Requirement |
| SRS | Software Requirement Specifications |
| SVN | SubVersioN |
| TCB | Task Control Block |

# 4 Scope

The document H698-003-011-CMU specifies the Software Low Level Requirements for CMU Module of Airbus Engine Data Acquisition Unit (EDAU) and Configuration Management Unit plus NVM (CMU+). The document H698-003-011-CMU is written according to Software Design Standards (H398-001-007) and satisfies RTCA DO-178B section 11.7 for Software Design Data.

# 5 References

Table 3: Reference document

|  |  |  |
| --- | --- | --- |
| **Source** | **Document No.** | **Title** |
| RTCA | DO-178B | Software Considerations in Airborne Systems and Equipment Certification |
| Howell | SYS2160SRS | Airbus Helicopter AS532U2 Engine Instrument System Requirements Specification |
| Aeronautical Radio, Inc. | ARINC SPECIFICATION 825-2 | GENERAL STANDARDIZATION OF CAN (CONTROLLER AREA NETWORK) BUS PROTOCOL FOR AIRBORNE USE |
| ALTEN Global Technologies **Private Limited** | H398-001-002 | Software Development Plan |
| ALTEN Global Technologies **Private Limited** | H398-001-007 | Software Design Standards |
| ALTEN Global Technologies **Private Limited** | H698-002-001-CMU | Software Requirements Specification for Configuration Module Unit of EDAU |
| ALTEN Global Technologies **Private Limited** | H698-003-001-CMU | Software Architectural Design for CMU Module of Engine Data Acquisition Unit of Airbus EDAU and CMU+ |
| STMicroelectronics | RM0090 Reference manual | Reference manual for STM32F405xx/07xx, STM32F415xx/17xx, STM32F42xxx and STM32F43xxx advanced ARM®-based 32-bit MCUs |
| STMicroelectronics | PM0214 Programming manual | STM32F3 and STM32F4 Series Cortex®-M4 programming manual |

# 6 Document Control

According to the H398-001-004 (SCMP), this document shall be maintained in Alten Software & Software Configuration management System (CMS) after the QA review, under the following path

http://192.168.1.230/svn/A21HOWBLDAU/V2\_H398/ACCORD/SW/Branches/SOI2/Documents/Design/LLD

# 7 Distributions

The Software Low Level Requirements Document of CMU Module(H698-003-011-CMU) is distributed to Howell Instruments, Inc through a secured File Transform Protocol server.

# 8 Traceability

The Software Low Level Requirements Document for CMU Module of Airbus Engine Data Acquisition Unit (EDAU) and Configuration Management Unit plus NVM (CMU+) ( H698-003-011-CMU) is derived from the Software High level Requirements present in the Software High Level Requirements Specification Document for CMU Module of Airbus Engine Data Acquisition Unit (EDAU) and Configuration Management Unit plus NVM (CMU+) (H698-002-001-CMU) and Software Architectural Design for CMU Module of Airbus Engine Data Acquisition Unit (EDAU) and Configuration Management Unit plus NVM (CMU+) (H698-003-001-CMU). Traceability of H698-003-011-CMU to H698-002-001-CMU and H698-003-001-CMU will be provided in Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-012-CMU).

# 9 Responsibilities

1. Software Development Team members specify the Software Low Level Requirements.

2. Reviewers review the Software Low Level Requirements based on the DO-178B check points provided as attributes in ReMa.

3. Once all the comments from the Reviewer are 'Looked Into' and the Software Low Level Requirements status is ‘Closed’ in ReMa, QA changes the QA Review Status Requirement to "QA Approved" and closed.

4. Engineering Manager is responsible for baselining the document to SVN through ReMa, as well as the Exported document, into the path

http://192.168.1.230/svn/A21HOWBLDAU/V2\_H398/ACCORD/SW/Branches/SOI2/Documents/Design/LLD

# 10 Software Low Level Requirements - Flight Module

Software Low Level Requirements-Flight Module section specifies the Software Low Level Requirements for CMU+ Flight Module.

Refer Appendix A: Data Structures of H698-003-001-CMU for Data Structures and Enumerations.

## 10.1 daucmfa825comm

daucmfa825comm CSC implements the A825 communications portion.

### 10.1.1 VSendPartNumberData

Low Level Design Details about CSU VSendPartNumberData will follow in the sub sections.

#### 10.1.1.1 Brief Description

VProcessNOC function transmits software part number and crc on the NOC.

#### 10.1.1.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-011-CMU).

#### 10.1.1.3 List of global variables accessed and modified

Accessed : None

Modified : None

#### 10.1.1.4 Parameter list (Input/Output)

Inputs : T\_UINT16 u16\_doc - IN DOC ID for the associated data

T\_UINT8 \*pu8\_part\_number - IN Address of part number string

T\_UINT32 u32\_crc - IN The part number CRC

Outputs : None

#### 10.1.1.5 Return Value

None

#### 10.1.1.6 Other CSUs called by this CSU

A825Xmit

HwCopy

#### 10.1.1.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to VSendPartNumberData

##### 10.1.1.7.1 daucmfa825comm-VSendPartNumberData-LLR-001

Requirement ID: H698-LLD-CMU-FNC-20

The function shall do the following when pu8\_part\_number is not equal to M\_HW\_NULL, And loop up to the range “M\_ZERO” to M\_REVNO\_LEN minus 1 by incrementing the sizeof u8\_payload of message.

1)Send the partnumber.

2)Copy 8 bytes of part number into message by calling HwCopy with parameters u8\_payload of message, pu8\_part\_number added with u32\_i , sizeof u8\_payload of message,

3)Transmit the data by calling A825Xmit with reference of Message.

4)Increment the DOC ID by increment doc of Broadcast ID of Server Identifer ID of Message.

##### 10.1.1.7.2 daucmfa825comm-VSendPartNumberData-LLR-002

Requirement ID: H698-LLD-CMU-FNC-21

The function shall send CRC by u8\_paysize of message is assigned with size of T\_UINT32 and u8\_payload of message assigned with u32\_crc.

##### 10.1.1.7.3 daucmfa825comm-VSendPartNumberData-LLR-003

Requirement ID: H698-LLD-CMU-FNC-22

The function shall Transmit the data by calling A825Xmit with reference of Message.

### 10.1.2 VSendNvmData

Low Level Design Details about CSU VSendNvmData will follow in the sub sections.

#### 10.1.2.1 Brief Description

VSendNvmData function transmits CMU+ NVM data on the NOC.

#### 10.1.2.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-011-CMU).

#### 10.1.2.3 List of global variables accessed and modified

Accessed : None

Modified : None

#### 10.1.2.4 Parameter list (Input/Output)

Inputs : T\_UINT16 u16\_doc

Outputs : None

#### 10.1.2.5 Return Value

None

#### 10.1.2.6 Other CSUs called by this CSU

HwCopy

OsTimeDly

StartLogger

A825Xmit

#### 10.1.2.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to VSendNvmData

##### 10.1.2.7.1 daucmfa825comm-VSendNvmData-LLR-001

Requirement ID: H698-LLD-CMU-FNC-31

The function shall

a) Set au8\_nvm with index subtraction of (cu32\_transfer\_size and (product of M\_TWO and size of T\_UINT32)) to M\_ZERO, when cu8\_extra\_bytes greater than ZERO.

b) Get Copy BIT data from RAM in case NVM is bad by calling HwCopy with parameters Nvm data, reference to Bit\_status, sizeof (Bit\_status).After disable interrupts.

c) Get a copy of the NVM data by calling HwCopy with parameters reference to Nvm data with index as sizeof(Bit\_status), M\_MEMMAP\_NVM\_DATA, subtraction of (cu32 \_nvm\_data\_size and sizeof(Bit\_status)). After that enable the interrupts.

d) Reset the CRC Data register (DR) before the CRC check.

e) Compute and store the 32-bit CRC of the NVM data to be transfered to au8\_nvm with index subtraction of (cu32\_transfer\_size and size of T\_UINT32) the return value of CrcCalcBlockCrc with parameters au8\_nvm and substraction of (cu32\_transfer\_size divided by size of T\_UINT32) and M\_ONE.

##### 10.1.2.7.2 daucmfa825comm-VSendNvmData-LLR-002

Requirement ID: H698-LLD-CMU-FNC-32

The function shall loop up to the range M\_ZERO to cu32\_nvm\_data\_size minus 1 by incrementing M\_A825\_PAYLOAD\_SIZE.And perform the following: -

1. Set u32\_bytes\_remaining to cu32\_transfer\_size subtracted from u32\_i.

2. When u32\_bytes\_remaining greater than equal to M\_A825\_PAYLOAD\_SIZE set u8\_paysize of message to M\_A825\_PAYLOAD\_SIZE otherwise the perform Modulus Operation with

u32 \_bytes\_remaining and M\_A825\_PAYLOAD\_SIZE take the reminder and set to u8\_paysize of message.

##### 10.1.2.7.3 daucmfa825comm-VSendNvmData-LLR-003

Requirement ID: H698-LLD-CMU-FNC-33

The function shall loop up to the range M\_ZERO to cu32\_nvm\_data\_size minus 1 by incrementing M\_A825\_PAYLOAD\_SIZE and perform the following:-

1. Call the function HwCopy to Copy one payload of data into message with parameters u8\_payload of message, au8\_nvm added to u32\_i, and u8\_paysize of message.
2. Call the function A825Xmit(performing bitwise and operation of messages) function to Transmit data.
3. Call the function OsTimeDly with parameter M\_OS\_TIME\_DELAY\_ONE\_MS to give the time delay for 1ms.
4. Increment the Doc id of Broadcast ID of Server Identifer ID of Message.

##### 10.1.2.7.4 daucmfa825comm-VSendNvmData-LLR-004

Requirement ID: H698-LLD-CMU-FNC-34

The function shall call OsTimeDly with parameter M\_OS\_TIME\_DELAY\_TEN\_MS to perform time delay for 10ms.

##### 10.1.2.7.5 daucmfa825comm-VSendNvmData-LLR-005

Requirement ID: H698-LLD-CMU-FNC-35

The function shall call StartLogger to shuts down the Configuration Management Application and transfers control to the Data Logger Application.

### 10.1.3 VProcessNOC

Low Level Design Details about CSU VProcessNOC will follow in the sub sections.

#### 10.1.3.1 Brief Description

VProcessNOC function processes NOC messages.

#### 10.1.3.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-011-CMU).

#### 10.1.3.3 List of global variables accessed and modified

Accessed : None

Modified : None

#### 10.1.3.4 Parameter list (Input/Output)

Inputs : T\_A825\_MSG smess -IN A825 Message structure.

Outputs : None

#### 10.1.3.5 Return Value

None

#### 10.1.3.6 Other CSUs called by this CSU

None

#### 10.1.3.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to VProcessNOC

##### 10.1.3.7.1 daucmfa825comm-VProcessNOC-LLR-001

Requirement ID: H698-LLD-CMU-FNC-44

The function shall set L\_noc\_cmd to doc ID of broadcast ID of Server Identifer ID of in A825 message structure and returns the function.

### 10.1.4 VProcessNSC

Low Level Design Details about CSU VProcessNSC will follow in the sub sections.

#### 10.1.4.1 Brief Description

VProcessNSC function processes NSC messages.

#### 10.1.4.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-011-CMU).

#### 10.1.4.3 List of global variables accessed and modified

Accessed : None

Modified : None

#### 10.1.4.4 Parameter list (Input/Output)

Inputs : T\_A825\_MSG smess - IN A825 Message structure

Outputs : None

#### 10.1.4.5 Return Value

None

#### 10.1.4.6 Other CSUs called by this CSU

RterrDisplayErrorId

OsSemPostA825Xmit

#### 10.1.4.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to VProcessNSC.

##### 10.1.4.7.1 daucmfa825comm-VProcessNSC-LLR-001

Requirement ID: H698-LLD-CMU-FNC-53

The function shall return when rci of pid of sid of smess is not M\_ZERO, Otherwise do nothing

##### 10.1.4.7.2 daucmfa825comm-VProcessNSC-LLR-002

Requirement ID: H698-LLD-CMU-FNC-54

The function shall do the following when when NSC mode is NSC\_IDLE

- Set command1 with base address of u8\_payload with index M\_ZERO of reference to smess.

a. Command1 with index M\_ZERO equal to SC\_DDS AND command1 with index M\_ONE equal to SDI\_CONFIG\_DATA AND sid of pid of sid of smess equal to CONFIG\_SID

- Set the L not handshaking with DAU to FALSE

- Set the following fields of A825 message

- lcc to NSC

- cfid of pid of sid to M\_HOWELL\_DAU\_FID

- smt of pid of sid to A825\_SMT\_RESPONSE

- lcl of pid of sid to A825\_LCL\_LOCAL

- pvt to A825\_PVT\_PUBLIC

- sfid of pid of sid to M\_HOWELL\_DAU\_FID

- sid of pid of sid to CONFIG\_SID

- rci of pid of sid to M\_ZERO

- u8\_paysize to M\_EIGHT

-u8\_payload with index M\_ZERO to SC\_DDS

- u8\_payload with index M\_TWO to M\_ONE

- u8\_payload with index M\_THREE to M\_ONE

-u8\_payload with index M\_FOUR to M\_MEMMAP\_DATA\_SIZE

- Call A825Xmit with parameter as reference of A825 message structure and store the return value in

Result

-otherwise do nothing.

##### 10.1.4.7.3 daucmfa825comm-VProcessNSC-LLR-003

Requirement ID: H698-LLD-CMU-FNC-55

The function shall do the following and return from the function when NSC mode is NSC\_IDLE

a. Command1 with index M\_ZERO equal to SC\_DDS AND Command1 with index M\_ONE equal to SDI\_CONFIG\_DATA AND sid of pid of sid of smess equal to CONFIG\_SID

- A825\_TX\_OK is equal to result

- set NSC mode to NSC\_SETUP\_1

- A825\_TX\_OK is not equal to result

-Set NSC mode to NSC\_RESET

##### 10.1.4.7.4 daucmfa825comm-VProcessNSC-LLR-004

Requirement ID: H698-LLD-CMU-FNC-56

The function shall do the following when NSC mode is NSC\_IDLE,

Command1 with index M\_ZERO equal to NSC\_GET\_CM\_RCI AND sid of pid of sid of smess equal to CONFIG\_SID.

- lcc to NSC

- cfid of pid of sid to M\_HOWELL\_DAU\_FID

- smt of pid of sid to A825\_SMT\_RESPONSE

- lcl of pid of sid to A825\_LCL\_LOCAL

- pvt of pid of sid to A825\_PVT\_PRIVATE

- sfid of pid of sid to M\_HOWELL\_DAU\_FID

- sid of pid of sid to CONFIG\_SID

- rci of pid of sid to M\_ZERO

- u8\_paysize to M\_FIVE

- Command1 to base address of u8\_payload with index M\_ZERO of A825 message

- Dereferencing of Command1 to NSC\_GET\_CM\_RCI

- Set u8\_payload with index M\_ZERO of A825 message with the following:

- u8\_payload with index M\_TWO to M\_MY\_RCI

- u8\_payload with index M\_THREE to M\_ONE

- u8\_payload with index M\_FOUR to TRUE

- u8\_payload with index M\_FIVE to M\_ZERO

- u8\_payload with index M\_SIX to M\_ZERO

- u8\_payload with index M\_SEVEN to M\_ZERO

- Call A825Xmit with parameter as reference of A825 message structure and store the return value in

Result

-Otherwise do nothing

##### 10.1.4.7.5 daucmfa825comm-VProcessNSC-LLR-005

Requirement ID: H698-LLD-CMU-FNC-57

The function shall set the following(a to g) when the below conditions are satisfied

1. NSC mode is NSC\_SETUP\_1,

2. command1 is set to u8\_payload of index “M\_ZERO ” of reference to smess.

3. command2 is set to u8\_payload of index “M\_FOUR” of reference to smess.

4. command 1 of index “M\_ZERO” is equal to SC\_DDS AND u8\_payload of index “M\_TWO” of smess is equal to M\_ONE AND dereference of command2 is equal to M\_MEMMAP\_DATA\_SIZE AND sid of pid of sid of smess is equal to CONFIG\_SID.

1. lcc to NSC
2. cfid of pid of sid to M\_HOWELL\_DAU\_FID
3. smt of pid of sid to A825\_SMT\_RESPONSE
4. lcl of pid of sid to A825\_LCL\_LOCAL
5. pvt of pid of sid to A825\_PVT\_PUBLIC
6. sfid of pid of sid to M\_HOWELL\_DAU\_FID
7. sid of pid of sid to CONFIG\_SID
8. rci of pid of sid to M\_ZERO
9. u8\_paysize to M\_EIGHT
10. Set u8\_payload of A825 message with the following:
11. u8\_payload with index “M\_ZERO ” to SC\_DDS.
12. u8\_payload with index M\_TWO to M\_ONE.
13. u8\_payload with index M\_THREE to M\_ZERO.
14. u8\_payload with index M\_FOUR to M\_MEMMAP\_DATA\_SIZE.
15. u8\_payload with index M\_SIX to M\_ZERO.

k. Call A825Xmit with parameter as reference of A825 message structure and store the return value in Result.

l. Otherwise do nothing.

##### 10.1.4.7.6 daucmfa825comm-VProcessNSC-LLR-006

Requirement ID: H698-LLD-CMU-FNC-58

The function shall do the following(a to c) when the below conditions are satisfied

1. NSC mode is NSC\_SETUP\_1,

2. command1 is set to u8\_payload of index “M\_ZERO ” of reference to smess.

3. command2 is set to u8\_payload of index “M\_FOUR” of reference to smess.

4. command 1 of index “M\_ZERO” is equal to SC\_DDS AND u8\_payload of index “M\_TWO” of smess is equal to M\_ONE AND dereference of command2 is equal to M\_MEMMAP\_DATA\_SIZE AND sid of pid of sid of smess is equal to CONFIG\_SID.

5. A825\_TX\_OK is assigned to result

a. Set Trigger data to TRUE.

b. Call function RterrDisplayErrorId with parameters M\_APP\_ID, RterrForever, OS\_KERNEL\_ERR bitwise OR with RT\_ERROR\_1 when M\_OS\_NO\_ERR is not equal to the function call OsSemPost with parameter Sem\_a825\_comm\_task1.

c. Otherwise do nothing.

##### 10.1.4.7.7 daucmfa825comm-VProcessNSC-LLR-007

Requirement ID: H698-LLD-CMU-FNC-59

The function shall set NSC mode to NSC\_IDLE when NSC mode is NSC\_RESET.

##### 10.1.4.7.8 daucmfa825comm-VProcessNSC-LLR-008

Requirement ID: H698-LLD-CMU-FNC-60

The function shall do nothing when NSC mode is not equal to NSC\_IDLE, NSC\_SETUP\_1, and NSC\_RESET.

### 10.1.5 A825CommTask

Low Level Design Details about CSU A825CommTask will follow in the sub sections.

#### 10.1.5.1 Brief Description

A825CommTask function is signalled from the A825 receiver routine to process the received message.

#### 10.1.5.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-011-CMU).

#### 10.1.5.3 List of global variables accessed and modified

Accessed : Delayred

Modified : Delayred

#### 10.1.5.4 Parameter list (Input/Output)

Inputs : void \*pdata - Pointer to the data (not used)

Outputs : None

#### 10.1.5.5 Return Value

None

#### 10.1.5.6 Other CSUs called by this CSU

OsTimeDly

GpioSetBits

OsSemPend

VSendPartNumberData

VsendNvmData

HwCopy

A825Xmit

#### 10.1.5.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to A825CommTask.

##### 10.1.5.7.1 daucmfa825comm-A825CommTask-LLR-001

Requirement ID: H698-LLD-CMU-FNC-69

The function shall do the following

* Call the macro M\_UNUSED\_PARAM with parameter pdata to document the unused parameter.

##### 10.1.5.7.2 daucmfa825comm-A825CommTask-LLR-002

Requirement ID: H698-LLD-CMU-FNC-70

The function shall Call function GpioSetBits with parameters (M\_GPIOC and M\_HW\_LED\_GREEN ) and GpioResetBits with parameters (M\_GPIOC and M\_HW\_LED\_RED) when delay red is greater than M\_FIVE, otherwise do nothing.

##### 10.1.5.7.3 daucmfa825comm-A825CommTask-LLR-003

Requirement ID: H698-LLD-CMU-FNC-71

The function shall loop infinitely to process the received message and check delay Red is less than M\_DELAY\_FIVE\_SECONDS

##### 10.1.5.7.4 daucmfa825comm-A825CommTask-LLR-004

Requirement ID: H698-LLD-CMU-FNC-72

The function shall loop infinitely and do the following :

-Increment the delay Red variable when the delay Red is less than M\_DELAY\_FIVE\_SECONDS.

##### 10.1.5.7.5 daucmfa825comm-A825CommTask-LLR-005

Requirement ID: H698-LLD-CMU-FNC-73

The function shall loop infinitely and wait for trigger to send applicatiion data by calling OsSemPend function with parameters Semaphore to signal A825 input processing, M\_ZERO and reference to error code.

##### 10.1.5.7.6 daucmfa825comm-A825CommTask-LLR-006

Requirement ID: H698-LLD-CMU-FNC-74

The function shall loop infinitely and do the following:

-increment the delay when the delay is less than M\_DELAY\_SEVENTEEN.

##### 10.1.5.7.7 daucmfa825comm-A825CommTask-LLR-007

Requirement ID: H698-LLD-CMU-FNC-75

The function shall loop infinitely and do the following:

-Increment delay by one

-set the LED green by calling function GpioSetBits with parameters M\_GPIOC, M\_HW\_LED\_RED(pin15) and GpioSetBits with parameters M\_GPIOC, M\_HW\_LED\_GREEN when

* delay is less than M\_DELAY\_THIRTY
* delay is greater than M\_DELAY\_SEVENTEEN.

##### 10.1.5.7.8 daucmfa825comm-A825CommTask-LLR-008

Requirement ID: H698-LLD-CMU-FNC-76

The function shall loop infinitely and do the following when delay does not satisfy the below condition.

* Delay less than M\_DELAY\_FIVE\_SECONDS
* Delay less than M\_DELAY\_SEVENTEEN
* Delay less than M\_DELAY\_THIRTY

-Call function GpioSetBits with parameters (M\_GPIOC and M\_HW\_LED\_GREEN ) and GpioResetBits with parameters (M\_GPIOC and M\_HW\_LED\_RED) when delay red is greater than M\_FIVE, otherwise do nothing.

* set the boot flags to M\_BOOT\_FLAG\_ADDR
* set the dereference of boot flags by performing Bitwise OR with BOOT\_WARM\_START
* call the function StartLogger

when the L not handshaking with dau AND with (pre incremented value of delay is greater than or equal to M\_TIMEOUT\_SEVEN\_SECOND)

##### 10.1.5.7.9 daucmfa825comm-A825CommTask-LLR-009

Requirement ID: H698-LLD-CMU-FNC-77

The function shall loop infinitely and do the following when NSC\_SETUP\_1 assigned to NSC mode .

1. Increment the time out counter till 2 seconds.
2. When time out counter is greater than M\_TIMEOUT\_TWO\_SECOND then set NSC mode to NSC\_IDLE and trigger data to FALSE. Otherwise do nothing.

##### 10.1.5.7.10 daucmfa825comm-A825CommTask-LLR-010

Requirement ID: H698-LLD-CMU-FNC-78

The function shall loop infinitely and set time out counter to M\_ZERO when NSC\_IDLE assigned to NSC mode otherwise do nothing .

##### 10.1.5.7.11 daucmfa825comm-A825CommTask-LLR-011

Requirement ID: H698-LLD-CMU-FNC-79

The function shall loop infinitely and when trigger data is TRUE then set the following:-

1. Trigger data to FALSE.
2. Calls the OsTimeDly with parameter M\_OS\_TIME\_DELAY\_TEN\_MS.

##### 10.1.5.7.12 daucmfa825comm-A825CommTask-LLR-012

Requirement ID: H698-LLD-CMU-FNC-80

The function shall loop infinitely and do the following when index ranges from M\_ZERO to (M\_MEMMAP\_DATA\_SIZE divided by M\_EIGHT) minus 1 .

And when Trigger data is TRUE.

- lcc to NSC

- cfid of pid of sid to M\_HOWELL\_DAU\_FID

- smt of pid of sid to A825\_SMT\_RESPONSE

- lcl of pid of sid to A825\_LCL\_LOCAL

- pvt of pid of sid to A825\_PVT\_PRIVATE

- sfid of pid of sid to M\_HOWELL\_DAU\_FID

- sid of pid of sid to CONFIG\_SID

- rci of pid of sid to M\_ZERO

- u8\_paysize of out meesage to M\_EIGHT.

-call the function HwCopy with parameters u8\_payload of out message, bitwise and of appdata of index M\_TWO, M\_DATA\_COPY\_COUNT

-call the function OsTimeDly with parameter M\_OS\_TIME\_DELAY\_ONE\_MS.

- Call A825Xmit with parameter as reference of A825 message structure and store the return value in

Result

##### 10.1.5.7.13 daucmfa825comm-A825CommTask-LLR-013

Requirement ID: H698-LLD-CMU-FNC-81

The function shall loop infinitely and call the function OsTimeDly with parameter M\_OS\_TIME\_DELAY\_POINT\_FIVE\_MS and decrement the index when the following conditions are met.

1. Trigger data is TRUE.
2. index ranges from M\_ZERO to (M\_MEMMAP\_DATA\_SIZE divided by M\_EIGHT) minus 1 .
3. A825 transmit OK is not equal to result.

##### 10.1.5.7.14 daucmfa825comm-A825CommTask-LLR-014

Requirement ID: H698-LLD-CMU-FNC-82

The function shall loop infinitely and do nothing when the following conditions are met

1. Trigger data is TRUE
2. index ranges from from M\_ZERO to (M\_MEMMAP\_DATA\_SIZE divided by M\_EIGHT) minus 1 .
3. A825 transmit OK is equal to result.

##### 10.1.5.7.15 daucmfa825comm-A825CommTask-LLR-015

Requirement ID: H698-LLD-CMU-FNC-83

The function shall loop infinitely and set NSC mode to NSC\_IDLE when the following conditions are met.

1. Trigger data is TRUE
2. index is greater than or equal to (M\_MEMMAP\_DATA\_SIZE divided by M\_EIGHT).

##### 10.1.5.7.16 daucmfa825comm-A825CommTask-LLR-016

Requirement ID: H698-LLD-CMU-FNC-84

The function shall loop infinitely and do the following when Trigger data is FALSE and Noc command is DOC\_ACQUIRE\_CRC\_PN .

1. Set the L not handshaking with dau to FALSE

2. call VSendPartNumberData with parameter DOC\_CMUP\_BL\_APP\_PN, M\_MEMMAP\_BOOTLOADER\_REV\_ADDR, M\_MEMMAP\_BOOTLOADER\_CRC\_ADDR. For bootloader application.

3. Call VSendPartNumberData with parameter DOC\_CMUP\_BL\_CONFIG\_PN, M\_MEMMAP\_BL\_CONFIG\_REV\_ADDR, M\_MEMMAP\_BL\_CONFIG\_CRC\_ADDR for Bootloader Configuration.

4. Call VSendPartNumberData with parameter DOC\_CMUP\_CONFIG\_FLIGHT\_PN, M\_MEMMAP\_CONFIG\_FLIGHT\_REV\_ADDR, M\_MEMMAP\_CONFIG\_FLIGHT\_CRC\_ADDR for Configuration Management Flight Application.

5. Call VSendPartNumberData with parameter DOC\_DAU\_CONFIG\_PN, M\_HW\_NULL, M\_MEMMAP\_DATA\_1\_CRC\_ADDR for Aircraft Configuration Data for DAU.

##### 10.1.5.7.17 daucmfa825comm-A825CommTask-LLR-017

Requirement ID: H698-LLD-CMU-FNC-85

The function shall loop infinitely

1. Set the L not handshaking with dau to FALSE
2. call VSendNvmData with parameter DOC\_CMUP\_NVM\_STARTwhen Trigger data is FALSE and Noc command is DOC\_ACQUIRE\_NVM.

##### 10.1.5.7.18 daucmfa825comm-A825CommTask-LLR-018

Requirement ID: H698-LLD-CMU-FNC-86

The function shall loop infinitely and do nothing when trigger data is FALSE and Noc command is not equal to DOC\_ACQUIRE\_CRC\_PN or DOC\_ACQUIRE\_NVM.

##### 10.1.5.7.19 daucmfa825comm-A825CommTask-LLR-019

Requirement ID: H698-LLD-CMU-FNC-87

The function shall loop infinitely and Reset NOC command to idle.

### 10.1.6 A825CommInit

Low Level Design Details about CSU A825CommInit will follow in the sub sections.

#### 10.1.6.1 Brief Description

A825CommInit function initializes the task to handle commands coming across the A825 bus.

#### 10.1.6.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-011-CMU).

#### 10.1.6.3 List of global variables accessed and modified

Accessed : A825\_comm\_task\_stk

Modified : A825\_comm\_task\_stk

#### 10.1.6.4 Parameter list (Input/Output)

Inputs : None

Outputs : None

#### 10.1.6.5 Return Value

None

#### 10.1.6.6 Other CSUs called by this CSU

TbaseTaskSignaling

OsTaskCreate

OsSemCreate

RterrDisplayErrorId

RterrForever

A825CommTask

#### 10.1.6.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to A825CommInit

##### 10.1.6.7.1 daucmfa825comm-A825CommInit-LLR-002

Requirement ID: H698-LLD-CMU-FNC-97

The function shall loop through M\_ZERO to M\_A825COMM\_TASK\_STK\_SIZE minus 1, increment the loop counter and do the following:  
Set A825\_comm\_task\_stk with index as loop counter to M\_CBIT\_TASK\_STK\_VAL.

##### 10.1.6.7.2 daucmfa825comm-A825CommInit-LLR-003

Requirement ID: H698-LLD-CMU-FNC-98

The function shall

* install task signalling parameters by calling TbaseTaskSignaling function with parameters M\_A825COMM\_TASK\_TICKS, return value of

function OsSemCreate with parameter M\_ZERO stored in Semaphore to signal A825 input

processing.

* calls the function RterrDisplayErrorId with parameter (M\_APP\_ID, RterrForever, OS\_KERNEL\_ERR bitwise OR with RT\_ERROR\_2) when Semaphore to signal A825 input processing is M\_NULL, otherwise do nothing.

##### 10.1.6.7.3 daucmfa825comm-A825CommInit-LLR-004

Requirement ID: H698-LLD-CMU-FNC-99

The function shall create the A825Comm task by performing the following when OsTaskCreate with parameters (pointer to A825CommTask function, M\_HW\_NULL, reference of A825\_comm\_task\_stk with index size M\_A825COMM\_TASK\_STK\_SIZE and M\_A825COMM\_TASK\_PRIO) returns value other than M\_OS\_NO\_ERR, otherwise do nothing

-call the function RterrDisplayErrorId with parameters

(M\_APP\_ID, RterrForever, OS\_KERNEL\_ERR Bitwise OR with RT\_ERROR\_3).

### 10.1.7 ResetHardware

Low Level Design Details about CSU ResetHardware will follow in the sub sections.

#### 10.1.7.1 Brief Description

ResetHardware function disables the CAN interfaces and disable and clears pending interrupts.

#### 10.1.7.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-011-CMU).

#### 10.1.7.3 List of global variables accessed and modified

Accessed : None

Modified : None

#### 10.1.7.4 Parameter list (Input/Output)

Inputs : None

Outputs : None

#### 10.1.7.5 Return Value

None

#### 10.1.7.6 Other CSUs called by this CSU

CanItConfig

CanDeInit

OsTimeDly

#### 10.1.7.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to ResetHardware

##### 10.1.7.7.1 daucmfa825comm-ResetHardware-LLR-001

Requirement ID: H698-LLD-CMU-FNC-108

The function shall wait 10ms for pending ARINC825 messages to clear by calling OsTimeDly with parameter M\_OS\_TIME\_DELAY\_TEN\_MS

##### 10.1.7.7.2 daucmfa825comm-ResetHardware-LLR-002

Requirement ID: H698-LLD-CMU-FNC-109

The function shall Disable Interrupts

##### 10.1.7.7.3 daucmfa825comm-ResetHardware-LLR-003

Requirement ID: H698-LLD-CMU-FNC-110

The function shall disable CAN interrupts by calling the following functions

* Call CanItConfig with parameters M\_CANX, M\_CAN\_IT\_FMP0, DISABLE
* Call CanItConfig with parameters M\_CANX, M\_CAN\_IT\_TME, DISABLE
* Call CanItConfig with parameters M\_CANY, M\_CAN\_IT\_FMP0, DISABLE
* Call CanItConfig with parameters M\_CANY, M\_CAN\_IT\_TME, DISABLE
* Call CanDeInit with parameter M\_CANX
* Call CanDeInit with parameters M\_CANY

##### 10.1.7.7.4 daucmfa825comm-ResetHardware-LLR-004

Requirement ID: H698-LLD-CMU-FNC-111

The function shall set ahb1rstr of M\_RCC to Bitwise OR of (ahb1rstr of M\_RCC, M\_RCC\_AHB1PERIPH\_GPIOA, M\_RCC\_AHB1PERIPH\_GPIOB, M\_RCC\_AHB1PERIPH\_GPIOC) to enable Peripherals.

##### 10.1.7.7.5 daucmfa825comm-ResetHardware-LLR-005

Requirement ID: H698-LLD-CMU-FNC-112

The function shall set ahb1rstr of M\_RCC to Negation of (ahb1rstr of M\_RCC Bitwise AND M\_RCC\_AHB1PERIPH\_GPIOA Bitwise OR M\_RCC\_AHB1PERIPH\_GPIOB Bitwise OR M\_RCC\_AHB1PERIPH\_GPIOC) to disable Peripherals.

##### 10.1.7.7.6 daucmfa825comm-ResetHardware-LLR-006

Requirement ID: H698-LLD-CMU-FNC-113

The function shall disable Interrupts in NVIC by setting icer [M\_ZERO to M\_TWO ]of M\_NVIC to 0xFFFFFFFFU

##### 10.1.7.7.7 daucmfa825comm-ResetHardware-LLR-007

Requirement ID: H698-LLD-CMU-FNC-114

The function shall Clear Pending Interrupts in NVIC by setting icpr[M\_ZERO to M\_TWO] of M\_NVIC to 0xFFFFFFFFU

##### 10.1.7.7.8 daucmfa825comm-ResetHardware-LLR-008

Requirement ID: H698-LLD-CMU-FNC-115

The function shall set ctrl of M\_SYSTICK to M\_ZERO.

### 10.1.8 SetStack

Low Level Design Details about CSU SetStack will follow in the sub sections.

#### 10.1.8.1 Brief Description

SetStack function sets the value of the Main Stack Pointer (MSP) and sets the processor to use only

The MSP in "priviledged" mode.

#### 10.1.8.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-011-CMU).

#### 10.1.8.3 List of global variables accessed and modified

Accessed : None

Modified : None

#### 10.1.8.4 Parameter list (Input/Output)

Inputs : T\_UINT32 topOfMainStack - IN Address of the top the stack.

Outputs : None

#### 10.1.8.5 Return Value

None

#### 10.1.8.6 Other CSUs called by this CSU

None

#### 10.1.8.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to SetStack

##### 10.1.8.7.1 daucmfa825comm-SetStack-LLR-001

Requirement ID: H698-LLD-CMU-DRQ-124

The function shall perform the following

* Load the address of the new stack(topOfMainStack) in the MSP
* Reset the CONTROL register to 0. This sets the processor to use only

the MSP and sets the mode to priviledged.

### 10.1.9 StartLogger

Low Level Design Details about CSU StartLogger will follow in the sub sections.

#### 10.1.9.1 Brief Description

StartLogger function shuts down the Configuration Management Application and transfers control to the Data Logger Application.

#### 10.1.9.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-011-CMU).

#### 10.1.9.3 List of global variables accessed and modified

Accessed : None

Modified : None

#### 10.1.9.4 Parameter list (Input/Output)

Inputs : None

Outputs : None

#### 10.1.9.5 Return Value

None

#### 10.1.9.6 Other CSUs called by this CSU

ResetHardware

SetStack

jumptoapplication

#### 10.1.9.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to StartLogger

##### 10.1.9.7.1 daucmfa825comm-StartLogger-LLR-001

Requirement ID: H698-LLD-CMU-FNC-133

The function shall:

- set the function pointer ‘jump to Application’ to addition of M\_MEMMAP\_LOGGER\_FLIGHT\_ADDR and size of data type (T\_UINT32).

- Call function CrcResetDr.

- copy the power\_on status of Bit\_status to deference of function GetPbitNvmAddress. when

- The return value of CrcCalcBlockCrc function with parameters (M\_MEMMAP\_LOGGER\_FLIGHT\_ADDR, M\_MEMMAP\_LOGGER\_FLIGHT\_CRC\_CNT) is equal to M\_MEMMAP\_LOGGER\_FLIGHT\_CRC\_ADDR.

-Otherwise call RterrDisplayErrorId function with parameters (M\_APP\_ID, RterrForever, CRC\_ERROR bitwise OR RT\_ERROR\_2)

##### 10.1.9.7.2 daucmfa825comm-StartLogger-LLR-002

Requirement ID: H698-LLD-CMU-FNC-134

The function shall disable peripherals and interrupts by calling ‘ResetHardware’.

##### 10.1.9.7.3 daucmfa825comm-StartLogger-LLR-003

Requirement ID: H698-LLD-CMU-FNC-135

The function shall initialize the Data Logger's Stack Pointer by calling ‘SetStack’ with parameter as dereferencing of M\_MEMMAP\_LOGGER\_FLIGHT\_ADDR.

##### 10.1.9.7.4 daucmfa825comm-StartLogger-LLR-004

Requirement ID: H698-LLD-CMU-FNC-136

The function shall Jump to the Data Logger Application by calling function pointer ‘ jump to Application’.

## 10.2 daucmfarinc825

daucmfarinc825 module contains implementation of A825 routines.

### 10.2.1 RxQueueRemove

Low Level Design Details about CSU RxQueueRemove will follow in the sub sections.

#### 10.2.1.1 Brief Description

The function RxQueueRemove retrieves a message from the specified receive queue and decomposes the CAN message into an A825 message.

#### 10.2.1.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD (H698-003-012-CMU).

#### 10.2.1.3 List of global variables accessed and modified

Accessed : None

Modified : None

#### 10.2.1.4 Parameter list (Input/Output)

Inputs : T\_CAN\_RX\_QUEUE \*ps\_which\_q - The receive queue to retrieve from

Outputs : T\_A825\_MSG \*ps\_destination - Where to place the retrieved message

T\_CAN\_RX\_QUEUE \*ps\_which\_q - The receive queue to retrieve from buffer

#### 10.2.1.5 Return Value

T\_A825\_REC\_RESULT - A825 Receiver routines result

- returns A825\_RX\_QUEUE\_EMPTY if the message queue is empty

- returns A825\_RX\_OK if A825 MESSAGE is successfully copied to the destination.

#### 10.2.1.6 Other CSUs called by this CSU

CanItConfig

#### 10.2.1.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to RxQueueRemove

##### 10.2.1.7.1 daucmfarinc825-RxQueueRemove-LLR-001

Requirement ID: H698-LLD-CMU-FNC-180

The function shall return A825\_RX\_QUEUE\_EMPTY when the number of messages in the Rx queue is M\_ZERO (rx\_cntr of ps\_which\_q) otherwise do nothing.

##### 10.2.1.7.2 daucmfarinc825-RxQueueRemove-LLR-002

Requirement ID: H698-LLD-CMU-FNC-181

The function shall disable the Can Receiver Transmit Interrupt by calling CanItConfig function with

parameters M\_CANX, M\_CAN\_IT\_FMP0 and DISABLE.

##### 10.2.1.7.3 daucmfarinc825-RxQueueRemove-LLR-003

Requirement ID: H698-LLD-CMU-FNC-182

The function shall retrieve the CAN message from the receive queue (rx\_head of ps\_which\_q).

##### 10.2.1.7.4 daucmfarinc825-RxQueueRemove-LLR-004

Requirement ID: H698-LLD-CMU-FNC-183

The function shall adjust the head and check for a wrap by decrementing number of messages in the

Rx queue (rx\_cntr of ps\_which\_q) by M\_ONE and incrementing the next message to receive (rx\_head of ps\_which\_q) by M\_ONE.

##### 10.2.1.7.5 daucmfarinc825-RxQueueRemove-LLR-005

Requirement ID: H698-LLD-CMU-FNC-184

The function shall set next message to receive (rx\_head of ps\_which\_q) to the Rx buffer (rx\_buf of ps\_which\_q) when the next message to receive is in the message queue (rx\_head of ps\_which\_q) size range (rx\_buf of M\_CAN\_R\_Q\_SIZE of ps\_which\_q) otherwise do nothing.

##### 10.2.1.7.6 daucmfarinc825-RxQueueRemove-LLR-006

Requirement ID: H698-LLD-CMU-FNC-185

The function shall enable the Can Receiver Interrupt by calling CanItConfig function with parameters M\_CANX, M\_CAN\_IT\_FMP0 and ENABLE.

##### 10.2.1.7.7 daucmfarinc825-RxQueueRemove-LLR-007

Requirement ID: H698-LLD-CMU-FNC-186

The function shall set the Logical communication channel of A825 Broadcast MESSAGE to ((can message extended identifier bit shifted to right by M\_LCC\_LOC) bitwise AND with M\_LCC\_MSK)

(Ref. to the design document ARINC SPECIFICATION 825-2 for extracting the data from CAN message)

- LCC [bit 26...28].

##### 10.2.1.7.8 daucmfarinc825-RxQueueRemove-LLR-008

Requirement ID: H698-LLD-CMU-FNC-187

The function shall translate CAN MESSAGE to A825 Broadcast MESSAGE

when the message is a broadcast message i.e. LCC of A825 message is NOC

(Ref. to the design document ARINC SPECIFICATION 825-2 for extracting the data from CAN message)

- (can message extended identifier bit shifted to right by M\_RCI\_LOC) bitwise AND with M\_RCI\_MSK)

- RCI [bit 0...1]

- (can message extended identifier bit shifted to right by M\_DOC\_LOC) bitwise AND with M\_DOC\_MSK)

- DOC [bit 2 ... 15]

- (can message extended identifier bit shifted to right by M\_PVT\_LOC) bitwise AND with M\_PVT\_MSK)

- PVT [bit 16]

- (can message extended identifier bit shifted to right by M\_LCL\_LOC) bitwise AND with M\_LCL\_MSK)

- LCL [bit 17]

- (can message extended identifier bit shifted to right by M\_RSD\_LOC) bitwise AND with M\_RSD\_MSK)

- RSD [bit 18]

- (can message extended identifier bit shifted to right by M\_SRC\_FID\_LOC) bitwise AND with M\_SRC\_FID\_MSK)

- Source FID [bit 19 ... 25]

##### 10.2.1.7.9 daucmfarinc825-RxQueueRemove-LLR-009

Requirement ID: H698-LLD-CMU-FNC-188

The function shall translate CAN MESSAGE to A825 Peer to Peer MESSAGE

if the message is a peer to peer message i.e. LCC of A825 message is other than NOC

(Ref. to the design document ARINC SPECIFICATION 825-2 for extracting the data from CAN message)

- (can message extended identifier bit shifted to right by M\_RCI\_LOC) bitwise AND with M\_RCI\_MSK)

- RCI [bit 0...1]

- (can message extended identifier bit shifted to right by M\_SID\_LOC) bitwise AND with M\_SID\_MSK)

- SID [bit 2 ... 8]

- (can message extended identifier bit shifted to right by M\_SRC\_FID\_LOC) bitwise AND with M\_SRC\_FID\_MSK)

- Source FID [bit 9 ... 15]

- (can message extended identifier bit shifted to right by M\_PVT\_LOC) bitwise AND with M\_PVT\_MSK)

- PVT [bit 16]

- (can message extended identifier bit shifted to right by M\_LCL\_LOC) bitwise AND with M\_LCL\_MSK)

- LCL [bit 17]

- (can message extended identifier bit shifted to right by M\_SMT\_LOC) bitwise AND with M\_SMT\_MSK)

- SMT [bit 18]

- (can message extended identifier bit shifted to right by M\_CFID\_LOC) bitwise AND with M\_CFID\_MSK)

- Client FID [bit 19 ... 25]

##### 10.2.1.7.10 daucmfarinc825-RxQueueRemove-LLR-010

Requirement ID: H698-LLD-CMU-FNC-189

The function shall translate CAN message DLC to A825 pay size.

##### 10.2.1.7.11 daucmfarinc825-RxQueueRemove-LLR-011

Requirement ID: H698-LLD-CMU-FNC-190

The function shall set number of bytes of payload of A825 message to M\_A825\_PAYLOAD\_SIZE when number of bytes of payload of A825 message is greater than M\_A825\_PAYLOAD\_SIZE, otherwise do nothing.

##### 10.2.1.7.12 daucmfarinc825-RxQueueRemove-LLR-012

Requirement ID: H698-LLD-CMU-FNC-191

The function shall loop till the loop counter is less than A825 payload size and copies the data from CAN message to the A825 payload.

##### 10.2.1.7.13 daucmfarinc825-RxQueueRemove-LLR-013

Requirement ID: H698-LLD-CMU-FNC-192

The function shall copy A825 MESSAGE to the destination (ps\_destination) and returns A825\_RX\_OK.

### 10.2.2 A825GetMessage

Low Level Design Details about CSU A825GetMessage will follow in the sub sections.

#### 10.2.2.1 Brief Description

The function A825GetMessage retrieves a message from the specified communication channel.

#### 10.2.2.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD (H698-003-012-CMU).

#### 10.2.2.3 List of global variables accessed and modified

Accessed : None

Modified : None

#### 10.2.2.4 Parameter list (Input/Output)

Inputs : T\_LCC\_TYPE echannel - Select the logic communication channel

T\_A825\_MSG \*ps\_destination - Destination address to copy the A825 message

Outputs : None

#### 10.2.2.5 Return Value

T\_A825\_REC\_RESULT - A825 Receiver routines result

- return A825\_RX\_QUEUE\_EMPTY if the message queue is empty

- return A825\_RX\_OK if A825 MESSAGE is successfully copied to the destination.

- return A825\_RX\_BAD\_CHANNEL if the LCC is other than NOC and NSC

#### 10.2.2.6 Other CSUs called by this CSU

RxQueueRemove

#### 10.2.2.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to A825GetMessage.

##### 10.2.2.7.1 daucmfarinc825-A825GetMessage-LLR-001

Requirement ID: H698-LLD-CMU-FNC-201

The function shall retrieve a message from the specified receive queue by calling RxQueueRemove function with parameters (ps\_destination and reference to NOC receive queue) and stores return value of this function in result variable, when the channel is Normal Operating Channel

##### 10.2.2.7.2 daucmfarinc825-A825GetMessage-LLR-002

Requirement ID: H698-LLD-CMU-FNC-202

The function shall retrieve a message from the specified receive queue by calling RxQueueRemove function with parameters (ps\_destination and reference to NSC receive queue) and stores return value of this function in result variable, when the channel is Node Service Channel.

##### 10.2.2.7.3 daucmfarinc825-A825GetMessage-LLR-003

Requirement ID: H698-LLD-CMU-FNC-203

The function shall set the result variable to A825\_RX\_BAD\_CHANNEL when the channel is other than

NOC and NSC.

##### 10.2.2.7.4 daucmfarinc825-A825GetMessage-LLR-004

Requirement ID: H698-LLD-CMU-FNC-204

The function shall return the result variable.

### 10.2.3 RxQueueInsert

Low Level Design Details about CSU RxQueueInsert will follow in the sub sections.

#### 10.2.3.1 Brief Description

RxQueueInsert function inserts a CAN message into the selected queue.

#### 10.2.3.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD (H698-003-012-CMU).

#### 10.2.3.3 List of global variables accessed and modified

Accessed: None

Modified: None

#### 10.2.3.4 Parameter list (Input/Output)

Inputs: T\_CAN\_RX\_MSG \*ps\_new\_mess - The new message to insert

T\_CAN\_RX\_QUEUE \*ps\_which\_q - The queue to place the message in buffer.

Outputs: T\_CAN\_RX\_QUEUE \*ps\_which\_q - The queue to place the message in buffer.

#### 10.2.3.5 Return Value

T\_A825\_REC\_RESULT - A825 Receiver routines result

- return A825\_RX\_OVERFLOW if the buffer is full

- return A825\_RX\_OK if the received message is successfully placed in the queue

#### 10.2.3.6 Other CSUs called by this CSU

None

#### 10.2.3.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to RxQueueInsert.

##### 10.2.3.7.1 daucmfarinc825-RxQueueInsert-LLR-001

Requirement ID: H698-LLD-CMU-FNC-213

The function shall return A825\_RX\_OVERFLOW when

number of messages in the Rx queue (rx\_cntr of ps\_which\_q) is M\_CAN\_R\_Q\_SIZE otherwise do nothing.

##### 10.2.3.7.2 daucmfarinc825-RxQueueInsert-LLR-002

Requirement ID: H698-LLD-CMU-FNC-214

The function shall insert the new message(ps\_new\_mess) into the next message in the queue (rx\_tail of ps\_which\_q).

##### 10.2.3.7.3 daucmfarinc825-RxQueueInsert-LLR-003

Requirement ID: H698-LLD-CMU-FNC-215

The function shall adjust the tail and check for a wrap by incrementing number of messages in the Rx queue (rx\_cntr of ps\_which\_q) by M\_ONE

and incrementing the next message to insert (rx\_tail of ps\_which\_q) by M\_ONE.

##### 10.2.3.7.4 daucmfarinc825-RxQueueInsert-LLR-004

Requirement ID: H698-LLD-CMU-FNC-216

The function shall set next message to insert (rx\_tail of ps\_which\_q) to the Rx buffer (rx\_buf of ps\_which\_q) when the next message to insert (rx\_tail of ps\_which\_q) is

in the message queue (rx\_buf[M\_CAN\_R\_Q\_SIZE] of ps\_which\_q) otherwise do nothing.

##### 10.2.3.7.5 daucmfarinc825-RxQueueInsert-LLR-005

Requirement ID: H698-LLD-CMU-FNC-217

The function shall return A825\_RX\_OK when the received message is successfully placed in the queue.

### 10.2.4 A825ReceiveMessage

Low Level Design Details about CSU A825ReceiveMessage will follow in the sub sections.

#### 10.2.4.1 Brief Description

A825ReceiveMessage function is called directly from the Interrupt Service

Routine to buffer incoming messages for later processing.

The LCC must conform to one of the main A825 channels or the message will be discarded.

#### 10.2.4.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD (H698-003-012-CMU).

#### 10.2.4.3 List of global variables accessed and modified

Accessed: None

Modified: None

#### 10.2.4.4 Parameter list (Input/Output)

Inputs: None

Outputs: None

#### 10.2.4.5 Return Value

None.

#### 10.2.4.6 Other CSUs called by this CSU

OsIntEnter

OsIntExit

RxQueueInsert

A825GetMessage

VProcessNOC

VProcessNSC

CanReceive

#### 10.2.4.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to A825ReceiveMessage

##### 10.2.4.7.1 daucmfarinc825-A825ReceiveMessage-LLR-001

Requirement ID: H698-LLD-CMU-FNC-226

The function shall call OsIntEnter to inform uCOS that it is entering the ISR.

##### 10.2.4.7.2 daucmfarinc825-A825ReceiveMessage-LLR-002

Requirement ID: H698-LLD-CMU-FNC-227

The function shall retrieve the CAN message by calling CanReceive function with parameters M\_CANX,

M\_CAN\_FIFO0 and reference to CAN Rx message.

##### 10.2.4.7.3 daucmfarinc825-A825ReceiveMessage-LLR-003

Requirement ID: H698-LLD-CMU-FNC-228

The function shall set Logical communication channel to ( (ext\_id of CAN Rx message bit shifted to right by M\_LCC\_LOC) bitwise AND with M\_LCC\_MSK)

(Ref. to the design document ARINC SPECIFICATION 825-2 for extracting the data from CAN message)

- LCC [bit 26...28].

##### 10.2.4.7.4 daucmfarinc825-A825ReceiveMessage-LLR-004

Requirement ID: H698-LLD-CMU-FNC-229

The function shall do the following:

- Set result variable to return value of function RxQueueInsert with parameters (reference of CAN Rx message, reference of Noc rx queue)

-process the NOC message by calling function VProcessNOC with parameter A825 message structure

when

a) Logical communication channel is NOC

b) Result variable is A825\_RX\_OK

c) Call the function A825GetMessage with parameters (NOC and reference to A825 message structure), when it returns other than A825\_RX\_QUEUE\_EMPTY, otherwise do nothing.

##### 10.2.4.7.5 daucmfarinc825-A825ReceiveMessage-LLR-005

Requirement ID: H698-LLD-CMU-FNC-230

The function shall do nothing when result variable is not equal to A825\_RX\_OK and Logical communication channel is NOC

##### 10.2.4.7.6 daucmfarinc825-A825ReceiveMessage-LLR-006

Requirement ID: H698-LLD-CMU-FNC-231

The function shall DO THE FOLLOWING:

- Set result variable to return value of function RxQueueInsert with parameters (reference of CAN Rx message, reference of Nsc rx queue)

-process the NSC message by calling function VProcessNSC with parameter A825 message structure

when

a) Logical communication channel is NSC

b) result is A825\_RX\_OK

c) call the function A825GetMessage with parameters (NSC and reference to A825 message structure) , when it returns other than A825\_RX\_QUEUE\_EMPTY,otherwise do nothing.

##### 10.2.4.7.7 daucmfarinc825-A825ReceiveMessage-LLR-007

Requirement ID: H698-LLD-CMU-FNC-232

The function shall do nothing when result variable is other than A825\_RX\_OK and Logical communication channel is NSC

##### 10.2.4.7.8 daucmfarinc825-A825ReceiveMessage-LLR-008

Requirement ID: H698-LLD-CMU-FNC-233

The function shall do nothing when the Logical communication channel is other than NOC and NSC.

##### 10.2.4.7.9 daucmfarinc825-A825ReceiveMessage-LLR-009

Requirement ID: H698-LLD-CMU-FNC-234

The function shall call OsIntExit to inform uCOS that it is leaving the ISR.

### 10.2.5 TxQueueInsert

Low Level Design Details about CSU TxQueueInsert will follow in the sub sections.

#### 10.2.5.1 Brief Description

TxQueueInsert function inserts a CAN message into the selected queue.

#### 10.2.5.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD (H698-003-012-CMU).

#### 10.2.5.3 List of global variables accessed and modified

Accessed: None

Modified: None

#### 10.2.5.4 Parameter list (Input/Output)

Inputs: T\_CAN\_TX\_MSG \*ps\_new\_mess - The new message to insert

T\_CAN\_TX\_QUEUE \*ps\_which\_q - The queue to place the message in

Outputs: T\_CAN\_TX\_QUEUE \*ps\_which\_q - The queue to place the message in

#### 10.2.5.5 Return Value

T\_A825\_XMIT\_RESULT - A825 transmit routines result

- return A825\_TX\_OVERFLOW if the buffer is full

- return A825\_TX\_OK if the transmitted message is successfully placed in the queue

#### 10.2.5.6 Other CSUs called by this CSU

CanItConfig

A825BufferCheck

#### 10.2.5.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to TxQueueInsert.

##### 10.2.5.7.1 daucmfarinc825-TxQueueInsert-LLR-001

Requirement ID: H698-LLD-CMU-FNC-243

The function shall disable the Can Transmit Interrupt by calling CanItConfig function with parameters M\_CANX, M\_CAN\_IT\_TME and DISABLE

##### 10.2.5.7.2 daucmfarinc825-TxQueueInsert-LLR-002

Requirement ID: H698-LLD-CMU-FNC-244

The function shall enable the Can Transmit Interrupt by calling CanItConfig function with parameters

M\_CANX, M\_CAN\_IT\_TME and ENABLE and returns A825\_TX\_OVERFLOW

when number of messages in the Tx queue (tx\_cntr of ps\_which\_q) is equal to M\_CAN\_X\_Q\_SIZE

otherwise do nothing.

##### 10.2.5.7.3 daucmfarinc825-TxQueueInsert-LLR-003

Requirement ID: H698-LLD-CMU-FNC-245

The function shall insert the new message(ps\_new\_mess) into the next message in the queue (tx\_tail of ps\_which\_q).

##### 10.2.5.7.4 daucmfarinc825-TxQueueInsert-LLR-004

Requirement ID: H698-LLD-CMU-FNC-246

The function shall adjust the tail and check for a wrap by incrementing number of messages in the Tx queue (tx\_cntr of ps\_which\_q) by M\_ONE

and next message to insert (tx\_tail of ps\_which\_q) by M\_ONE.

##### 10.2.5.7.5 daucmfarinc825-TxQueueInsert-LLR-005

Requirement ID: H698-LLD-CMU-FNC-247

The function shall set next message to insert (tx\_tail of ps\_which\_q) to the Tx buffer (tx\_buf of ps\_which\_q)

when the next message to insert (tx\_tail of ps\_which\_q) is in the Transmit Message Queue Size range(tx\_buf[M\_CAN\_X\_Q\_SIZE] of ps\_which\_q) otherwise do nothing.

##### 10.2.5.7.6 daucmfarinc825-TxQueueInsert-LLR-006

Requirement ID: H698-LLD-CMU-FNC-248

The function shall enable the Can Transmit Interrupt by calling CanItConfig function with parameters M\_CANX, M\_CAN\_IT\_TME and ENABLE.

##### 10.2.5.7.7 daucmfarinc825-TxQueueInsert-LLR-007

Requirement ID: H698-LLD-CMU-FNC-249

The function shall call A825BufferCheck for immediate transmit and returns A825\_TX\_OK when the transmitted message is successfully placed in the queue

### 10.2.6 BuildBroadcastExtId

Low Level Design Details about CSU BuildBroadcastExtId will follow in the sub sections.

#### 10.2.6.1 Brief Description

BuildBroadcastExtId function builds the 29 bit CAN identifier for an A825 Broadcast type message.

#### 10.2.6.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD (H698-003-012-CMU).

#### 10.2.6.3 List of global variables accessed and modified

Accessed: None

Modified: None

#### 10.2.6.4 Parameter list (Input/Output)

Inputs: T\_A825\_MSG \*ps\_xm - Pointer to an A825 message

Outputs: None

#### 10.2.6.5 Return Value

T\_UINT32 29-bit identifier

#### 10.2.6.6 Other CSUs called by this CSU

None

#### 10.2.6.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to BuildBroadcastExtId

##### 10.2.6.7.1 daucmfarinc825-BuildBroadcastExtId-LLR-001

Requirement ID: H698-LLD-CMU-FNC-258

The function shall build the 29 bit CAN identifier for an A825 broadcast type message as mentioned below:

((lcl of bid of sid of ps\_xm bit shifted to left by M\_LCL\_LOC) bitwise OR

(pvt of bid of sid of ps\_xm bit shifted to left by M\_PVT\_LOC) bitwise OR

(doc of bid of sid of ps\_xm bit shifted to left by M\_DOC\_LOC) bitwise OR

(rci of bid of sid of ps\_xm bit shifted to left by M\_RCI\_LOC) bitwise OR

(rsd of bid of sid of ps\_xm bit shifted to left by M\_RSD\_LOC) bitwise OR

(sfid of bid of sid of ps\_xm bit shifted to left by M\_SRC\_FID\_LOC) bitwise OR

(lcc of ps\_xm bit shifted to left by M\_LCC\_LOC))

- RCI [bit 0...1],

- DOC [bit 2 ... 15],

- PVT [bit 16],

- LCL [bit 17],

- RSD [bit 18],

- Source FID [bit 19 ... 25]

- LCC [bit 26 ... 28].

(Ref. to the design document ARINC SPECIFICATION 825-2 for extracting the data from CAN message)

##### 10.2.6.7.2 daucmfarinc825-BuildBroadcastExtId-LLR-002

Requirement ID: H698-LLD-CMU-FNC-259

The function shall return the 29 bit CAN identifier.

### 10.2.7 BuildPeerToPeerExtId

Low Level Design Details about CSU BuildPeerToPeerExtId will follow in the sub sections.

#### 10.2.7.1 Brief Description

BuildPeerToPeerExtId function builds the 29 bit CAN identifier for an A825 Peer to Peer type message.

#### 10.2.7.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD (H698-003-012-CMU).

#### 10.2.7.3 List of global variables accessed and modified

Accessed: None

Modified: None

#### 10.2.7.4 Parameter list (Input/Output)

Inputs: T\_A825\_MSG \*ps\_xm - Pointer to an A825 message

Outputs: None

#### 10.2.7.5 Return Value

T\_UINT32 29-bit identifier

#### 10.2.7.6 Other CSUs called by this CSU

None

#### 10.2.7.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to BuildPeerToPeerExtId

##### 10.2.7.7.1 daucmfarinc825-BuildPeerToPeerExtId-LLR-001

Requirement ID: H698-LLD-CMU-FNC-268

The function shall build the 29 bit CAN identifier for an A825 peer to peer type message as mentioned below:

((cfid of pid of sid of ps\_xm bit shifted to left by M\_CFID\_LOC) bitwise OR

(smt of pid of sid of ps\_xm bit shifted to left by M\_SMT\_LOC) bitwise OR

(lcl of pid of sid of ps\_xm bit shifted to left by M\_LCL\_LOC) bitwise OR

(pvt of pid of sid of ps\_xm bit shifted to left by M\_PVT\_LOC) bitwise OR

(sfid of pid of sid of ps\_xm bit shifted to left by M\_SER\_FID\_LOC) bitwise OR

(sid of pid of sid of ps\_xm bit shifted to left by M\_SID\_LOC) bitwise OR

(rci of pid of sid of ps\_xm bit shifted to left by M\_RCI\_LOC) bitwise OR

(lcc of ps\_xm bit shifted to left by M\_LCC\_LOC))

- RCI [bit 0...1],

- SID [bit 2 ... 8],

- Server FID [bit 9 ... 15]

- PVT [bit 16],

- LCL [bit 17],

- SMT [bit 18],

- Client FID [bit 19 ... 25]

- LCC [bit 26 ... 28].

(Ref. to the design document ARINC SPECIFICATION 825-2 for extracting the data from CAN message)

##### 10.2.7.7.2 daucmfarinc825-BuildPeerToPeerExtId-LLR-002

Requirement ID: H698-LLD-CMU-FNC-269

The function shall return the 29 bit CAN identifier.

### 10.2.8 A825Xmit

Low Level Design Details about CSU A825Xmit will follow in the sub sections.

#### 10.2.8.1 Brief Description

A825Xmit function builds a CAN message from the user supplied A825 message.

Place the newly built CAN message in the queue for that message type.

#### 10.2.8.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD (H698-003-012-CMU).

#### 10.2.8.3 List of global variables accessed and modified

Accessed: None

Modified: None

#### 10.2.8.4 Parameter list (Input/Output)

Inputs: T\_A825\_MSG \*ps\_xm - Pointer to the A825 message to transmit

Outputs: None

#### 10.2.8.5 Return Value

T\_A825\_XMIT\_RESULT - A825 transmit routines result

- return A825\_TX\_OVERFLOW if the buffer is full

- return A825\_ TX \_OK if the transmitted message is successfully placed in the queue

- return A825\_BAD\_CHANNEL if the selected lcc is of UNKNOWN Channel Type

- return A825\_BAD\_PAYLOAD\_SIZE if the payload size is out of range.

#### 10.2.8.6 Other CSUs called by this CSU

BuildBroadcastExtId

BuildPeerToPeerExtId

TxQueueInsert

#### 10.2.8.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to A825Xmit

##### 10.2.8.7.1 daucmfarinc825-A825Xmit-LLR-001

Requirement ID: H698-LLD-CMU-FNC-278

The function shall set NOC transmit queue to local message queue and builds the 29-bit ID by calling BuildBroadcastExtId

with parameter pointer to the A825 message(ps\_xm)

when the selected channel is NOC (lcc of ps\_xm).

##### 10.2.8.7.2 daucmfarinc825-A825Xmit-LLR-002

Requirement ID: H698-LLD-CMU-FNC-279

The function shall set NSC transmit queue to local message queue and builds the 29-bit ID by calling BuildPeerToPeerExtId

with parameter pointer to the A825 message(ps\_xm)

when

a) selected channel is not NOC (lcc of ps\_xm).

b) selected channel is NSC (lcc of ps\_xm).

##### 10.2.8.7.3 daucmfarinc825-A825Xmit-LLR-003

Requirement ID: H698-LLD-CMU-FNC-280

The function shall return A825\_BAD\_CHANNEL when selected channel (lcc of ps\_xm) is other than NOC and NSC.

##### 10.2.8.7.4 daucmfarinc825-A825Xmit-LLR-004

Requirement ID: H698-LLD-CMU-FNC-281

The function shall return A825\_BAD\_PAYLOAD\_SIZE when number of bytes of payload (u8\_paysize of ps\_xm) is greater than M\_A825\_PAYLOAD\_SIZE otherwise do nothing.

##### 10.2.8.7.5 daucmfarinc825-A825Xmit-LLR-005

Requirement ID: H698-LLD-CMU-FNC-282

The function shall fill in the CAN TX message Data as:

i.e. extended identifier is set to the built 29-bit ID, type of identifier to M\_CAN\_ID\_EXT,

type of frame for the message to M\_CAN\_RTR\_DATA and length of the frame to the payload size (u8\_paysize of ps\_xm).

##### 10.2.8.7.6 daucmfarinc825-A825Xmit-LLR-006

Requirement ID: H698-LLD-CMU-FNC-283

The function shall loop till the loop counter is less than A825 paysize (u8\_paysize of ps\_xm) and copies the payload (u8\_payload of ps\_xm) to CAN data (data of CAN TX message).

##### 10.2.8.7.7 daucmfarinc825-A825Xmit-LLR-007

Requirement ID: H698-LLD-CMU-FNC-284

The function shall

1. Insert a CAN message into the selected queue by calling TxQueueInsert function with parameters

reference to Message to be transmitted and pointer to message queue and return value gets stored in result variable.

b) Return result variable.

### 10.2.9 XmitFromBuff

Low Level Design Details about CSU XmitFromBuff will follow in the sub sections.

#### 10.2.9.1 Brief Description

XmitFromBuff function checks if a CAN mailbox is ready, transmit a message from the supplied queue.

#### 10.2.9.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD (H698-003-012-CMU).

#### 10.2.9.3 List of global variables accessed and modified

Accessed: None

Modified: None

#### 10.2.9.4 Parameter list (Input/Output)

Inputs: T\_CAN\_TX\_QUEUE \*ps\_which\_q - CAN Tx queue to transmit message

Outputs: T\_CAN\_TX\_QUEUE \*ps\_which\_q - CAN Tx queue to transmit message

#### 10.2.9.5 Return Value

T\_UINT8 - The number of the mailbox that is used for transmission

#### 10.2.9.6 Other CSUs called by this CSU

CanTransmit

#### 10.2.9.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to XmitFromBuff.

##### 10.2.9.7.1 daucmfarinc825-XmitFromBuff-LLR-001

Requirement ID: H698-LLD-CMU-FNC-293

The function shall transmit a CAN frame message by calling CanTransmit with parameters

M\_CANX, next message to transmit (tx\_head of ps\_which\_q).

##### 10.2.9.7.2 daucmfarinc825-XmitFromBuff-LLR-002

Requirement ID: H698-LLD-CMU-FNC-294

The function shall adjust the head and check for a wrap by decrementing number of messages in the Tx queue (tx\_cntr of ps\_which\_q) by M\_ONE

and incrementing next message to transmit (tx\_head of ps\_which\_q) by M\_ONE

when the result of the function CanTransmit is other than M\_CAN\_TXSTATUS\_NOMAILBOX.

##### 10.2.9.7.3 daucmfarinc825-XmitFromBuff-LLR-003

Requirement ID: H698-LLD-CMU-FNC-295

The function shall set next message to transmit (tx\_head of ps\_which\_q) to the Tx buffer (tx\_buf of ps\_which\_q)

when

1. Result of the function CanTransmit is other than M\_CAN\_TXSTATUS\_NOMAILBOX
2. Next message to transmit (tx\_head of ps\_which\_q) is in the Transmit Message Queue Size range(tx\_buf[M\_CAN\_X\_Q\_SIZE] of ps\_which\_q), otherwise do nothing.

##### 10.2.9.7.4 daucmfarinc825-XmitFromBuff-LLR-004

Requirement ID: H698-LLD-CMU-FNC-296

The function shall do nothing when result of the function CanTransmit is M\_CAN\_TXSTATUS\_NOMAILBOX.

##### 10.2.9.7.5 daucmfarinc825-XmitFromBuff-LLR-005

Requirement ID: H698-LLD-CMU-FNC-297

The function shall return the number of the mailbox that is used for transmission.

### 10.2.10 A825BufferCheck

Low Level Design Details about CSU A825BufferCheck will follow in the sub sections.

#### 10.2.10.1 Brief Description

A825BufferCheck function checks the various transmit buffers (in priority order) and fills the transmit buffers.

#### 10.2.10.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD (H698-003-012-CMU).

#### 10.2.10.3 List of global variables accessed and modified

Accessed: None

Modified: None

#### 10.2.10.4 Parameter list (Input/Output)

Inputs: None

Outputs: None

#### 10.2.10.5 Return Value

None

#### 10.2.10.6 Other CSUs called by this CSU

CanItConfig

XmitFromBuff

#### 10.2.10.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to A825BufferCheck

##### 10.2.10.7.1 daucmfarinc825-A825BufferCheck-LLR-001

Requirement ID: H698-LLD-CMU-FNC-306

The function shall loop till number of messages in the NOC transmit queue is not equal to M\_ZERO and perform the following:

Returns from the function when there is no mailbox free for transmission i.e. the return value of the function XmitFromBuff with parameter reference to NOC transmit queue is M\_CAN\_TXSTATUS\_NOMAILBOX

otherwise do nothing.

##### 10.2.10.7.2 daucmfarinc825-A825BufferCheck-LLR-002

Requirement ID: H698-LLD-CMU-FNC-307

The function shall loop till number of messages in the Tx queue (tx\_cntr of NSC transmit queue) is not equal to M\_ZERO and perform the following:

Returns when there is no mailbox free for transmission i.e. the return value of the function XmitFromBuff

with parameter reference to NSC transmit queue is M\_CAN\_TXSTATUS\_NOMAILBOX

otherwise do nothing.

##### 10.2.10.7.3 daucmfarinc825-A825BufferCheck-LLR-003

Requirement ID: H698-LLD-CMU-FNC-308

The function shall disable the Can Transmit Interrupt by calling CanItConfig function with parameters M\_CANX, M\_CAN\_IT\_TME and DISABLE.

### 10.2.11 CanConfig

Low Level Design Details about CSU CanConfig will follow in the sub sections.

#### 10.2.11.1 Brief Description

CanConfig function performs low level initialization of the CAN Peripheral.

#### 10.2.11.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD (H698-003-012-CMU).

#### 10.2.11.3 List of global variables accessed and modified

Accessed: None

Modified: None

#### 10.2.11.4 Parameter list (Input/Output)

Inputs: None

Outputs: None

#### 10.2.11.5 Return Value

None

#### 10.2.11.6 Other CSUs called by this CSU

RccAhb1PeriphClockCmd

GpioPinAFConfig

RccApb1PeriphClockCmd

GpioInit

CanDeInit

CanFilterInit

CanItConfig

CanInit

#### 10.2.11.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to CanConfig

##### 10.2.11.7.1 daucmfarinc825-CanConfig-LLR-001

Requirement ID: H698-LLD-CMU-FNC-317

The function shall enable GPIO Clock by calling the function RccAhb1PeriphClockCmd with parameters

M\_HW\_CAN1\_GPIO\_CLK and ENABLE.

##### 10.2.11.7.2 daucmfarinc825-CanConfig-LLR-002

Requirement ID: H698-LLD-CMU-FNC-318

The function shall connect CAN pins to AF9(PB8) by calling the function GpioPinAFConfig

with parameters M\_HW\_CAN1\_GPIO\_PORT, M\_HW\_CAN1\_RX\_SOURCE and M\_HW\_CAN1\_AF\_PORT.

##### 10.2.11.7.3 daucmfarinc825-CanConfig-LLR-003

Requirement ID: H698-LLD-CMU-FNC-319

The function shall connect CAN pins to AF9 (PB9) by calling the function GpioPinAFConfig with

parameters M\_HW\_CAN1\_GPIO\_PORT, M\_HW\_CAN1\_TX\_SOURCE and M\_HW\_CAN1\_AF\_PORT.

##### 10.2.11.7.4 daucmfarinc825-CanConfig-LLR-004

Requirement ID: H698-LLD-CMU-FNC-320

The function shall

- set GPIO pins to (M\_HW\_CAN1\_RX\_PIN bitwise OR M\_HW\_CAN1\_TX\_PIN)

- set operating mode for the selected pin to GPIO\_MODE\_AF

- set speed for the selected pin to GPIO\_SPEED\_50MHZ

- set operating output type for the selected pin to GPIO\_OTYPE\_PP

- set operating Pull-up/Pull down for the selected pin to GPIO\_PUPD\_UP

-initializes the GPIOB Peripheral by calling GpioInit function with parameters M\_HW\_CAN1\_GPIO\_PORT

and reference to GPIO Init structure.

##### 10.2.11.7.5 daucmfarinc825-CanConfig-LLR-005

Requirement ID: H698-LLD-CMU-FNC-321

The function shall enable CAN Clock by calling the function RccApb1PeriphClockCmd with parameters

M\_HW\_CAN1\_CLK and ENABLE.

##### 10.2.11.7.6 daucmfarinc825-CanConfig-LLR-006

Requirement ID: H698-LLD-CMU-FNC-322

The function shall de-initialize the CAN Peripheral registers by calling the function CanDeInit with parameter M\_CANX.

##### 10.2.11.7.7 daucmfarinc825-CanConfig-LLR-007

Requirement ID: H698-LLD-CMU-FNC-323

The function shall

- disable time triggered comm mode of CAN Init structure

- enable automatic bus-off management of CAN Init structure

- disable the automatic wake-up mode of CAN Init structure

- disable the non-automatic retransmission mode of CAN Init structure

- disable the Receive FIFO Locked mode of CAN Init structure

- disable the Transmit FIFO priority of CAN Init structure

- set CAN operating mode of CAN Init structure to M\_CAN\_MODE\_NORMAL

- set CAN synchronization jump width of CAN Init structure to M\_CAN\_SJW\_1TQ

- set time quanta in Bit Segment 1 of CAN Init structure to M\_CAN\_BS1\_15TQ

- set time quanta in Bit Segment 2 of CAN Init structure to M\_CAN\_BS2\_5TQ

- set length of a time quantum of CAN Init structure to M\_TWO

- initialize the CAN Peripheral registers by calling the function CanInit with parameters M\_CANX and reference to CAN init structure.

##### 10.2.11.7.8 daucmfarinc825-CanConfig-LLR-008

Requirement ID: H698-LLD-CMU-FNC-324

The function shall

- set filter number of CAN Filter Init structure to M\_ZERO

- set filter mode to be initialized of CAN Filter Init structure to M\_CAN\_FILTERMODE\_IDMASK

- set filter scale of CAN Filter Init structure to M\_CAN\_FILTERSCALE\_32BIT

- set filter identification number high, filter identification number low, filter mask number high, filter mask number low of CAN Filter Init structure

to M\_HEXA\_ZERO

- set FIFO assigned to the filter of CAN Filter Init structure to M\_ZERO

- set can filter activation of CAN Filter Init structure to ENABLE

- configures the CAN reception filter by calling the function CanFilterInit with parameter reference to CAN Filter init structure.

##### 10.2.11.7.9 daucmfarinc825-CanConfig-LLR-009

Requirement ID: H698-LLD-CMU-FNC-325

The function shall enable FIFO 0 message pending Interrupt by calling CanItConfig function with parameters

M\_CANX, M\_CAN\_IT\_FMP0 and ENABLE.

### 10.2.12 NvicConfig

Low Level Design Details about CSU NvicConfig will follow in the sub sections.

#### 10.2.12.1 Brief Description

NvicConfig function sets up the Nested Vectored Interrupt Controller to respond to CAN receive and transmit interrupts.

#### 10.2.12.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD (H698-003-012-CMU).

#### 10.2.12.3 List of global variables accessed and modified

Accessed : None

Modified : None

#### 10.2.12.4 Parameter list (Input/Output)

Inputs : None

Outputs : None

#### 10.2.12.5 Return Value

None

#### 10.2.12.6 Other CSUs called by this CSU

NvicInit

IntrInstall

A825ReceiveMessage

A825BufferCheck

#### 10.2.12.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to NvicConfig.

##### 10.2.12.7.1 daucmfarinc825-NvicConfig-LLR-001

Requirement ID: H698-LLD-CMU-FNC-334

The function shall set up the CAN receive interrupt vector by calling the function IntrInstall with parameters

INTR\_CAN\_1\_RX\_0 and A825ReceiveMessage function.

##### 10.2.12.7.2 daucmfarinc825-NvicConfig-LLR-002

Requirement ID: H698-LLD-CMU-FNC-335

The function shall set up the CAN transmit interrupt vector by calling the function IntrInstall with parameters

INTR\_CAN\_1\_TX and A825BufferCheck function.

##### 10.2.12.7.3 daucmfarinc825-NvicConfig-LLR-003

Requirement ID: H698-LLD-CMU-FNC-336

The function shall

- set IRQ channel of NVIC init structure to CAN1\_RX0\_IRQN

- set pre-emption priority, sub priority level for the IRQ channel CAN1\_RX0\_IRQN of NVIC init structure to M\_HEX\_ZERO

- set IRQ channel defined in CAN1\_RX0\_IRQN of NVIC init structure to ENABLE

- initialize the NVIC Peripheral for the IRQ channel CAN1\_RX0\_IRQN by calling the function NvicInit with parameter reference to NVIC init structure.

##### 10.2.12.7.4 daucmfarinc825-NvicConfig-LLR-004

Requirement ID: H698-LLD-CMU-FNC-337

The function shall

- set IRQ channel of NVIC init structure to CAN1\_TX\_IRQN

- set pre-emption priority, sub priority level for the IRQ channel CAN1\_TX\_IRQN of NVIC init structure to M\_HEX\_ZERO

- set IRQ channel defined in CAN1\_TX\_IRQN of NVIC init structure to ENABLE

- initialize the NVIC Peripheral for the IRQ channel CAN1\_TX\_IRQN by calling the function NvicInit with parameter reference to NVIC init structure.

### 10.2.13 A825Init

Low Level Design Details about CSU A825Init will follow in the sub sections.

#### 10.2.13.1 Brief Description

A825Init function performs initialization necessary for A825 communication.

#### 10.2.13.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD (H698-003-012-CMU).

#### 10.2.13.3 List of global variables accessed and modified

Accessed: None

Modified: None

#### 10.2.13.4 Parameter list (Input/Output)

Inputs: None

Outputs: None

#### 10.2.13.5 Return Value

None

#### 10.2.13.6 Other CSUs called by this CSU

NvicConfig

CanConfig

#### 10.2.13.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to A825Init.

##### 10.2.13.7.1 daucmfarinc825-A825Init-LLR-001

Requirement ID: H698-LLD-CMU-FNC-346

The function shall perform NVIC configuration by calling NvicConfig function.

##### 10.2.13.7.2 daucmfarinc825-A825Init-LLR-002

Requirement ID: H698-LLD-CMU-FNC-347

The function shall perform CAN configuration by calling CanConfig function.

## 10.3 daucmfcbit

daucmfcbit module implements Continuous Built-In-Test functionality.

### 10.3.1 StackTest

Low Level Design Details about CSU StackTest will follow in the sub sections.

#### 10.3.1.1 Brief Description

The function performs Stack Test.

#### 10.3.1.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD (H698-003-012-CMU).

#### 10.3.1.3 List of global variables accessed and modified

Accessed : None

Modified : None

#### 10.3.1.4 Parameter list (Input/Output)

Inputs : T\_OS\_STK \*p\_bottom\_of\_stack - Pointer to the bottom-of-stack address

T\_UINT32 u32\_stack\_size - size of stack.

Outputs : None

#### 10.3.1.5 Return Value

None

#### 10.3.1.6 Other CSUs called by this CSU

RterrDisplayErrorId

RterrForever

#### 10.3.1.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to StackTest.

##### 10.3.1.7.1 daucmfcbit-StackTest-LLR-001

Requirement ID: H698-LLD-CMU-FNC-531

The function shall do the following:

a. Set the dereference of pbuffer to addition of (p\_bottom\_of\_stack and M\_CALC\_30\_PERCENT\_STACKSIZE(u32\_stack\_size))

b. Stack overflow condition check for the last 30 percent of the stack by looping the index till M\_TEN minus 1:

call the RterrDisplayErrorId function with parameters (M\_APP\_ID, RterrForever, (STACK\_OVERFLOW bitwise OR with RT\_ERROR\_1)) when M\_CBIT\_TASK\_STK\_VAL is not equal to dereference of pbuffer decrement by one otherwise do nothing.

### 10.3.2 ContinuousBitTask

Low Level Design Details about CSU ContinuousBitTask will follow in the sub sections.

#### 10.3.2.1 Brief Description

The function ContinuousBitTask increments system tick for signalling semaphore for Continuous Built-In-Test.

#### 10.3.2.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD (H698-003-012-CMU).

#### 10.3.2.3 List of global variables accessed and modified

Accessed : A825\_comm\_task\_stk

Init\_task\_stk

Os\_task\_idle\_stk

Modified : None

#### 10.3.2.4 Parameter list (Input/Output)

Inputs: void \* p\_data - Unused pointer to M\_HW\_NULL data

Outputs: None

#### 10.3.2.5 Return Value

None

#### 10.3.2.6 Other CSUs called by this CSU

StackTest

OsSemPend

#### 10.3.2.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to ContinuousBitTask.

##### 10.3.2.7.1 daucmfcbit-ContinuousBitTask-LLR-001

Requirement ID: H698-LLD-CMU-FNC-540

The function shall document an unused parameter and eliminates the related compiler error using M\_UNUSED\_PARAM with parameter p\_data.

##### 10.3.2.7.2 daucmfcbit-ContinuousBitTask-LLR-002

Requirement ID: H698-LLD-CMU-FNC-541

The function shall loop continuously to perform the following

- call 'OsSemPend' for waiting on a semaphore with the following parameters

- Allocated Semaphore for the task Continuous Built-In-Test

- pend timeout value as M\_ZERO

- Address of error code.

- call 'StackTest' with parameters A825\_comm\_task\_stk, M\_A825COMM\_TASK\_STK\_SIZE.

-call 'StackTest' with parameters Init\_task\_stk, M\_INIT\_TASK\_STK\_SIZE.

- call 'StackTest' with parameters Cbit\_task\_stack, M\_CBIT\_TASK\_STK\_SIZE.

-call 'StackTest' with parameters Os\_task\_idle\_stk, M\_OS\_IDLE\_TASK\_STK\_SIZE.

### 10.3.3 ContinuousBitTaskInit

Low Level Design Details about CSU ContinuousBitTaskInit will follow in the sub sections.

#### 10.3.3.1 Brief Description

The function ContinuousBitTaskInit initializes stack and creates task to implement Continuous Built-In-Test.

#### 10.3.3.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD (H698-003-012-CMU).

#### 10.3.3.3 List of global variables accessed and modified

Accessed : None

Modified : None

#### 10.3.3.4 Parameter list (Input/Output)

Inputs : None

Outputs : None

#### 10.3.3.5 Return Value

None

#### 10.3.3.6 Other CSUs called by this CSU

TbaseTaskSignaling

OsSemCreate

OsTaskCreate

RterrDisplayErrorId

RterrForever

ContinuousBitTask

#### 10.3.3.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to ContinuousBitTaskInit.

##### 10.3.3.7.1 daucmfcbit-ContinuousBitTaskInit-LLR-001

Requirement ID: H698-LLD-CMU-FNC-550

The function shall

a) install task signalling parameters by calling TbaseTaskSignaling function with parameters M\_CBIT\_TASK\_TICKS, return value of

function OsSemCreate with parameter M\_ZERO stored in Semaphore to signal Continuous Built-In-Test.

b) calls the function RterrDisplayErrorId with parameter (M\_APP\_ID, RterrForever function, OS\_KERNEL\_ERR bitwise OR with RT\_ERROR\_4)

when Semaphore to signal Continuous Built-In-Test is M\_NULL, otherwise do nothing.

##### 10.3.3.7.2 daucmfcbit-ContinuousBitTaskInit-LLR-002

Requirement ID: H698-LLD-CMU-FNC-551

The function shall initialize all the stack array indices used for run time checks with M\_CBIT\_TASK\_STK\_VAL by looping through M\_ZERO to M\_CBIT\_TASK\_STK\_SIZE.

##### 10.3.3.7.3 daucmfcbit-ContinuousBitTaskInit-LLR-003

Requirement ID: H698-LLD-CMU-FNC-552

The function 'OsTaskCreate' shall be called to create ContinuousBitTask with arguments Pointer to the ContinuousBitTask function, M\_NULL, reference of Continuous Bit Task stack of index M\_CBIT\_TASK\_STK\_SIZE, Task priority as M\_CBIT\_TASK\_PRIO and calls the function RterrDisplayErrorId with parameter (M\_APP\_ID, RterrForever, OS\_KERNEL\_ERR bitwise OR with RT\_ERROR\_5) when the function OsTaskCreate returns other than M\_OS\_NO\_ERR, otherwise do nothing.

## 10.4 daucmfcputest

daucmfcputest module contains the routine to implement the CPU test.

### 10.4.1 AluTest

Low Level Design Details about CSU AluTest will follow in the sub sections.

#### 10.4.1.1 Brief Description

AluTest function checks all the Arithmetic and Logic operations.

#### 10.4.1.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD (H698-003-012-CMU).

#### 10.4.1.3 List of global variables accessed and modified

Accessed : Cpu\_test\_res

Modified : Cpu\_test\_res

#### 10.4.1.4 Parameter list (Input/Output)

Inputs : None

Outputs : None

#### 10.4.1.5 Return Value

None

#### 10.4.1.6 Other CSUs called by this CSU

None

#### 10.4.1.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to AluTest.

##### 10.4.1.7.1 daucmfcputest-AluTest-LLR-001

Requirement ID: H698-LLD-CMU-FNC-598

The function shall perform the addition operation on registers (R1, R2) with operands M\_CONST\_SIX and M\_CONST\_FIVE and if the obtained result (R2) matches with the value M\_CONST\_ELEVEN then perform carry flag test otherwise exit from the subroutine.

##### 10.4.1.7.2 daucmfcputest-AluTest-LLR-002

Requirement ID: H698-LLD-CMU-FNC-599

The function shall perform the addition operation on registers (R1, R2) with operands M\_CONST\_SIX and M\_CONST\_FIVE and store the status of the carry flag from APSR into the register (R4).

If the obtained result (R4) matches with the value M\_CONST\_CARRY\_FLAG then perform zero flag test otherwise exit from the subroutine.

##### 10.4.1.7.3 daucmfcputest-AluTest-LLR-003

Requirement ID: H698-LLD-CMU-FNC-600

The function shall perform the addition operation on registers (R1, R2) with operands M\_CONST\_SIX and M\_CONST\_FIVE and if the obtained result (R2) matches with M\_CONST\_ELEVEN then store the status of the overflow flag from APSR into the register (R4).

If the obtained result (R4) matches with the value M\_CONST\_ZERO\_FLAG then perform overflow flag test otherwise exit from the subroutine.

##### 10.4.1.7.4 daucmfcputest-AluTest-LLR-004

Requirement ID: H698-LLD-CMU-FNC-601

The function shall perform the addition operation on registers (R1, R2) with operands M\_CONST\_PATTERN and M\_CONST\_PATTERN and store the status of the zero flag from APSR into the register (R4).

If the obtained result (R4) matches with the value M\_CONST\_OVERFLOW\_FLAG then perform subtract test otherwise exit from the subroutine.

##### 10.4.1.7.5 daucmfcputest-AluTest-LLR-005

Requirement ID: H698-LLD-CMU-FNC-602

The function shall perform the subtract operation on registers (R1, R2) with operands M\_CONST\_SIX and M\_CONST\_FIVE and if the obtained result (R1) matches with the value M\_CONST\_ONE then perform multiply test otherwise exit from the subroutine.

##### 10.4.1.7.6 daucmfcputest-AluTest-LLR-006

Requirement ID: H698-LLD-CMU-FNC-603

The function shall perform the multiply operation on registers (R1, R2) with operands M\_CONST\_THREE and M\_CONST\_TWO and if the obtained result (R1) matches with the value M\_CONST\_SIX then perform division test otherwise exit from the subroutine.

##### 10.4.1.7.7 daucmfcputest-AluTest-LLR-007

Requirement ID: H698-LLD-CMU-FNC-604

The function shall perform the unsigned integerdivision operation on registers (R1, R2) with operands M\_CONST\_FOUR and M\_CONST\_TWO and if the obtained result (R1) matches with the value M\_CONST\_TWO then perform logical and test otherwise exit from the subroutine.

##### 10.4.1.7.8 daucmfcputest-AluTest-LLR-008

Requirement ID: H698-LLD-CMU-FNC-605

The function shall perform the logical AND operation on registers (R1,R2) with operands M\_CONST\_HEX\_ONE and M\_CONST\_SEVENTEEN and if the obtained result (R3) matches with the value M\_CONST\_ONE then perform logical or test otherwise exit from the subroutine.

##### 10.4.1.7.9 daucmfcputest-AluTest-LLR-009

Requirement ID: H698-LLD-CMU-FNC-606

The function shall perform the logical OR operation on registers (R1, R2) with operands M\_CONST\_SIXTEEN and M\_CONST\_HEX\_ONE and if the obtained result (R3) matches with the value M\_CONST\_SEVENTEEN then perform exclusive or test otherwise exit from the subroutine.

##### 10.4.1.7.10 daucmfcputest-AluTest-LLR-010

Requirement ID: H698-LLD-CMU-FNC-607

The function shall perform the logical XOR operation on registers (R1, R2) with operands M\_CONST\_SEVENTEEN and M\_CONST\_HEX\_ONE and if the obtained result (R3) matches with the value M\_CONST\_SIXTEEN then branch to Test result otherwise exit from the subroutine.

##### 10.4.1.7.11 daucmfcputest-AluTest-LLR-011

Requirement ID: H698-LLD-CMU-FNC-608

The function shall initialize R4 with M\_CONST\_ZERO and load the address of Cpu\_test\_res to the R2 register to set the global variable Cpu\_test\_res to 0.

## 10.5 daucmfcrt0

daucmfcrt0 module copies and resets the Vector Table for STM32F40x.

After Reset the Cortex-M4 processor is in Thread mode, priority is Privileged and the Stack is set to Main.

### 10.5.1 Crt0TransferData

Low Level Design Details about CSU Crt0TransferData will follow in the sub sections.

#### 10.5.1.1 Brief Description

Crt0TransferData function is reset handler for reset condition assigned to the interrupt section.

#### 10.5.1.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD (H698-003-012-CMU).

#### 10.5.1.3 List of global variables accessed and modified

Accessed : \_data\_rom

\_data\_start

\_data\_end

\_bss\_start

\_bss\_end

Modified : \_bss\_start

\_data\_start

#### 10.5.1.4 Parameter list (Input/Output)

Inputs: None

Outputs: None

#### 10.5.1.5 Return Value

None

#### 10.5.1.6 Other CSUs called by this CSU

MainFunc

PowerOnRamTest

#### 10.5.1.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to Crt0TransferData.

##### 10.5.1.7.1 daucmfcrt0-Crt0TransferData-LLR-001

Requirement ID: H698-LLD-CMU-FNC-618

The function shall call the function PowerOnRamTest with paramters CCM\_RAM\_START, CCM\_RAM\_SIZE and call the function PowerOnRamTest with paramters RAM\_START, RAM\_SIZE

##### 10.5.1.7.2 daucmfcrt0-Crt0TransferData-LLR-002

Requirement ID: H698-LLD-CMU-FNC-619

The function shall move from dereference of source to dereference of destination by looping till reference of \_data\_end and increment the source and destination by one for each iteration.

##### 10.5.1.7.3 daucmfcrt0-Crt0TransferData-LLR-004

Requirement ID: H698-LLD-CMU-FNC-621

The function shall call MainFunc to branch to main function.

### 10.5.2 PowerOnRamTest

Low Level Design Details about PowerOnRamTest will follow in the sub sections.

#### 10.5.2.1 Brief Description

This function does the Internal RAM test for all RAM by writing a set of data to each address in the memory device and verifies the data by reading it back. If all the values read back are the same as those that were written, then the memory device is said to pass the test.

#### 10.5.2.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-012-CMU).

#### 10.5.2.3 List of global variables accessed and modified

Accessed: None

Modified: None

#### 10.5.2.4 Parameter list (Input/Output)

Input: T\_UINT32 \*pu32\_start – In start value

T\_UINT32 u32\_size – In size

Output: None

#### 10.5.2.5 Return Value

None

#### 10.5.2.6 Other CSUs called by this CSU

RterrDisplayErrorId

RterrForever

#### 10.5.2.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to PowerOnRamTest

##### 10.5.2.7.1 daucmfcrt0-PowerOnRamTest-LLR-001

Requirement ID: H698-LLD-CMU-FNC-2078

The function shall do the following when it's b status good AND memory is greater than start.

* Set the TEST\_AA data to pre decrement of particular address in the memory device, status remains good only if read back value is same as written value. (b status good AND (TEST\_AA is equal to reference of pu32 memory)).
* Set the TEST\_55 data to particular address in the memory device, status remains good only if read back value is same as written value (b status good AND (TEST\_55 is equal to reference of pu32 memory)).
* Set the TEST\_FF data to particular address in the memory device, status remains good only if read back value is same as written value (b status good AND (TEST\_FF is equal to reference of pu32 memory)).
* Set the TEST\_00 data to particular address in the memory device, status remains good only if read back value is same as written value (b status good AND (TEST\_00 is equal to reference of pu32 memory)).

##### 10.5.2.7.2 daucmfcrt0-PowerOnRamTest-LLR-002

Requirement ID: H698-LLD-CMU-FNC-2079

The PowerOnRamTest shall calls the RterrDisplayErrorId function with parameters M\_APP\_ID, RterrForever, MEMORY\_ERROR bitwise OR with RT\_ERROR\_1 to display the input error on the Digital Display and then locks up the indicator when any pattern fails (b status good is equals to FALSE).

## 10.6 daucmfhw

daucmfhw CSC provides definition of some common routines and Processor start-up routine.

### 10.6.1 HwInit

Low Level Design Details about CSU HwInit will follow in the sub sections.

#### 10.6.1.1 Brief Description

HwInit function initialize Hardware related components.

#### 10.6.1.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-012-CMU).

#### 10.6.1.3 List of global variables accessed and modified

Accessed: None

Modified: None

#### 10.6.1.4 Parameter list (Input/Output)

Inputs: None

Outputs: None

#### 10.6.1.5 Return Value

None

#### 10.6.1.6 Other CSUs called by this CSU

RterrDisplayErrorId

IntrInit

IntrInstall

TbaseIntrHandler

PendSvHandler

RccDeInit

RccHseConfig

RccWaitForHseStartUp

RccApb1PeriphClockCmd

PwrMainRegulatorModeConfig

RccHclkConfig

RccPclk1Config

RccPclk2Config

FlashSetLatency

FlashInstructionCacheCmd

FlashDataCacheCmd

FlashPrefetchBufferCmd

RccPllConfig

RccPllCmd

RccGetFlagStatus

RccSysClkConfig

RccGetSysClkSource

NvicSetVectorTable

NvicPriorityGroupConfig

NvicSetPriority

RccAhb1PeriphClockCmd

GpioInit

GpioReadInputDataBit

GpioSetBits

RterrForever

#### 10.6.1.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to HwInit.

##### 10.6.1.7.1 daucmfhw-HwInit-LLR-001

Requirement ID: H698-LLD-CMU-FNC-631

The function shall reset the RCC Clock configuration to the default reset state by calling 'RccDeInit'.

##### 10.6.1.7.2 daucmfhw-HwInit-LLR-002

Requirement ID: H698-LLD-CMU-FNC-632

The function shall enable the External High Speed oscillator (HSE) by calling 'RccHseConfig' with parameter M\_RCC\_HSE\_ON.

##### 10.6.1.7.3 daucmfhw-HwInit-LLR-003

Requirement ID: H698-LLD-CMU-FNC-633

The function shall display the input error by calling function ‘RterrDisplayErrorId’ with parameters M\_APP\_ID, RterrForever function and CPU\_ERROR bitwise OR with RT\_ERROR\_1 when ‘RccWaitForHseStartUp’ function returns ERROR, otherwise do nothing.

##### 10.6.1.7.4 daucmfhw-HwInit-LLR-004

Requirement ID: H698-LLD-CMU-FNC-634

The function shall enable the Low Speed APB(APB1) Peripheral Power Interface Clock by calling 'RccApb1PeriphClockCmd' with parameters

M\_RCC\_APB1PERIPH\_PWR, ENABLE.

##### 10.6.1.7.5 daucmfhw-HwInit-LLR-005

Requirement ID: H698-LLD-CMU-FNC-635

The function shall set Regulator voltage scaling output selection bit to scale 1 mode

in the Power Control Register by calling 'PwrMainRegulatorModeConfig'

with parameter M\_PWR\_REGULATOR\_VOLTAGE\_SCALE1.

##### 10.6.1.7.6 daucmfhw-HwInit-LLR-006

Requirement ID: H698-LLD-CMU-FNC-636

The function shall configure the AHB Clock (HCLK) to System Clock(SYSCLK) by calling 'RccHclkConfig' with parameter M\_RCC\_SYSCLK\_DIV1.

##### 10.6.1.7.7 daucmfhw-HwInit-LLR-007

Requirement ID: H698-LLD-CMU-FNC-637

The function shall configure the Low Speed APB Clock (PCLK1) by calling 'RccPclk1Config' with parameter M\_RCC\_HCLK\_DIV4.

##### 10.6.1.7.8 daucmfhw-HwInit-LLR-008

Requirement ID: H698-LLD-CMU-FNC-638

The function shall configure the High Speed APB Clock (PCLK2) by calling 'RccPclk2Config' with parameter M\_RCC\_HCLK\_DIV2.

##### 10.6.1.7.9 daucmfhw-HwInit-LLR-009

Requirement ID: H698-LLD-CMU-FNC-639

The function shall set the code latency value by calling 'FlashSetLatency' with parameter M\_FLASH\_LATENCY\_5.

##### 10.6.1.7.10 daucmfhw-HwInit-LLR-010

Requirement ID: H698-LLD-CMU-FNC-640

The function shall enable the Instruction Cache feature by calling 'FlashInstructionCacheCmd' with parameter ENABLE.

##### 10.6.1.7.11 daucmfhw-HwInit-LLR-011

Requirement ID: H698-LLD-CMU-FNC-641

The function shall enable the Data Cache feature by calling 'FlashDataCacheCmd' with parameter ENABLE.

##### 10.6.1.7.12 daucmfhw-HwInit-LLR-012

Requirement ID: H698-LLD-CMU-FNC-642

The function shall enable Pre-fetch buffer by calling 'FlashPrefetchBufferCmd' with parameter ENABLE.

##### 10.6.1.7.13 daucmfhw-HwInit-LLR-013

Requirement ID: H698-LLD-CMU-FNC-643

The function shall configure the main PLL Clock source, multiplication and division factors by calling 'RccPllConfig' with

parameters M\_RCC\_PLLSOURCE\_HSE, M\_HW\_PLL\_M, M\_HW\_PLL\_N, M\_HW\_PLL\_P and M\_HW\_PLL\_Q.

##### 10.6.1.7.14 daucmfhw-HwInit-LLR-014

Requirement ID: H698-LLD-CMU-FNC-644

The function shall enable main PLL by calling 'RccPllCmd' with parameter ENABLE.

##### 10.6.1.7.15 daucmfhw-HwInit-LLR-015

Requirement ID: H698-LLD-CMU-FNC-645

The function shall enter into wait state until PLL is ready i.e. function RccGetFlagStatus with parameter M\_RCC\_FLAG\_PLLRDY returns SET.

##### 10.6.1.7.16 daucmfhw-HwInit-LLR-016

Requirement ID: H698-LLD-CMU-FNC-646

The function shall set the PLL as System Clock by calling 'RccSysClkConfig' with parameter M\_RCC\_SYSCLKSOURCE\_PLLCLK.

##### 10.6.1.7.17 daucmfhw-HwInit-LLR-017

Requirement ID: H698-LLD-CMU-FNC-647

The function shall enter into wait state until PLL is used as System Clock source i.e. function RccGetSysClkSource returns value M\_PLL\_SYS\_CLK.

##### 10.6.1.7.18 daucmfhw-HwInit-LLR-018

Requirement ID: H698-LLD-CMU-FNC-648

The function shall initialize the interrupt Vector Table in RAM by calling 'IntrInit'.

##### 10.6.1.7.19 daucmfhw-HwInit-LLR-019

Requirement ID: H698-LLD-CMU-FNC-649

The function shall set the Vector Table base address at RAM by calling 'NvicSetVectorTable' with parameters M\_NVIC\_VECTTAB\_RAM and M\_VT\_BSE\_ADDR\_OFFSET.

##### 10.6.1.7.20 daucmfhw-HwInit-LLR-020

Requirement ID: H698-LLD-CMU-FNC-650

The function shall configure the group priority and sub-priority by calling 'NvicPriorityGroupConfig' with parameter M\_NVIC\_PRIORITYGROUP\_4.

##### 10.6.1.7.21 daucmfhw-HwInit-LLR-021

Requirement ID: H698-LLD-CMU-FNC-651

The function shall set up the System tick interrupt into vector table in RAM by calling IntrInstall function with parameters INTR\_SYS\_TICK and TbaseIntrHandler function.

##### 10.6.1.7.22 daucmfhw-HwInit-LLR-022

Requirement ID: H698-LLD-CMU-FNC-652

The function shall set up the priority of the System tick interrupt by calling NvicSetPriority function with parameters SYSTICK\_IRQN and M\_SET\_PRIORITY\_15.

##### 10.6.1.7.23 daucmfhw-HwInit-LLR-023

Requirement ID: H698-LLD-CMU-FNC-653

The function shall set up the PendSV interrupt into Vector Table in RAM by calling IntrInstall function with parameters

INTR\_PEND\_SV and PendSvHandler function.

##### 10.6.1.7.24 daucmfhw-HwInit-LLR-024

Requirement ID: H698-LLD-CMU-FNC-654

The function shall set up the priority of the PendSV interrupt by calling NvicSetPriority function with parameters PENDSV\_IRQN and M\_SET\_PRIORITY\_14.

##### 10.6.1.7.25 daucmfhw-HwInit-LLR-025

Requirement ID: H698-LLD-CMU-FNC-655

The function shall enable the CRC Peripheral Clock by calling 'RccAhb1PeriphClockCmd' with parameters M\_RCC\_AHB1PERIPH\_CRC and ENABLE.

##### 10.6.1.7.26 daucmfhw-HwInit-LLR-026

Requirement ID: H698-LLD-CMU-FNC-656

The function shall initialize the FPU context save not in lazy mode (i.e M\_FPU\_CONTEXT\_NOT\_IN\_LAZYMODE) in fpccr (Floating-point context control register) of M\_FPU.

##### 10.6.1.7.27 daucmfhw-HwInit-LLR-027

Requirement ID: H698-LLD-CMU-FNC-657

The function shall enable full access privileges for coprocessors (i.e M\_ENABLE\_FP\_CPACR) in cpacr (Coprocessor access control register) of M\_SCB.

##### 10.6.1.7.28 daucmfhw-HwInit-LLR-028

Requirement ID: H698-LLD-CMU-FNC-658

The function shall enable Memory management Fault,Bus Fault,and Usage Fault exceptions by setting the bits 16,17 and 18 to one of System Handler Control and State Register(shcsr) of System Control Block(SCB) (i.e M\_SCB\_SHCSR\_USGFAULTENA\_MSK bitwise OR M\_SCB\_SHCSR\_BUSFAULTENA\_MSK bitwise OR M\_SCB\_SHCSR\_MEMFAULTENA\_MSK)

##### 10.6.1.7.29 daucmfhw-HwInit-LLR-029

Requirement ID: H698-LLD-CMU-FNC-659

The function shall enable Usage Fault when the processor executes DIV(division) instruction with a divisor of 0 (M\_DIVIDE\_ZERO) by setting bit 4 of Configuration Control Register (ccr) of System Control Block(SCB).

##### 10.6.1.7.30 daucmfhw-HwInit-LLR-030

Requirement ID: H698-LLD-CMU-FNC-660

The function shall enable GPIOC Peripheral Clock by calling ‘RccAhb1PeriphClockCmd’ function with parameters M\_RCC\_AHB1PERIPH\_GPIOC and ENABLE.

##### 10.6.1.7.31 daucmfhw-HwInit-LLR-034

Requirement ID: H698-LLD-CMU-FNC-664

The function shall

- set GPIO pins of GPIO Init structure to be configured to (M\_HW\_LED\_GREEN bitwise OR M\_HW\_LED\_RED)

- set speed for the selected pins of GPIO Init structure to GPIO\_SPEED\_50MHZ

- set operating mode for the selected pins of GPIO Init structure to GPIO\_MODE\_OUT

- set operating output type for the selected pins of GPIO Init structure to GPIO\_OTYPE\_PP

- set operating Pull-up/Pull down for the selected pins of GPIO Init structure to GPIO\_PUPD\_DOWN

- initializes the GPIOC Peripheral by calling function GpioInit with parameters M\_GPIOC and reference to

GPIO Init structure

- Turn on the green LED by calling GpioSetBits function with parameters M\_GPIOC and

M\_HW\_LED\_GREEN.

### 10.6.2 HwCopy

Low Level Design Details about CSU HwCopy will follow in the sub sections.

#### 10.6.2.1 Brief Description

HwCopy function copies data from source buffer to destination buffer.

#### 10.6.2.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-012-CMU).

#### 10.6.2.3 List of global variables accessed and modified

Accessed : None

Modified : None

#### 10.6.2.4 Parameter list (Input/Output)

Inputs: const T\_UINT8 \* p\_src - pointer to source buffer

T\_UINT16 count - number of bytes to be copied

Outputs: T\_UINT8 \* p\_dest - pointer to destination buffer

#### 10.6.2.5 Return Value

None

#### 10.6.2.6 Other CSUs called by this CSU

None

#### 10.6.2.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to HwCopy.

##### 10.6.2.7.1 daucmfhw-HwCopy-LLR-001

Requirement ID: H698-LLD-CMU-DRQ-673

The function shall loop through M\_ZERO to count minus 1 to copy the data from p\_src buffer into p\_dest buffer.

## 10.7 daucmfinit

daucmfinit CSC initializes all the tasks for the OS to handle.

### 10.7.1 InitTask

Low Level Design Details about CSU InitTask will follow in the sub sections.

#### 10.7.1.1 Brief Description

InitTask function is the task that will never exit, used by the OS to create the thread/task for this module.

#### 10.7.1.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-012-CMU).

#### 10.7.1.3 List of global variables accessed and modified

Accessed : None

Modified : None

#### 10.7.1.4 Parameter list (Input/Output)

Inputs: void \* p\_data - Kill compiler warning

Outputs: void \* p\_data - Kill compiler warning

#### 10.7.1.5 Return Value

None

#### 10.7.1.6 Other CSUs called by this CSU

WdogInit

TbaseInit

WdogKickWatchDog

XramInit

A825CommInit

A825Init

RTCHWInit

ContinuousBitTaskInit

HeartBeatInit

OsSemPend

GpioToggleBits

Si2cInit

#### 10.7.1.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to InitTask.

##### 10.7.1.7.1 daucmfinit-InitTask-LLR-001

Requirement ID: H698-LLD-CMU-FNC-683

The function shall do the following:

1. Documents an unused parameter and eliminates the related compiler error using macro M\_UNUSED\_PARAM with parameter p\_data.
2. Initializes the independent watchdog by calling ‘WdogInit’.

##### 10.7.1.7.2 daucmfinit-InitTask-LLR-002

Requirement ID: H698-LLD-CMU-FNC-684

The function shall

1. Initialize the time base of the system by calling ‘TbaseInit’.
2. Call 'WdogKickWatchDog' to reload watchdog.

##### 10.7.1.7.3 daucmfinit-InitTask-LLR-003

Requirement ID: H698-LLD-CMU-FNC-685

The function shall

1. Initialize the external NVRAM and NOR Flash by calling ‘XramInit’.
2. Call 'WdogKickWatchDog' to reload watchdog.

##### 10.7.1.7.4 daucmfinit-InitTask-LLR-004

Requirement ID: H698-LLD-CMU-FNC-686

The function shall

1. Initialize the task to handle commands coming across the A825 bus (CAN bus #1) by calling ‘A825CommInit’.
2. Call 'WdogKickWatchDog' to reload watchdog.

##### 10.7.1.7.5 daucmfinit-InitTask-LLR-005

Requirement ID: H698-LLD-CMU-FNC-687

The function shall

1. Initialize the A825 Bus (CAN bus #1) by calling ‘A825Init’.
2. Call 'WdogKickWatchDog' to reload watchdog.
3. Call the function 'Si2cInit' to initialize SI2C hardware and interrupts
4. Call 'WdogKickWatchDog' to reload watchdog.
5. Call 'RTCHWInit' with parameter Sem\_init\_task to check pbit battery low status and oscillator interrupted status.
6. Call 'WdogKickWatchDog' to reload watchdog.

##### 10.7.1.7.6 daucmfinit-InitTask-LLR-011

Requirement ID: H698-LLD-CMU-FNC-693

The function shall

1. Call 'ContinuousBitTaskInit' to initialize Continuous Built-In-Test Task.
2. Call 'WdogKickWatchDog' to reload watchdog.

##### 10.7.1.7.7 daucmfinit-InitTask-LLR-012

Requirement ID: H698-LLD-CMU-FNC-694

The function shall

1. Enable GPIOB Clock and configures LED and chip select by calling ‘HeartBeatInit’.
2. Call 'WdogKickWatchDog' to reload watchdog.

##### 10.7.1.7.8 daucmfinit-InitTask-LLR-013

Requirement ID: H698-LLD-CMU-FNC-695

The function shall loop infinitely and perform the following operations within loop

a) Pend on Semaphore by calling OsSemPend with the following parameters

- Allocated Semaphore for the task sem init task

- pend timeout value as M\_ZERO

- Address of local error code.

b) Call WdogKickWatchDog to reload watchdog counter.

c) Call GpioToggleBits(M\_HW\_LED\_HB\_TOGGLE) with reference to GPIO port B (M\_GPIOB) and M\_HW\_LED\_HB (GPIO pin 2 (M\_GPIO\_PIN\_2)) as parameters to toggle the heartbeat LED.

### 10.7.2 HeartBeatInit

Low Level Design Details about CSU HeartBeatInit will follow in the sub sections.

#### 10.7.2.1 Brief Description

The function HeartBeatInit enablee GPIOB Clock and configures LED and chip select.

#### 10.7.2.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-012-CMU).

#### 10.7.2.3 List of global variables accessed and modified

Accessed : None

Modified : None

#### 10.7.2.4 Parameter list (Input/Output)

Inputs: None

Outputs: None

#### 10.7.2.5 Return Value

None

#### 10.7.2.6 Other CSUs called by this CSU

RccAhb1PeriphClockCmd

GpioInit

#### 10.7.2.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to HeartBeatInit.

##### 10.7.2.7.1 daucmfinit-HeartBeatInit-LLR-001

Requirement ID: H698-LLD-CMU-FNC-704

The function shall

- enable GPIOB Clock by calling ‘RccAhb1PeriphClockCmd’ function with parameters

M\_RCC\_AHB1PERIPH\_GPIOB and ENABLE.

- set GPIO pins to be configured of the GPIO Init structure to M\_HW\_LED\_HB.

- set speed for the selected pins of the GPIO Init structure to GPIO\_SPEED\_50MHZ.

- set operating mode for the selected pins of the GPIO Init structure to GPIO\_MODE\_OUT.

- set operating output type for the selected pins of the GPIO Init structure to GPIO\_OTYPE\_PP.

- set operating Pull-up/Pull down for the selected pins of the GPIO Init structure to

GPIO\_PUPD\_NOPULL.

- initialize GPIOB Peripheral by calling function ‘GpioInit’ with parameters M\_GPIOB and reference to GPIO Init structure.

### 10.7.3 InitInit

Low Level Design Details about CSU InitInit will follow in the sub sections.

#### 10.7.3.1 Brief Description

InitInit function installs Semaphore into timebase and creates init OS task.

#### 10.7.3.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-012-CMU).

#### 10.7.3.3 List of global variables accessed and modified

Accessed : Init\_task\_stk

Modified : None

#### 10.7.3.4 Parameter list (Input/Output)

Inputs: None

Outputs: None

#### 10.7.3.5 Return Value

None

#### 10.7.3.6 Other CSUs called by this CSU

TbaseTaskSignaling

OsTaskCreate

OsSemCreate

RterrDisplayErrorId

RterrForever

InitTask

#### 10.7.3.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to InitInit.

##### 10.7.3.7.1 daucmfinit-InitInit-LLR-001

Requirement ID: H698-LLD-CMU-FNC-713

The function shall

a) Install task signalling parameters for the init task by calling 'TbaseTaskSignaling' with parameters

i) M\_INIT\_TASK\_TICKS as task ticks and

ii) Return value of function OsSemCreate with parameter M\_ZERO stored in Semaphore to signal

Init task.

b) call the function RterrDisplayErrorId with parameter (M\_APP\_ID, RterrForever function,

OS\_KERNEL\_ERR bitwise OR with RT\_ERROR\_6)

When Semaphore to signal Init task is M\_NULL, otherwise do nothing.

##### 10.7.3.7.2 daucmfinit-InitInit-LLR-002

Requirement ID: H698-LLD-CMU-FNC-714

The function shall create the Init Task as the first task to run of RTOS by calling 'OsTaskCreate' with parameters pointer to function InitTask, M\_HW\_NULL as task entry point, Reference to top of stack of Init task stack(Init\_task\_stk) with index [M\_INIT\_TASK\_STK\_SIZE], M\_INIT\_TASK\_PRIO as task priority and calls the function RterrDisplayErrorId with parameter (M\_APP\_ID, RterrForever, OS\_KERNEL\_ERR bitwise OR with RT\_ERROR\_7) when the function OsTaskCreate returns other than M\_OS\_NO\_ERR, otherwise do nothing.

## 10.8 daucmfintr

daucmfintr CSC initializes the Vector Table and allows installing and uninstalling interrupts into the table.

### 10.8.1 IntrInit

Low Level Design Details about CSU IntrInit will follow in the sub sections.

#### 10.8.1.1 Brief Description

IntrInit function copies the ISR functions from ROM to RAM.

#### 10.8.1.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-012-CMU).

#### 10.8.1.3 List of global variables accessed and modified

Accessed : None

Modified : None

#### 10.8.1.4 Parameter list (Input/Output)

Inputs: None

Outputs: None

#### 10.8.1.5 Return Value

None

#### 10.8.1.6 Other CSUs called by this CSU

SpuriousInterrupt

#### 10.8.1.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to IntrInit.

##### 10.8.1.7.1 daucmfintr-IntrInit-LLR-001

Requirement ID: H698-LLD-CMU-FNC-724

The function shall store the interrupt functions of particular index from the interrupt Vector Table in ROM into the corresponding index of the interrupt Vector Table in RAM, by looping through INTR\_RESET to INTR\_MAX and the interrupt vector table in ROM for corresponding index is not empty (i.e. not equal to M\_ZERO).

##### 10.8.1.7.2 daucmfintr-IntrInit-LLR-002

Requirement ID: H698-LLD-CMU-FNC-725

The function shall store the reference of the function ‘SpuriousInterrupt’ into the corresponding index of the interrupt Vector Table in RAM, by looping through INTR\_RESET to INTR\_MAX and the interrupt Vector Table in ROM for corresponding index is empty (i.e. equal to M\_ZERO).

### 10.8.2 IntrInstall

Low Level Design Details about CSU IntrInstall will follow in the sub sections.

#### 10.8.2.1 Brief Description

The function IntrInstall loads the ISR into the Vector Table for the received interrupt index.

#### 10.8.2.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-012-CMU).

#### 10.8.2.3 List of global variables accessed and modified

Accessed : U32\_critical\_sr

Modified : U32\_critical\_sr

#### 10.8.2.4 Parameter list (Input/Output)

Inputs: T\_UINT8 u8\_index - interrupt number index in the Vector Table

T\_INTR\_FN intr\_function - ISR function

Outputs: None

#### 10.8.2.5 Return Value

None

#### 10.8.2.6 Other CSUs called by this CSU

SaveStatusReg

RestoreStatusReg

#### 10.8.2.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to IntrInstall.

##### 10.8.2.7.1 daucmfintr-IntrInstall-LLR-001

Requirement ID: H698-LLD-CMU-FNC-734

The function shall set U32\_critical\_sr to return value of ‘SaveStatusReg’ (M\_OS\_ENTER\_CRITICAL) to disable interrupts before loading the ISR into the interrupt Vector Table.

##### 10.8.2.7.2 daucmfintr-IntrInstall-LLR-002

Requirement ID: H698-LLD-CMU-FNC-735

The function shall store the intr\_function into index u8\_index of the interrupt Vector Table in RAM when u8\_index is less than INTR\_MAX, otherwise do nothing.

##### 10.8.2.7.3 daucmfintr-IntrInstall-LLR-003

Requirement ID: H698-LLD-CMU-FNC-736

The function shall Call RestoreStatusReg (M\_OS\_EXIT\_CRITICAL) with parameter U32\_critical\_sr

to restore the priority mask register after the loading is complete.

### 10.8.3 ResetIsr

Low Level Design Details about CSU ResetIsr will follow in the sub sections.

#### 10.8.3.1 Brief Description

ResetIsr function is the reset routine on Power On.

#### 10.8.3.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-012-CMU).

#### 10.8.3.3 List of global variables accessed and modified

Accessed : None

Modified : None

#### 10.8.3.4 Parameter list (Input/Output)

Inputs: None

Outputs: None

#### 10.8.3.5 Return Value

None

#### 10.8.3.6 Other CSUs called by this CSU

Crt0TransferData

#### 10.8.3.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to ResetIsr.

##### 10.8.3.7.1 daucmfintr-ResetIsr-LLR-001

Requirement ID: H698-LLD-CMU-FNC-745

The function shall call Crt0TransferData to run the reset routine.

### 10.8.4 NonMaskable

Low Level Design Details about CSU NonMaskable will follow in the sub sections.

#### 10.8.4.1 Brief Description

NonMaskable function is an interrupt error handler for nonMaskable errors.

#### 10.8.4.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-012-CMU).

#### 10.8.4.3 List of global variables accessed and modified

Accessed : None

Modified : None

#### 10.8.4.4 Parameter list (Input/Output)

Inputs: None

Outputs: None

#### 10.8.4.5 Return Value

None

#### 10.8.4.6 Other CSUs called by this CSU

RterrDisplayErrorId

RterrForever

#### 10.8.4.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to NonMaskable.

##### 10.8.4.7.1 daucmfintr-NonMaskable-LLR-001

Requirement ID: H698-LLD-CMU-FNC-754

The function shall display the interrupt error handler for NonMaskable errors by calling function RterrDisplayErrorId with parameters M\_APP\_ID, RterrForever function and INTR\_ERR bitwise OR with RT\_ERROR\_1.

### 10.8.5 HardFault

Low Level Design Details about CSU HardFault will follow in the sub sections.

#### 10.8.5.1 Brief Description

HardFault is an interrupt error handler for HardFault errors.

#### 10.8.5.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-012-CMU).

#### 10.8.5.3 List of global variables accessed and modified

Accessed : None

Modified : None

#### 10.8.5.4 Parameter list (Input/Output)

Inputs: None

Outputs: None

#### 10.8.5.5 Return Value

None

#### 10.8.5.6 Other CSUs called by this CSU

RterrDisplayErrorId

RterrForever

#### 10.8.5.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to HardFault.

##### 10.8.5.7.1 daucmfintr-HardFault-LLR-001

Requirement ID: H698-LLD-CMU-FNC-763

The function shall display the interrupt error handler for HardFault errors by calling function RterrDisplayErrorId with parameters M\_APP\_ID,

RterrForever function and INTR\_ERR bitwise OR with RT\_ERROR\_2.

### 10.8.6 MemManage

Low Level Design Details about CSU MemManage will follow in the sub sections.

#### 10.8.6.1 Brief Description

MemManage function is an interrupt error handler for Memory Management errors.

#### 10.8.6.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-012-CMU).

#### 10.8.6.3 List of global variables accessed and modified

Accessed : None

Modified : None

#### 10.8.6.4 Parameter list (Input/Output)

Inputs: None

Outputs: None

#### 10.8.6.5 Return Value

None

#### 10.8.6.6 Other CSUs called by this CSU

RterrDisplayErrorId

RterrForever

#### 10.8.6.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to MemManage.

##### 10.8.6.7.1 daucmfintr-MemManage-LLR-001

Requirement ID: H698-LLD-CMU-FNC-772

The function shall display the interrupt error handler for memory management errors

by calling function RterrDisplayErrorId with parameters M\_APP\_ID, RterrForever function and INTR\_ERR bitwise OR with RT\_ERROR\_3.

### 10.8.7 BusFault

Low Level Design Details about CSU BusFault will follow in the sub sections.

#### 10.8.7.1 Brief Description

BusFault function is an interrupt error handler for bus fault errors.

#### 10.8.7.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-012-CMU).

#### 10.8.7.3 List of global variables accessed and modified

Accessed : None

Modified : None

#### 10.8.7.4 Parameter list (Input/Output)

Inputs: None

Outputs: None

#### 10.8.7.5 Return Value

None

#### 10.8.7.6 12.14.7.6 Other CSUs called by this CSU

RterrDisplayErrorId

RterrForever

#### 10.8.7.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to BusFault.

##### 10.8.7.7.1 daucmfintr-BusFault-LLR-001

Requirement ID: H698-LLD-CMU-FNC-781

The function shall display the interrupt error handler for bus fault errors by calling function RterrDisplayErrorId with parameters M\_APP\_ID,

RterrForever function and INTR\_ERR bitwise OR with RT\_ERROR\_4.

### 10.8.8 UsageFault

Low Level Design Details about CSU UsageFault will follow in the sub sections.

#### 10.8.8.1 Brief Description

UsageFault function is an interrupt error handler for usage fault errors.

#### 10.8.8.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-012-CMU).

#### 10.8.8.3 List of global variables accessed and modified

Accessed : None

Modified : None

#### 10.8.8.4 Parameter list (Input/Output)

Inputs: None

Outputs: None

#### 10.8.8.5 Return Value

None

#### 10.8.8.6 Other CSUs called by this CSU

RterrDisplayErrorId

RterrForever

#### 10.8.8.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to UsageFault.

##### 10.8.8.7.1 daucmfintr-UsageFault-LLR-001

Requirement ID: H698-LLD-CMU-FNC-790

The function shall display the interrupt error handler for usage fault errors by calling function RterrDisplayErrorId with parameters M\_APP\_ID, RterrForever function and INTR\_ERR bitwise OR with RT\_ERROR\_5.

### 10.8.9 SpuriousInterrupt

Low Level Design Details about CSU SpuriousInterrupt will follow in the sub sections.

#### 10.8.9.1 Brief Description

SpuriousInterrupt function is an interrupt error handler for spurious interrupts.

#### 10.8.9.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-012-CMU).

#### 10.8.9.3 List of global variables accessed and modified

Accessed : None

Modified : None

#### 10.8.9.4 Parameter list (Input/Output)

Inputs: None

Outputs: None

#### 10.8.9.5 Return Value

None

#### 10.8.9.6 Other CSUs called by this CSU

RterrDisplayErrorId

RterrForever

#### 10.8.9.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to SpuriousInterrupt.

##### 10.8.9.7.1 daucmfintr-SpuriousInterrupt-LLR-001

Requirement ID: H698-LLD-CMU-FNC-799

The function shall display the interrupt error handler for spurious interrupts by calling function RterrDisplayErrorId with parameters M\_APP\_ID,

RterrForever function and INTR\_ERR bitwise OR with RT\_ERROR\_6.

## 10.9 daucmfmain

daucmfmain CSC initializes the HW, CRC sections, Kernel, the tasks and performs Power Up Built-In-Test check.

### 10.9.1 MainFunc

Low Level Design Details about CSU MainFunc will follow in the sub sections.

#### 10.9.1.1 Brief Description

MainFunc function initializes the HW, CRC sections, Kernel, the tasks and performs Power Up Built-In-Test check.

#### 10.9.1.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-012-CMU).

#### 10.9.1.3 List of global variables accessed and modified

Accessed : None

Modified : None

#### 10.9.1.4 Parameter list (Input/Output)

Inputs: None

Outputs: None

#### 10.9.1.5 Return Value

T\_UINT16 - returns zero

#### 10.9.1.6 Other CSUs called by this CSU

HwInit

PbitCheck

OsInit

InitInit

OsStart

#### 10.9.1.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to MainFunc.

##### 10.9.1.7.1 daucmfmain-MainFunc-LLR-001

Requirement ID: H698-LLD-CMU-FNC-809

The function shall initialize the Microcontroller Hardware by calling ‘HwInit’ function.

##### 10.9.1.7.2 daucmfmain-MainFunc-LLR-002

Requirement ID: H698-LLD-CMU-FNC-810

The function shall perform Power Up Built-In-Test check by calling the function ‘PbitCheck’.

##### 10.9.1.7.3 daucmfmain-MainFunc-LLR-003

Requirement ID: H698-LLD-CMU-FNC-811

The function shall initialize the Kernel by calling function ‘OsInit’.

##### 10.9.1.7.4 daucmfmain-MainFunc-LLR-004

Requirement ID: H698-LLD-CMU-FNC-812

The function shall initialize the Init task by calling function ‘InitInit’.

##### 10.9.1.7.5 daucmfmain-MainFunc-LLR-005

Requirement ID: H698-LLD-CMU-FNC-813

The function shall start the Kernel by calling function ‘OsStart’.

##### 10.9.1.7.6 daucmfmain-MainFunc-LLR-006

Requirement ID: H698-LLD-CMU-FNC-814

The function shall return M\_ZERO.

## 10.10 daucmfoscpu

daucmfoscpu module defines the OS Task creation function.

### 10.10.1 OsTaskCreate

Low Level Design Details about CSU OsTaskCreate will follow in the sub sections.

#### 10.10.1.1 Brief Description

The function OsTaskCreate creates a task to be managed by µC/OS. Tasks can either be created prior to the start of multitasking or by a running task. A task cannot be created by an ISR.

#### 10.10.1.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD (H698-003-012-CMU).

#### 10.10.1.3 List of global variables accessed and modified

Accessed : Os\_tcb\_prio\_tbl

Os\_running

U32\_critical\_sr

Modified : U32\_critical\_sr

#### 10.10.1.4 Parameter list (Input/Output)

Inputs: void (\*ptask)(void \*dptr) - function pointer to the task

void \*pdata – pointer to task parameters

T\_UINT8 u8\_prio - task priority

void \*pt\_os – pointer to top of stack

Outputs: None

#### 10.10.1.5 Return Value

T\_UINT8 - returns task creation status

M\_OS\_NO\_ERR - Task created successfully

M\_OS\_PRIO\_EXIST - Task already exists

M\_OS\_PRIO\_INVALID - priority is higher than the maximum allowed

#### 10.10.1.6 Other CSUs called by this CSU

OsSched

OsTcbInit

RterrDisplayErrorId

SaveStatusReg

RestoreStatusReg

RterrForever

#### 10.10.1.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to OsTaskCreate.

##### 10.10.1.7.1 daucmfoscpu-OsTaskCreate-LLR-001

Requirement ID: H698-LLD-CMU-FNC-824

The function shall return M\_OS\_PRIO\_INVALID if u8\_prio is higher than the maximum allowed(M\_OS\_LOWEST\_PRIO) otherwise do nothing.

##### 10.10.1.7.2 daucmfoscpu-OsTaskCreate-LLR-002

Requirement ID: H698-LLD-CMU-FNC-825

The function shall set U32\_critical\_sr to return value of ‘SaveStatusReg’ (M\_OS\_ENTER\_CRITICAL) to disable interrupts.

##### 10.10.1.7.3 daucmfoscpu-OsTaskCreate-LLR-003

Requirement ID: H698-LLD-CMU-FNC-826

The function shall perform the following operations when the requested task priority 'u8\_prio' is not available in the TCB priority table 'Os\_tcb\_prio\_tbl' i.e Os\_tcb\_prio\_tbl with index u8\_prio is equal to M\_ZERO.

a) Enable the interrupt by calling RestoreStatusReg with parameter U32\_critical\_sr

b) Set the stack pointer with pt\_os.

c) Set (Stack top minus 1) with M\_FPCCR

d) Set (Stack top minus 2) with M\_FPSCR

e) Update the stack pointer by subtracting it with M\_SIXTEEN

f) Set (Stack top minus 19) with M\_XPSR

g) Set (Stack top minus 20) with the address of task code 'ptask'

h) Set (Stack top minus 21) with Link Register M\_R14\_LR

i) Update the stack pointer by subtracting it with M\_FOUR

j) Set (Stack top minus 26) with task parameters 'pdata'

k) Decrement the stack pointer by M\_TWENTY\_FOUR to save the remaining registers on process stack.

l) call OsTcbInit with the parameters u8\_prio and stack pointer Stack top to initialize the TCB.

m) call function OsSched to find highest priority task when the call to function OsTcbInit returned M\_OS\_NO\_ERR and flag indicating that kernel is running (Os\_running) is TRUE.

n) Do nothing when the call to function OsTcbInit returned M\_OS\_NO\_ERR and flag indicating that kernel is running (Os\_running) is not TRUE.

o) Return error when the call to function OsTcbInit returns an error.

p) Do nothing if the return value of the called function OsTcbInit is not equal to M\_OS\_NO\_ERR.

##### 10.10.1.7.4 daucmfoscpu-OsTaskCreate-LLR-004

Requirement ID: H698-LLD-CMU-FNC-827

The function shall perform the following when the task to be created already exists i.e Os\_tcb\_prio\_tbl with index u8\_prio is not equal to M\_ZERO:

a)enable the interrupt by calling RestoreStatusReg with parameter U32\_critical\_sr using M\_OS\_EXIT\_CRITICAL.

b)display the input error by calling function RterrDisplayErrorId with parameters M\_APP\_ID, RterrForever function, OS\_KERNEL\_ERR bitwise OR with RT\_ERROR\_8.

c)Return M\_OS\_PRIO\_EXIST

## 10.11 daucmfoscpua

daucmfoscpua module contains Implementation of asm routines for uCOS.

### 10.11.1 12.17.1 OsCtxSw

Low Level Design Details about CSU OsCtxSw will follow in the sub sections.

#### 10.11.1.1 Brief Description

OsCtxSw function performs a task level context switch.

NOTE: This function triggers the PendSV exception (PendSV is an interrupt-driven request for system-level service. Use PendSV for context switching when no other exception is active, PendSV handler will run when interrupts are re-enabled, after call to the M\_OS\_EXIT\_CRITICAL ()).

#### 10.11.1.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD (H698-003-012-CMU).

#### 10.11.1.3 List of global variables accessed and modified

Accessed : None

Modified : None

#### 10.11.1.4 Parameter list (Input/Output)

Inputs : None

Outputs : None

#### 10.11.1.5 Return Value

None

#### 10.11.1.6 Other CSUs called by this CSU

None

#### 10.11.1.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to OsCtxSw.

##### 10.11.1.7.1 daucmfoscpua-OsCtxSw-LLR-001

Requirement ID: H698-LLD-CMU-FNC-837

The function shall do the following to trigger the PendSV exception

- Load the Interrupt control state register (NVIC\_INT\_CTRL) to R0

- Load the Value of trigger PendSV isr (NVIC\_PENDSVSET) to R1

- write R1 to address R0 content.

### 10.11.2 OsIntCtxSw

Low Level Design Details about CSU OsIntCtxSw will follow in the sub sections.

#### 10.11.2.1 Brief Description

This function performs a task level context switch same as OSCtxSw.

Note: OSIntCtxSw() is called by OSIntExit() when it determines a context switch is needed as the result of an interrupt. This function simply triggers a PendSV exception which will be handled when there are no more interrupts active and interrupts are enabled.

#### 10.11.2.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD (H698-003-012-CMU).

#### 10.11.2.3 List of global variables accessed and modified

Accessed : None

Modified : None

#### 10.11.2.4 Parameter list (Input/Output)

Inputs : None

Outputs : None

#### 10.11.2.5 Return Value

None

#### 10.11.2.6 Other CSUs called by this CSU

None

#### 10.11.2.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to OsIntCtxSw.

##### 10.11.2.7.1 daucmfoscpua-OsIntCtxSw-LLR-001

Requirement ID: H698-LLD-CMU-FNC-846

The function shall do the following to trigger the PendSV exception

- Load the Interrupt control state register (NVIC\_INT\_CTRL) to R0

- Load the Value to trigger PendSV isr (NVIC\_PENDSVSET) to R1

- write R1 to address R0 content.

### 10.11.3 SaveStatusReg

Low Level Design Details about CSU SaveStatusReg will follow in the sub sections.

#### 10.11.3.1 Brief Description

SaveStatusReg function saves the interrupt mask register and then disables interrupts to implement OS\_CRITICAL\_METHOD.

This function is invoked by the M\_OS\_ENTER\_CRITICAL().

NOTE: When this function returns, R0 contains the state of the PRIMASK register which contains the global interrupt mask prior to disabling interrupts.

#### 10.11.3.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD (H698-003-012-CMU).

#### 10.11.3.3 List of global variables accessed and modified

Accessed : None

Modified : None

#### 10.11.3.4 Parameter list (Input/Output)

Inputs : None

Outputs : None

#### 10.11.3.5 Return Value

None

#### 10.11.3.6 Other CSUs called by this CSU

None

#### 10.11.3.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to SaveStatusReg.

##### 10.11.3.7.1 daucmfoscpua-SaveStatusReg-LLR-001

Requirement ID: H698-LLD-CMU-FNC-855

The function shall do the following to save the interrupt mask register and then disables interrupts.

- Read PRIMASK special register value and write it to R0

- Disable IRq by setting the PRIMASK special register value.

### 10.11.4 RestoreStatusReg

Low Level Design Details about CSU RestoreStatusReg will follow in the sub sections.

#### 10.11.4.1 Brief Description

RestoreStatusReg function restores the interrupt disable mask to its original value.

#### 10.11.4.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD (H698-003-012-CMU).

#### 10.11.4.3 List of global variables accessed and modified

Accessed : None

Modified : None

#### 10.11.4.4 Parameter list (Input/Output)

Inputs : None

Outputs : None

#### 10.11.4.5 Return Value

None

#### 10.11.4.6 Other CSUs called by this CSU

None

#### 10.11.4.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to RestoreStatusReg.

##### 10.11.4.7.1 daucmfoscpua-RestoreStatusReg-LLR-001

Requirement ID: H698-LLD-CMU-FNC-864

The function shall write the contents of R0 into PRIMASK register to enable interrupts(Restore saved interrupts).

### 10.11.5 OSStartHighRdy

Low Level Design Details about CSU OSStartHighRdy will follow in the sub sections.

#### 10.11.5.1 Brief Description

OsStartHighRdy function starts running the highest priority task.

NOTE: OsStartHighRdy() is called by OsStart(). OsStart() sets OSTCBHighRdy to point to the OS\_TCB of the highest priority task. The highest priority task should be created before call to OSStart.

#### 10.11.5.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD (H698-003-012-CMU).

#### 10.11.5.3 List of global variables accessed and modified

Accessed : Os\_running

Modified : None

#### 10.11.5.4 Parameter list (Input/Output)

Inputs : None

Outputs : None

#### 10.11.5.5 Return Value

None

#### 10.11.5.6 Other CSUs called by this CSU

None

#### 10.11.5.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to OSStartHighRdy.

##### 10.11.5.7.1 daucmfoscpua-OsStartHighRdy-LLR-001

Requirement ID: H698-LLD-CMU-FNC-873

The function shall do the following to set the PendSV exception priority

- Load the System prio register (NVIC\_SYSPRI14) to R0

- Load the PendSV priority value (NVIC\_PENDSV\_PRI) to R1

- Write R1 as unsigned byte to address R0 content.

##### 10.11.5.7.2 daucmfoscpua-OsStartHighRdy-LLR-002

Requirement ID: H698-LLD-CMU-FNC-874

The function shall do the following to set the PSP to 0 for initial context switch call

- Write value of M\_ZERO to R0

- Write the contents of R0 into PSP register.

##### 10.11.5.7.3 daucmfoscpua-OsStartHighRdy-LLR-003

Requirement ID: H698-LLD-CMU-FNC-875

The function shall do the following to set the Os\_running flag to TRUE

(Indicate that multitasking will start)

- Load the Os\_running flag address to R0

- Write value of M\_ONE to R1

- Write R1 as unsigned byte to address R0 content.

##### 10.11.5.7.4 daucmfoscpua-OsStartHighRdy-LLR-004

Requirement ID: H698-LLD-CMU-FNC-876

The function shall do the following to trigger the PendSV exception

- Load the Interrupt control state register (NVIC\_INT\_CTRL) to R0

- Load the Value to trigger PendSV isr (NVIC\_PENDSVSET) to R1

- write R1 to address R0 content.

##### 10.11.5.7.5 daucmfoscpua-OsStartHighRdy-LLR-005

Requirement ID: H698-LLD-CMU-FNC-877

The function shall do the following to enable interrupts

- Enable IRq by clearing the PRIMASK special register value.

##### 10.11.5.7.6 daucmfoscpua-OsStartHighRdy-LLR-006

Requirement ID: H698-LLD-CMU-FNC-878

The function shall enter infinite wait state when highest priority task could not be scheduled.

### 10.11.6 PendSvHandler

Low Level Design Details about CSU PendSvHandler will follow in the sub sections.

#### 10.11.6.1 Brief Description

PendSvHandler function handles all context switching for uCOS.

#### 10.11.6.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD (H698-003-012-CMU).

#### 10.11.6.3 List of global variables accessed and modified

Accessed : Os\_tcb\_cur

Os\_tcb\_high\_rdy

Modified : None

#### 10.11.6.4 Parameter list (Input/Output)

Inputs : None

Outputs : None

#### 10.11.6.5 Return Value

None

#### 10.11.6.6 Other CSUs called by this CSU

None

#### 10.11.6.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to PendSvHandler.

##### 10.11.6.7.1 daucmfoscpua-PendSvHandler-LLR-001

Requirement ID: H698-LLD-CMU-FNC-887

The function shall do the following to prevent interruption during context switch

- Disable IRq by setting the PRIMASK special register value.

##### 10.11.6.7.2 daucmfoscpua-PendSvHandler-LLR-002

Requirement ID: H698-LLD-CMU-FNC-888

The function shall do the following to save the process stack pointer (PSP)

- Read PSP register value and write it to R0.

##### 10.11.6.7.3 daucmfoscpua-PendSvHandler-LLR-003

Requirement ID: H698-LLD-CMU-FNC-889

The function shall do the following to skip register save the first time

- Forward branch to 'PendSvHandler\_NoSave' if R0 is zero

##### 10.11.6.7.4 daucmfoscpua-PendSvHandler-LLR-004

Requirement ID: H698-LLD-CMU-FNC-890

The function shall do the following to save remaining registers R4-R11 on process stack

- Decrement R0 Before each access and load the register R4-R11 to the Decremented address

##### 10.11.6.7.5 daucmfoscpua-PendSvHandler-LLR-005

Requirement ID: H698-LLD-CMU-FNC-891

The function shall do the following to save remaining floating point register S16-S31

- Decrement R0 Before each access and store the register S16-S31 to the Decremented address.

##### 10.11.6.7.6 daucmfoscpua-PendSvHandler-LLR-006

Requirement ID: H698-LLD-CMU-FNC-892

The function shall do the following to load the Stack pointer to the TCB stack

(i.e- os\_tcb\_stkptr of Os\_tcb\_cur = SP)

- Load the current TCB address (Os\_tcb\_cur) to R1

- Load the address content in R1 to R1

- write R0 to address R1 content.

##### 10.11.6.7.7 daucmfoscpua-PendSvHandler-LLR-007

Requirement ID: H698-LLD-CMU-FNC-893

The function shall do the following to set highest priority task ready to execute, into the current TCB when branch PendSvHandler\_NoSave is invoked

(i.e., Os\_tcb\_cur = Os\_tcb\_high\_rdy)

- Load the current TCB address (Os\_tcb\_cur) to R0

- Load the address of highest priority task ready to execute (Os\_tcb\_high\_rdy) to R1

- Load the address content in R1 to R2

- write R2 to address R0 content

- Load the address content in R2 to R0 (i.e - Set Stack Pointer (R0) =

os\_tcb\_stkptr of Os\_tcb\_cur)

- Increment R0 Before each access and restore the register S16- S31 from the Incremented

address

- Increment R0 after each access and load the register R4 - R11 from the Incremented address

- Load PSP with the new stack pointer (R0)

- Load the link register equal to link register bitwise or M\_FOUR

- Enable IRq by clear the PRIMASK special register value.

## 10.12 daucmfpbit

daucmfpbit module contains implementation of Power Up Built-In-Test functions.

### 10.12.1 ConfigDataCheckData

Low Level Design Details about CSU ConfigDataCheckData will follow in the sub sections.

#### 10.12.1.1 Brief Description

ConfigDataCheckData function checks the validity of the Configuration data.

#### 10.12.1.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD (H698-003-012-CMU).

#### 10.12.1.3 List of global variables accessed and modified

Accessed : U32\_checksum\_config

Modified : U32\_checksum\_config

#### 10.12.1.4 Parameter list (Input/Output)

Inputs: None

Outputs: None

#### 10.12.1.5 Return Value

None.

#### 10.12.1.6 Other CSUs called by this CSU

CrcResetDr

CrcCalcBlockCrc

BitErrorHandler

#### 10.12.1.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to ConfigDataCheckData.

##### 10.12.1.7.1 daucmfpbit-ConfigDataCheckData-LLR-001

Requirement ID: H698-LLD-CMU-FNC-903

The function shall reset the CRC Data register (DR) by calling CrcResetDr function before the Configuration data CRC check.

##### 10.12.1.7.2 daucmfpbit-ConfigDataCheckData-LLR-002

Requirement ID: H698-LLD-CMU-FNC-904

The function shall compute the 32-bit CRC by calling CrcCalcBlockCrc function with parameters

M\_MEMMAP\_DATA\_1\_ADDR and M\_MEMMAP\_DATA\_1\_CRC\_CNT, then stores the computed CRC in U32\_checksum\_config.

##### 10.12.1.7.3 daucmfpbit-ConfigDataCheckData-LLR-003

Requirement ID: H698-LLD-CMU-FNC-905

The function shall call the function BitErrorHandler with parameter CONFIG\_CRC\_ERR when the computed 32 bit CRC(U32\_checksum\_config) value is not equal to the local data crc2.

### 10.12.2 CpuTest

Low Level Design Details about CSU CpuTest will follow in the sub sections.

#### 10.12.2.1 Brief Description

CpuTest function performs the CPU test by performing the Arithmetic and logical operation check

#### 10.12.2.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD (H698-003-012-CMU).

#### 10.12.2.3 List of global variables accessed and modified

Accessed : Cpu\_test\_res

Modified : Cpu\_test\_res

#### 10.12.2.4 Parameter list (Input/Output)

Inputs: None

Outputs: None

#### 10.12.2.5 Return Value

None

#### 10.12.2.6 Other CSUs called by this CSU

AluTest

BitErrorHandler

#### 10.12.2.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to CpuTest.

##### 10.12.2.7.1 daucmfpbit-CpuTest-LLR-001

Requirement ID: H698-LLD-CMU-FNC-926

The function shall set the global variable Cpu\_test\_res to FAILED.

##### 10.12.2.7.2 daucmfpbit-CpuTest-LLR-002

Requirement ID: H698-LLD-CMU-FNC-927

The function shall call the function AluTest to perform the Arithmetic and logical operation check.

##### 10.12.2.7.3 daucmfpbit-CpuTest-LLR-003

Requirement ID: H698-LLD-CMU-FNC-928

The function shall call the function BitErrorHandler with parameter (CPU\_ERR) when the Cpu\_test\_res is equal to FAILED, otherwise do nothing.

### 10.12.3 BitErrorHandler

Low Level Design Details about CSU BitErrorHandler will follow in the sub sections.

#### 10.12.3.1 Brief Description

BitErrorHandler function performs BIT error handling.

#### 10.12.3.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD (H698-003-012-CMU).

#### 10.12.3.3 List of global variables accessed and modified

Accessed : None

Modified : None

#### 10.12.3.4 Parameter list (Input/Output)

Inputs: T\_ERR\_TYPE error\_code - Error number

CONFIG\_CRC\_ERR - if Configuration CRC check fails

CPU\_ERR - if CPU check fails

Outputs: None

#### 10.12.3.5 Return Value

None

#### 10.12.3.6 Other CSUs called by this CSU

RterrDisplayErrorId

RterrForever

#### 10.12.3.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to BitErrorHandler.

##### 10.12.3.7.1 daucmfpbit-BitErrorHandler-LLR-001

Requirement ID: H698-LLD-CMU-FNC-937

The function shall disable all interrupts.

##### 10.12.3.7.2 daucmfpbit-BitErrorHandler-LLR-002

Requirement ID: H698-LLD-CMU-FNC-938

The function shall call the function RterrDisplayErrorId with parameters as mentioned below when the error\_code received by the function is CRC\_ERR or CPU\_ERR, otherwise perform \_\_asm volatile of cpsie i.

Table 4: Error parameters

|  |  |
| --- | --- |
| **Error code** | **RterrDisplayErrorId Parameters** |
| CONFIG\_CRC\_ERR | M\_APP\_ID,RterrForever,CRC\_ERROR | RT\_ERROR\_1 |
| CPU\_ERR | M\_APP\_ID, RterrForever, CPU\_ERROR | RT\_ERROR\_2 |

### 10.12.4 PbitNvmReadWriteTest

Low Level Design Details about PbitNvmReadWriteTest will follow in the sub sections.

#### 10.12.4.1 Brief Description

This function performs NVM read/write test and sets the result in the Bit\_status structure.

#### 10.12.4.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD (H698-003-012-CMU).

#### 10.12.4.3 List of global variables accessed and modifed

Accessed: None

Modified: Bit\_status

#### 10.12.4.4 Parameters List (Input/Output

None

#### 10.12.4.5 Return value

None

#### 10.12.4.6 Others CSUs called by this CSU

OsTimeDly

#### 10.12.4.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to PbitNvmReadWriteTest.

##### 10.12.4.7.1 daucmfpbit-PbitNvmReadWriteTest-LLR-002

Requirement ID: H698-LLD-CMU-FNC-2325

The function shall do the following when loop through M\_ZERO to M\_STD\_NUMBER\_OF(au32testvalues) minus 1 AND NOT of btestfailed, pre-increment loop counter u32j

1. Set the dereference of pu32nvmdata to au32testvalues with index loop counter u32j.
2. Set btestfailed to dereference of pu32nvmdata not equal to au32testvalues with index loop counter u32j.

##### 10.12.4.7.2 daucmfpbit-PbitNvmReadWriteTest-LLR-003

Requirement ID: H698-LLD-CMU-FNC-2326

The function shall Set b\_nvm\_read\_write\_failed of bitwise\_status of power\_on of Bit\_status to btestfailed.

### 10.12.5 PbitCheck

Low Level Design Details about CSU PbitCheck will follow in the sub sections.

#### 10.12.5.1 Brief Description

PbitCheck function performs Power Up Built-In-Test check.

#### 10.12.5.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD (H698-003-012-CMU).

#### 10.12.5.3 List of global variables accessed and modified

Accessed : None

Modified : None

#### 10.12.5.4 Parameter list (Input/Output)

Inputs: None

Outputs: None

#### 10.12.5.5 Return Value

None

#### 10.12.5.6 Other CSUs called by this CSU

ConfigDataCheckData

CpuTest

GpioSetBits

GpioResetBits

#### 10.12.5.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to PbitCheck.

##### 10.12.5.7.1 daucmfpbit-PbitCheck-LLR-001

Requirement ID: H698-LLD-CMU-FNC-947

The function shall perform the heart beat function check by calling the following function using M\_LED\_RED:

- GpioSetBits with parameters M\_GPIOC and M\_HW\_LED\_RED

- GpioResetBits with parameters M\_GPIOC and M\_HW\_LED\_GREEN

##### 10.12.5.7.2 daucmfpbit-PbitCheck-LLR-002

Requirement ID: H698-LLD-CMU-FNC-948

The function shall call the function ConfigDataCheckData to perform the Configuration Data CRC check.

##### 10.12.5.7.3 daucmfpbit-PbitCheck-LLR-005

Requirement ID: H698-LLD-CMU-FNC-951

The function shall call the function CpuTest to perform the Cpu check.

## 10.13 daucmfrevno

daucmfrevno module contains the Revision number of the CMU Application Software.

It does not contain any functions.

### 10.13.1 Brief Description

The module daucmfrevno contains the Revision number of Flight Software Application

### 10.13.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD (H698-003-012-CMU).

### 10.13.3 List of global variables accessed and modified

Accessed : None

Modified : Revno\_id

### 10.13.4 Parameter list (Input/Output)

Inputs: None

Outputs: None

### 10.13.5 Return Value

None

### 10.13.6 Other CSUs called by this CSU

None

### 10.13.7 Description of list of LLRs allocated

daucmfrevno module does not provide any public operation as its functionality is to just provide the revision number.

Hence, the LLR name does not contain a CSU name.

#### 10.13.7.1 daucmfrevno-LLR-001

Requirement ID: H698-LLD-CMU-FNC-961

The CSC daucmfrevno shall set the Revno\_id of index M\_REVNO\_LEN (i.e CMU Application Software Part Number) to M\_REVNO\_REVISION

## 10.14 daucmfrtc

daucmfrtc CSC modifies and controls the RTC device.

### 10.14.1 BBatteryLevelIsGood

Low Level Design Details about CSU BbatteryLevelIsGood will follow in the sub sections.

#### 10.14.1.1 Brief Description

The function BbatteryLevelIsGood sends battery check messages.

#### 10.14.1.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-012-CMU).

#### 10.14.1.3 List of global variables accessed and modified

Accessed: None

Modified: None

#### 10.14.1.4 Parameter list (Input/Output)

Inputs: T\_OS\_EVENT \*pSemTask - Pointer to Semaphore for task.

Outputs: None

#### 10.14.1.5 Return Value

T\_BOOLEAN - returns the status of battery low flag

#### 10.14.1.6 Other CSUs called by this CSU

SI2CTransfer

#### 10.14.1.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to BBatteryLevelIsGood.

##### 10.14.1.7.1 daucmfrtc-BBatteryLevelIsGood-LLR-001

Requirement ID: H698-LLD-CMU-FNC-2089

The function shall do the following

* Set the u8 Bus Address of sinfo to BUS\_ADDRESS
* Set the u8 Data Address Size of sinfo to M\_ONE
* Set the u16 Data Size of sinfo to M\_ONE
* Set the pu8 Buffer of sinfo to reference of u8buffer
* Set the pu8 Status of sinfo to reference of b status
* Set the u8 One Byte of uData Address of sinfo to CNTRL\_3
* Call the function SI2Ctransfer with parameter ( reference of sinfo, M\_SI2C\_RX, reference of pSemTask)

##### 10.14.1.7.2 daucmfrtc-BBatteryLevelIsGood-LLR-002

Requirement ID: H698-LLD-CMU-FNC-2090

The function shall return the ( b status AND BATTERY\_GOOD is equal to (u8 buffer bitwise AND with BLF))

### 10.14.2 BIntegrityIsGood

Low Level Design Details about CSU BIntegrityIsGood will follow in the sub sections.

#### 10.14.2.1 Brief Description

The BIntegrityIsGood function checks the time messages.

#### 10.14.2.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-012-CMU).

#### 10.14.2.3 List of global variables accessed and modified

Accessed: None

Modified: None

#### 10.14.2.4 Parameter list (Input/Output)

Inputs: T\_OS\_EVENT \*pSemTask – IN semaphore task

Outputs: None

#### 10.14.2.5 Return Value

T\_BOOLEAN - returns the status of oscillator status

#### 10.14.2.6 Other CSUs called by this CSU

SI2CTransfer

#### 10.14.2.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to BIntegrityIsGood.

##### 10.14.2.7.1 daucmfrtc-BIntegrityIsGood-LLR-001

Requirement ID: H698-LLD-CMU-FNC-2099

The function shall do the following

* Set the u8 Bus Address of sinfo to BUS\_ADDRESS
* Set the u8 Data Address Size of sinfo to M\_ONE
* Set the u16 Data Size of sinfo to sizeof of a u8buffer
* Set pu8Buffer of sinfo to a u8buffer
* Set the pu8 Status of sinfo to reference of b status
* Set the u8 One Byte of uData Address of sinfo to SECONDS
* Call the function SI2Ctransfer with parameter ( reference of sinfo, M\_SI2C\_RX, reference of pSemTask)

##### 10.14.2.7.2 daucmfrtc-BIntegrityIsGood-LLR-002

Requirement ID: H698-LLD-CMU-FNC-2100

The function shall return the ( b status AND M\_ZERO is equal to (a8 buffer with index as M\_ZERO bitwise AND with OSC\_STABLE))

### 10.14.3 RTCHWInit

Low Level Design Details about CSU RTCHWInit will follow in the sub sections.

#### 10.14.3.1 Brief Description

The function RTCHWInit checks pbit battery low status and oscillator interrupted status

#### 10.14.3.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-012-CMU).

#### 10.14.3.3 List of global variables accessed and modified

Accessed: None

Modified: None

#### 10.14.3.4 Parameter list (Input/Output)

Inputs: T\_OS\_EVENT \*pSemTask - In semaphore task

Outputs: None.

#### 10.14.3.5 Return Value

None

#### 10.14.3.6 Other CSUs called by this CSU

BBatteryLevelIsGood  
BIntegrityIsGood

#### 10.14.3.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to RTCHWInit.

##### 10.14.3.7.1 daucmfrtc-RTCHWInit-LLR-001

Requirement ID: H698-LLD-CMU-FNC-2109

The function shall do the following

a)Set the dereference of pbit to address of power on of Bit\_status

b) Set the b\_rtc\_battery\_low of bitwise\_status of pbit to FALSE equal to the function BbatteryLevelIsGood with parameter pSemTask.

c) Set the b\_rtc\_oscillator\_interrupted of bitwise\_status of pbit to FALSE equal to the function BIntegrityIsGood with parameter pSemTask

## 10.15 daucmfsi2c

This module controls the SI2C bus connection

### 10.15.1 SI2CTransfer

Low Level Design Details about SI2CTransfer will follow in the sub sections.

#### 10.15.1.1 Brief Description

SI2CTransfer function transfers data across the SI2C bus.

#### 10.15.1.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-012-CMU).

#### 10.15.1.3 List of global variables accessed and modified

Accessed: None

Modified: None

#### 10.15.1.4 Parameter list (Input/Output)

Input: T\_SI2C\_INFO \*ps\_Si2c\_Info

T\_UINT8 u8\_Tx\_Rx

T\_OS\_EVENT \*\*pps\_Sem

Output: None

#### 10.15.1.5 Return Value

None

#### 10.15.1.6 Other CSUs called by this CSU

OsSemPend

I2cAcknowledgeConfig

I2cGenerateStart

#### 10.15.1.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to SI2CTransfer

##### 10.15.1.7.1 daucmfsi2c-SI2CTransfer-LLR-001

Requirement ID: H698-LLD-CMU-FNC-2119

The function shall loop infinitely and performs the following when the dereference of ps\_Si2c\_Info of pu8Status is not true AND pre-decrement of u8retrycount is greater than M\_ZERO.

* Call OsSemPend function with parameters Sem\_key\_i2c2, M\_ZERO, reference of u8 err to wait on a Semaphore

##### 10.15.1.7.2 daucmfsi2c-SI2CTransfer-LLR-002

Requirement ID: H698-LLD-CMU-FNC-2309

The function shall ,

* Set the dereference of ps\_Si2c\_Info to sSi2cInfo of Sdata\_blk
* Set the u8\_Tx\_Rx to u8TxRx of Sdata\_blk
* Set the pps\_Sem to ppsSemData of Sdata\_blk
* Set the M\_ZERO to u16DataCnt of Sdata\_blk

##### 10.15.1.7.3 daucmfsi2c-SI2CTransfer-LLR-003

Requirement ID: H698-LLD-CMU-FNC-2310

The function shall ,

* Set the SI2C\_BUS\_ADDR to eStage of Sdata\_blk
* Call I2cAcknowledgeConfig function with parameters M\_I2C2, ENABLE to enable I2C2 acknowledgement.
* Call I2cGenerateStart function with parameters M\_I2C2, ENABLE to send I2C2 START Condition.

##### 10.15.1.7.4 daucmfsi2c-SI2CTransfer-LLR-004

Requirement ID: H698-LLD-CMU-FNC-2311

The function shall Call OsSemPend function with parameters deference of ppssemdata of sdata blk, M\_ZERO, reference of u8 err to wait for the transfer to finish.

### 10.15.2 Si2cInit

Low Level Design Details about Si2cInit will follow in the sub sections.

#### 10.15.2.1 Brief Description

Si2cInit function initializes SI2C hardware and interrupts.

#### 10.15.2.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-012-CMU).

#### 10.15.2.3 List of global variables accessed and modified

Accessed: None

Modified: None

#### 10.15.2.4 Parameter list (Input/Output)

Input: None

Output: None

#### 10.15.2.5 Return Value

None

#### 10.15.2.6 Other CSUs called by this CSU

OsSemCreate

RccAhb1PeriphClockCmd

RccApb1PeriphClockCmd

IntrInstall

GpioPinAFConfig

GpioInit

I2cInit

NvicInit

I2cITConfig

I2cCmd  
Si2c2ErrIntr  
Si2c2Intr

#### 10.15.2.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to Si2cInit

##### 10.15.2.7.1 daucmfsi2c-Si2cInit-LLR-001

Requirement ID: H698-LLD-CMU-FNC-2128

The function shall do the following.  
1) Call the OsSemCreate function with parameters M\_ONE to initialize semaphore for accessing the I2C2 interface and store its return value in Sem\_key\_i2c2.

2)Call the RccAhb1PeriphClockCmd function with parameters M\_RCC\_AHB1PERIPH\_GPIOB, ENABLE to enable GPIOB clock.

3)Call the RccApb1PeriphClockCmd function with parameters M\_RCC\_APB1PERIPH\_I2C2, ENABLE to enable I2C2 clock.

##### 10.15.2.7.2 daucmfsi2c-Si2cInit-LLR-002

Requirement ID: H698-LLD-CMU-FNC-2129

The function shall do the following,

* Call the IntrInstall function with parameters INTR\_I2C\_2\_EV, Si2c2Intr to install I2C event interrupt into vector table in RAM.
* Call the IntrInstall function with parameters INTR\_I2C\_2\_ER, Si2c2ErrIntr to install I2C error interrupt into vector table in RAM.
* Call the GpioPinAFConfig function with parameters M\_GPIOB, M\_GPIO\_PINSOURCE10, M\_GPIO\_AF\_I2C2 for GPIOH configuration.
* Call the GpioPinAFConfig function with parameters M\_GPIOB, M\_GPIO\_PINSOURCE11, M\_GPIO\_AF\_I2C2 for GPIOH configuration.

##### 10.15.2.7.3 daucmfsi2c-Si2cInit-LLR-003

Requirement ID: H698-LLD-CMU-FNC-2130

The function shall do the following to configure PH4 and PH5 for EEPROM I2C interface.  
1)Set gpio\_pin of gpio init structure to M\_GPIOB\_I2C2\_SCL bitwise OR with M\_GPIOB\_I2C2\_SDA.  
2)Set gpio\_mode of gpio init structure to GPIO\_MODE\_AF.  
3)Set gpio\_speed of gpio\_init\_structure to GPIO\_SPEED\_100MHZ.  
4)Set gpio\_otype of gpio\_init\_structure to GPIO\_OTYPE\_OD.  
5)Set gpio\_pupd of gpio\_init\_structure to GPIO\_PUPD\_NOPULL.  
6)Call GpioInit function with parameters M\_GPIOB, reference to gpio init structure.

##### 10.15.2.7.4 daucmfsi2c-Si2cInit-LLR-004

Requirement ID: H698-LLD-CMU-FNC-2131

The function shall Call the I2cCmd function with parameters M\_I2C2, ENABLE to enable the I2C Peripheral.

The function shall do the following for I2C configuration.

* Set I2C\_Mode of i2c init structure to M\_I2C\_MODE\_I2C.
* Set I2C\_DutyCycle of i2c init structure to M\_I2C\_DUTYCYCLE\_2.
* Set I2C\_OwnAddress1 of i2c init structure to M\_TEN.
* Set I2C\_Ack of i2c init structure to M\_I2C\_ACK\_ENABLE.
* Set I2C\_AcknowledgedAddress of i2c init structure to M\_I2C\_ACKNOWLEDGEDADDRESS\_7BIT.
* Set I2C\_ClockSpeed of i2c init structure to M\_I2C\_CLOCKSPEED\_VALUE.
* Call the I2cInit function with parameters M\_I2C2, reference to i2c init structure.

##### 10.15.2.7.5 daucmfsi2c-Si2cInit-LLR-005

Requirement ID: H698-LLD-CMU-FNC-2312

The function shall do the following to configure and enable I2C2 event interrupt.

* Set nvic\_irq\_channel of nvic init structure to I2C2\_EV\_IRQN.
* Set nvic\_irq\_channel\_preemption\_priorityof nvic init structure to M\_SEVEN.
* Set nvic\_irq\_channel\_subpriority of nvic init structure to M\_ZERO.
* Set nvic\_irq\_channel\_cmd of nvic init structure to ENABLE.
* Call function NvicInit with parameter reference to nvic init structure.

The function shall do the following to configure and enable I2C2 error interrupt

* Set nvic\_irq\_channel of nvic init structure to I2C2\_ER\_IRQN
* Call function NvicInit with parameter reference to nvic init structure

The function shall call the I2cITConfig function with parameters M\_I2C2, M\_I2C\_IT\_EVT bitwise OR with M\_I2C\_IT\_BUF bitwise OR with M\_I2C\_IT\_ERR, ENABLE to enable I2C2 event, buffer and error interrupts.

### 10.15.3 TransferFinish

Low Level Design Details about TransferFinish will follow in the sub sections.

#### 10.15.3.1 Brief Description

TransferFinish function waits until the transfer is finished.

#### 10.15.3.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-012-CMU).

#### 10.15.3.3 List of global variables accessed and modified

Accessed: None

Modified: None

#### 10.15.3.4 Parameter list (Input/Output)

Input: T\_UINT8 u8\_Status

Output: None

#### 10.15.3.5 Return Value

None

#### 10.15.3.6 Other CSUs called by this CSU

OsSemPost

#### 10.15.3.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to TransferFinish

##### 10.15.3.7.1 daucmfsi2c-TransferFinish-LLR-001

Requirement ID: H698-LLD-CMU-FNC-2140

Set pu8Status of sSi2cInfo of dereference of data block to u8Status to save the transfer status.

Call the OsSemPost function with parameter ppsSemData of derefernce of data block.

Call the OsSemPost function with parameter Sem\_key\_i2c2.

### 10.15.4 Si2c2Intr

Low Level Design Details about Si2c2Intr will follow in the sub sections.

#### 10.15.4.1 Brief Description

Si2c2Intr function process the SI2C interrupt.

#### 10.15.4.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-012-CMU).

#### 10.15.4.3 List of global variables accessed and modified

Accessed: None

Modified: None

#### 10.15.4.4 Parameter list (Input/Output)

Input: None

Output: None

#### 10.15.4.5 Return Value

None

#### 10.15.4.6 Other CSUs called by this CSU

I2cSend7bitAddress

I2cGenerateStart

I2cSendData

I2cGenerateStop

TransferFinish

I2cAcknowledgeConfig

OsIntEnter

I2cReceiveData

I2cGetLastEvent

OsIntExit

#### 10.15.4.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to Si2c2Intr

##### 10.15.4.7.1 daucmfsi2c-Si2c2Intr-LLR-017

Requirement ID: H698-LLD-CMU-FNC-2308

The function shall call OsIntEnter function to tell uCOS that we are leaving the ISR.

##### 10.15.4.7.2 daucmfsi2c-Si2c2Intr-LLR-001

Requirement ID: H698-LLD-CMU-FNC-2149

The function shall call I2cSend7bitAddress with parameters M\_I2C2, u8BusAddr of sSi2cInfo of data block and M\_I2C\_DIRECTION\_TRANSMITTER to send slave address with write comment, when

* Return value of the function I2cGetLastEvent with parameter M\_I2C2 is M\_I2C\_EVENT\_MASTER\_MODE\_SELECT
* eStage of data block is equal to SI2C\_BUS\_ADDR.

##### 10.15.4.7.3 daucmfsi2c-Si2c2Intr-LLR-002

Requirement ID: H698-LLD-CMU-FNC-2150

The function shall do the following when return value of the function I2cGetLastEvent with parameter M\_I2C2 is M\_I2C\_EVENT\_MASTER\_MODE\_SELECT and eStage of data block is equal to SI2C\_BUS\_ADDR

1)Set estage of data block to SI2C\_DATA\_ADDR\_MSB when u8DataAddrSize of sSi2cInfo of data block is equal to M\_TWO otherwise set eStage of data block to SI2C\_DATA\_ADDR\_LSB.

##### 10.15.4.7.4 daucmfsi2c-Si2c2Intr-LLR-003

Requirement ID: H698-LLD-CMU-FNC-2151

The function shall call I2cSend7bitAddress with parameters M\_I2C2, u8BusAddr of sSi2cInfo of data block and M\_I2C\_DIRECTION\_RECEIVER to send slave address with read comment, when

* Return value of the function I2cGetLastEvent with parameter M\_I2C2 is M\_I2C\_EVENT\_MASTER\_MODE\_SELECT
* eStage of data block is equal to SI2C\_DATA\_RX.

##### 10.15.4.7.5 daucmfsi2c-Si2c2Intr-LLR-004

Requirement ID: H698-LLD-CMU-FNC-2152

The function shall not do anything while transmitting when return value of the function I2cGetLastEvent with parameter M\_I2C2 is M\_I2C\_EVT\_MAST\_BYTE\_TRANSMITTING.

##### 10.15.4.7.6 daucmfsi2c-Si2c2Intr-LLR-005

Requirement ID: H698-LLD-CMU-FNC-2153

The function shall call I2cSendData with parameters (M\_I2C2, u16TwoBytes of uDataAddr of sSi2cInfo of data block right shift by M\_EIGHT) to transmit MSB address and set eStage of data block to SI2C\_DATA\_ADDR\_LSB when following conditions achieves

* Return value of the function I2cGetLastEvent with parameter M\_I2C2 is M\_I2C\_EVT\_MASTER\_BYTE\_TRANSMITTED
* Return value of the function I2cGetLastEvent with parameter M\_I2C2 is M\_I2C\_EVT\_MASTER\_TRANS\_MODE\_SEL
* eStage of data block is equal to SI2C\_DATA\_ADDR\_MSB.

##### 10.15.4.7.7 daucmfsi2c-Si2c2Intr-LLR-006

Requirement ID: H698-LLD-CMU-FNC-2154

The function shall call I2cSendData with parameters (M\_I2C2, u8OneByte of uDataAddr of sSi2cInfo of data block) to transmit LSB address when following conditions achieves.

* Return value of the function I2cGetLastEvent with parameter M\_I2C2 is M\_I2C\_EVT\_MASTER\_BYTE\_TRANSMITTED
* Return value of the function I2cGetLastEvent with parameter M\_I2C2 is M\_I2C\_EVT\_MASTER\_TRANS\_MODE\_SEL
* eStage of data block is equal to SI2C\_DATA\_ADDR\_LSB.

##### 10.15.4.7.8 daucmfsi2c-Si2c2Intr-LLR-007

Requirement ID: H698-LLD-CMU-FNC-2155

The function shall set eStage of data block to SI2C\_RESTART when u8TxRx of data block is equal to M\_SI2C\_RX, otherwise set eStage of data block to SI2C\_DATA\_TX, it can be achieved when following conditions are satified.

* Return value of the function I2cGetLastEvent with parameter M\_I2C2 is M\_I2C\_EVT\_MASTER\_BYTE\_TRANSMITTED
* Return value of the function I2cGetLastEvent with parameter M\_I2C2 is M\_I2C\_EVT\_MASTER\_TRANS\_MODE\_SEL
* eStage of data block is equal to SI2C\_DATA\_ADDR\_LSB.

##### 10.15.4.7.9 daucmfsi2c-Si2c2Intr-LLR-008

Requirement ID: H698-LLD-CMU-FNC-2156

The function shall set eStage of data block to SI2C\_DATA\_RX and calls I2cGenerateStart with parameters M\_I2C2, ENABLE to generate restart condition when following conditions are achieved

* Return value of the function I2cGetLastEvent with parameter M\_I2C2 is M\_I2C\_EVT\_MASTER\_BYTE\_TRANSMITTED
* Return value of the function I2cGetLastEvent with parameter M\_I2C2 is M\_I2C\_EVT\_MASTER\_TRANS\_MODE\_SEL.
* eStage of data block is equal to SI2C\_RESTART.

##### 10.15.4.7.10 daucmfsi2c-Si2c2Intr-LLR-009

Requirement ID: H698-LLD-CMU-FNC-2157

The function shall call function I2cSendData with parameters (M\_I2C2, pu8Buffer with index (u16DataCnt of data block) of sSi2cInfo of data block) to Transmit data and increment u16DataCnt of data block by one when u16DataCnt of data block is less than u16DataSize of sSi2cInfo of data block, otherwise call I2cGenerateStop with parameter (M\_I2C2, ENABLE) to Generate stop condition, call TransferFinish with parameter(TRUE). it can be achieved when following conditions are satified.

* Return value of the function I2cGetLastEvent with parameter M\_I2C2 is M\_I2C\_EVT\_MASTER\_BYTE\_TRANSMITTED
* Return value of the function I2cGetLastEvent with parameter M\_I2C2 is M\_I2C\_EVT\_MASTER\_TRANS\_MODE\_SEL.
* eStage of data block is equal to SI2C\_DATA\_TX.

##### 10.15.4.7.11 daucmfsi2c-Si2c2Intr-LLR-010

Requirement ID: H698-LLD-CMU-FNC-2158

The function shall do nothing when

* Return value of the function I2cGetLastEvent with parameter M\_I2C2 is M\_I2C\_EVT\_MASTER\_BYTE\_TRANSMITTED
* Return value of the function I2cGetLastEvent with parameter M\_I2C2 is M\_I2C\_EVT\_MASTER\_TRANS\_MODE\_SEL.
* eStage of data block is other than SI2C\_DATA\_ADDR\_MSB, SI2C\_DATA\_ADDR\_LSB, SI2C\_RESTART, SI2C\_DATA\_TX.

##### 10.15.4.7.12 daucmfsi2c-Si2c2Intr-LLR-011

Requirement ID: H698-LLD-CMU-FNC-2159

The function shall call I2cAcknowledgeConfig with parameter (M\_I2C2, DISABLE) and I2cGenerateStop with parameter (M\_I2C2, ENABLE) when

* Return value of the function I2cGetLastEvent with parameter M\_I2C2 is M\_I2C\_EVENT\_MASTER\_REC\_MODE\_SEL
* u16DataCnt of data block added M\_ONE is equal to u16DataSize of sSi2cInfo of data block

##### 10.15.4.7.13 daucmfsi2c-Si2c2Intr-LLR-012

Requirement ID: H698-LLD-CMU-FNC-2160

The function shall set pu8Buffer with index (increment of u16DataCnt of data block) of sSi2cInfo of data block to return value of I2cReceiveData with parameter (M\_I2C2) when Return value of the function I2cGetLastEvent with parameter M\_I2C2 is M\_I2C\_EVENT\_MASTER\_BYTE\_RECEIVED.

##### 10.15.4.7.14 daucmfsi2c-Si2c2Intr-LLR-013

Requirement ID: H698-LLD-CMU-FNC-2161

The function shall call I2cAcknowledgeConfig with parameter (M\_I2C2, DISABLE) and I2cGenerateStop with parameter (M\_I2C2, ENABLE) when

* Return value of the function I2cGetLastEvent with parameter M\_I2C2 is M\_I2C\_EVENT\_MASTER\_BYTE\_RECEIVED.
* u16DataCnt of data block added M\_ONE is equal to u16DataSize of sSi2cInfo of data block.

##### 10.15.4.7.15 daucmfsi2c-Si2c2Intr-LLR-014

Requirement ID: H698-LLD-CMU-FNC-2162

The function shall call TransferFinish with parameter(TRUE), when u16DataCnt of data block is equal to u16DataSize of sSi2cInfo of data block.

##### 10.15.4.7.16 daucmfsi2c-Si2c2Intr-LLR-015

Requirement ID: H698-LLD-CMU-FNC-2163

The function shall do nothing when return value of the function I2cGetLastEvent with parameter M\_I2C2 is other than M\_I2C\_EVENT\_MASTER\_MODE\_SELECT, M\_I2C\_EVT\_MAST\_BYTE\_TRANSMITTING, M\_I2C\_EVT\_MASTER\_TRANS\_MODE\_SEL, M\_I2C\_EVT\_MAST\_BYTE\_TRANSMITTED, M\_I2C\_EVENT\_MASTER\_REC\_MODE\_SEL, M\_I2C\_EVENT\_MASTER\_BYTE\_RECEIVED.

##### 10.15.4.7.17 daucmfsi2c-Si2c2Intr-LLR-016

Requirement ID: H698-LLD-CMU-FNC-2164

The function shall call OsIntExit function to tell uCOS that we are leaving the ISR.

### 10.15.5 Si2c2ErrIntr

Low Level Design Details about Si2c2Intr will follow in the sub sections.

#### 10.15.5.1 Brief Description

Si2c2ErrIntr function process the interrupt for SI2C error.

#### 10.15.5.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-012-CMU).

#### 10.15.5.3 List of global variables accessed and modified

Accessed: None

Modified: None

#### 10.15.5.4 Parameter list (Input/Output)

Input: None

Output: None

#### 10.15.5.5 Return Value

None

#### 10.15.5.6 Other CSUs called by this CSU

OsIntEnter

I2cClearITPendingBit

I2cGetITStatus

I2cGenerateStop

TransferFinish

OsIntExit

#### 10.15.5.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to Si2c2ErrIntr

##### 10.15.5.7.1 daucmfsi2c-Si2c2ErrIntr-LLR-001

Requirement ID: H698-LLD-CMU-FNC-2173

The function shall call OsIntEnter function to tell uCOS that it is entering the ISR and function shall call I2cClearITPendingBit with parameters (M\_I2C2, M\_I2C\_IT\_AF) when the return of the function call I2cGetITStatus with parameters (M\_I2C2, M\_I2C\_IT\_AF) is equal to SET.

##### 10.15.5.7.2 daucmfsi2c-Si2c2ErrIntr-LLR-002

Requirement ID: H698-LLD-CMU-FNC-2174

The function shall call I2cClearITPendingBit with parameters (M\_I2C2, M\_I2C\_IT\_BERR) when the return of the function call I2cGetITStatus with parameters (M\_I2C2, M\_I2C\_IT\_BERR) is equal to SET.

##### 10.15.5.7.3 daucmfsi2c-Si2c2ErrIntr-LLR-003

Requirement ID: H698-LLD-CMU-FNC-2175

* The function shall call I2cGenerateStop with parameter (M\_I2C2, ENABLE) to reset the bus by sending stop condition.
* The function shall call TransferFinish function with parameter(FALSE) to terminate the transfer and set the error flag
* The function shall call OsIntExit function to tell uCOS that it is leaving the ISR.

## 10.16 daucmfrterr

daucmfrterr module provides routines to display run time errors.

### 10.16.1 InitializeLeds

Low Level Design Details about CSU InitializeLedswill follow in the sub sections.

#### 10.16.1.1 Brief Description

The function InitializeLeds initilizes the Red and Green LEDs.

#### 10.16.1.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD (H698-003-012-CMU).

#### 10.16.1.3 List of global variables accessed and modified

Accessed : None

Modified : None

#### 10.16.1.4 Parameter list (Input/Output)

Inputs : None

Outputs : None

#### 10.16.1.5 Return Value

None.

#### 10.16.1.6 Other CSUs called by this CSU

None

#### 10.16.1.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to InitializeLeds

##### 10.16.1.7.1 daucmfrterr-InitializeLeds-LLR-002

Requirement ID: H698-LLD-CMU-FNC-2065

The function shall enable GPIO port B Clock by setting M\_RCC\_AHB1PERIPH\_GPIOB to RCC AHB1 Peripheral Clock register(ahb1enr of M\_RCC)(i.e ahb1enr of M\_RCC to ahb1enr of M\_RCC bitwise OR M\_RCC\_AHB1PERIPH\_GPIOB)

##### 10.16.1.7.2 daucmfrterr-InitializeLeds-LLR-003

Requirement ID: H698-LLD-CMU-FNC-2066

The function shall turn on HB LED by setting following M\_GPIOB registers:

Set GPIO port mode register(moder of M\_GPIOB) with M\_MODER\_HB\_LED\_ON

Set GPIO port output speed register(ospeedr of M\_GPIOB) with M\_OSPEEDR\_HB\_LED\_ON

Set GPIO port output type register(otyper of M\_GPIOB) with M\_OTYPER\_HB\_LED\_ON

Set GPIO port pull-up/pull-down register(pupdr of M\_GPIOB) with M\_PUPDR\_HB\_LED\_ON

Set GPIO port bit set/reset low register(bsrrl of M\_GPIOB) with M\_HW\_LED\_HB.

##### 10.16.1.7.3 daucmfrterr-InitializeLeds-LLR-004

Requirement ID: H698-LLD-CMU-FNC-2067

The function shall enable the IO port C clock by setting the 3rd bit of the RCC AHB1 peripheral clock register (ahb1enr of M\_RCC) (i.e ahb1enr of M\_RCC to ahb1enr of M\_RCC bitwise OR M\_RCC\_AHB1PERIPH\_GPIOC).

##### 10.16.1.7.4 daucmfrterr-InitializeLeds-LLR-005

Requirement ID: H698-LLD-CMU-FNC-2068

The function shall configure the GPIO port C output pins as follows

- Mode register (moder of M\_GPIOC) set to M\_GPIOC\_MODER\_SET

- Output Speed register (ospeedr of M\_GPIOC) set to M\_GPIOC\_OSPEEDR\_SET

- Output type register (otyper of M\_GPIOC) set to M\_GPIOC\_OTYPER\_SET

- Pull-up/pull-down register (pupdr of M\_GPIOC) set to M\_GPIOC\_PUPDR\_SET.

##### 10.16.1.7.5 daucmfrterr-InitializeLeds-LLR-006

Requirement ID: H698-LLD-CMU-FNC-2069

The function shall set warn lines High to Disable Red LEDs by setting the pin 14 (M\_HW\_LED\_GREEN) and pin 15 (M\_HW\_LED\_RED) in the GPIO port C bit set/reset high register (bsrrh of M\_GPIOC).

### 10.16.2 RterrDisplayErrorId

Low Level Design Details about CSU RterrDisplayErrorId will follow in the sub sections.

#### 10.16.2.1 Brief Description

The function RterrDisplayErrorId attempts to display the input error on the Digital Display and then locks up the indicator.

#### 10.16.2.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD (H698-003-012-CMU).

#### 10.16.2.3 List of global variables accessed and modified

Accessed : None

Modified : None

#### 10.16.2.4 Parameter list (Input/Output)

Inputs : T\_ERROR\_TYPE ucerr - Error Number range [0,9]

T\_BOOL (\*forever)() - test forever function

T\_UINT16 number - Sub error number

Outputs : None

#### 10.16.2.5 Return Value

None.

#### 10.16.2.6 Other CSUs called by this CSU

WriteErrorToFlash

InitializeLeds

#### 10.16.2.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to RterrDisplayErrorId

##### 10.16.2.7.1 daucmfrterr-RterrDisplayErrorId-LLR-001

Requirement ID: H698-LLD-CMU-FNC-1050

The function shall disable all the interrupts and call InitializeLeds.

##### 10.16.2.7.2 daucmfrterr-RterrDisplayErrorId-LLR-007

Requirement ID: H698-LLD-CMU-FNC-1056

The function shall call WriteErrorToFlash with parameter u8\_app\_id and u16\_error.

##### 10.16.2.7.3 daucmfrterr-RterrDisplayErrorId-LLR-008

Requirement ID: H698-LLD-CMU-FNC-1057

The function shall loop forever, loops u32 bit from M\_SIXTEEN to M\_ZERO and loops the b on off from M\_ZERO to M\_TWO minus 1and performs the following

Set the bssrl of M\_GPIOC to when the (error code bitwise AND 1 left shift of u32 bit),set to M\_HW\_LED\_GREEN else set to M\_HW\_LED\_RED and Set the delay count to M\_FIVE

When b on off AND decrement of u32 bit less than M\_TWELVE

##### 10.16.2.7.4 daucmfrterr-RterrDisplayErrorId-LLR-009

Requirement ID: H698-LLD-CMU-FNC-1058

The function shall loop forever, loops u32 bit from M\_SIXTEEN to M\_ZERO and loops the b on off from M\_ZERO to M\_TWO minus 1and performs the following

Set the bssrh of M\_GPIOC to M\_HW\_LED\_GREEN bitwise OR with M\_HW\_LED\_RED and Set the u8 delay count to M\_TWO or M\_EIGHT when u32 bit modulus with M\_FOUR

When not of (b on off AND decrement of u32 bit less than M\_TWELVE)

##### 10.16.2.7.5 daucmfrterr-RterrDisplayErrorId-LLR-010

Requirement ID: H698-LLD-CMU-FNC-1059

The function shall loop forever, loops u32 bit from M\_SIXTEEN to M\_ZERO and loops the b on off from M\_ZERO to M\_TWO minus 1and performs the following

Set the kr of M\_IWDG to WATCHDOG\_RESET when loops from M\_ZERO to M\_DELAY minus 1

while loop decrements the u8 delay count

### 10.16.3 RterrForever

Low Level Design Details about CSU RterrForever will follow in the sub sections.

#### 10.16.3.1 Brief Description

The function RterrForever returns TRUE for infinite loops.

#### 10.16.3.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD (H698-003-012-CMU).

#### 10.16.3.3 List of global variables accessed and modified

Accessed : None

Modified : None

#### 10.16.3.4 Parameter list (Input/Output)

Inputs : None

Outputs : None

#### 10.16.3.5 Return Value

T\_BOOL - return TRUE

#### 10.16.3.6 Other CSUs called by this CSU

None

#### 10.16.3.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to RterrForever.

##### 10.16.3.7.1 daucmfrterr-RterrForever-LLR-001

Requirement ID: H698-LLD-CMU-FNC-1073

The function shall return TRUE.

### 10.16.4 CheckValidErrorFlash

Low Level Design Details about CSU CheckValidErrorFlash will follow in the sub sections.

#### 10.16.4.1 Brief Description

CheckValidErrorFlash function checks CRC of Error Flash area.

#### 10.16.4.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD (H698-003-012-CMU).

#### 10.16.4.3 List of global variables accessed and modified

Accessed : None

Modified : None

#### 10.16.4.4 Parameter list (Input/Output)

Inputs : None

Outputs : None

#### 10.16.4.5 Return Value

T\_BOOL - return validity of Error Flash area

#### 10.16.4.6 Other CSUs called by this CSU

None

#### 10.16.4.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to CheckValidErrorFlash.

##### 10.16.4.7.1 daucmfrterr-CheckValidErrorFlash-LLR-001

Requirement ID: H698-LLD-CMU-FNC-1082

The function shall set temporary pointer with M\_MEMMAP\_FIRST\_ERR\_REPORT\_ADDR and CRC Control register(cr of M\_CRC) with M\_CRC\_RESET\_BIT.

##### 10.16.4.7.2 daucmfrterr-CheckValidErrorFlash-LLR-002

Requirement ID: H698-LLD-CMU-FNC-1083

The function shall set local buffer with M\_MEMMAP\_FIRST\_ERR\_REPORT\_ADDR and count value equal to (division of sizeof (T\_RTERR\_REPORTSTRUCT) by M\_FOUR) subtract with M\_ONE.

##### 10.16.4.7.3 daucmfrterr-CheckValidErrorFlash-LLR-003

Requirement ID: H698-LLD-CMU-FNC-1084

The function shall store the error log (local buffer value) in CRC Data register (dr of M\_CRC), for all the indices of local buffer from M\_ZERO to count value.

##### 10.16.4.7.4 daucmfrterr-CheckValidErrorFlash-LLR-004

Requirement ID: H698-LLD-CMU-FNC-1085

The function shall store CRC Data register value (dr of M\_CRC) in check sum calculation variable and crc value of temporary pointer in check sum calculation2 variable.

##### 10.16.4.7.5 daucmfrterr-CheckValidErrorFlash-LLR-005

Requirement ID: H698-LLD-CMU-FNC-1086

The function shall return FALSE when check sum calculation is not equal to check sum calculation2, otherwise returns TRUE.

### 10.16.5 WriteErrorToFlash

Low Level Design Details about CSU WriteErrorToFlash will follow in the sub sections.

#### 10.16.5.1 Brief Description

WriteErrorToFlash function writes the Error to Flash.

#### 10.16.5.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD (H698-003-012-CMU).

#### 10.16.5.3 List of global variables accessed and modified

Accessed : None

Modified : None

#### 10.16.5.4 Parameter list (Input/Output)

Inputs : T\_SINT32 err\_code - error number

T\_SINT32 err\_sub\_num - Sub error number

Outputs : None

#### 10.16.5.5 Return Value

None

#### 10.16.5.6 Other CSUs called by this CSU

CheckValidErrorFlash

EraseErrSection

ProgramWord

#### 10.16.5.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to WriteErrorToFlash

##### 10.16.5.7.1 daucmfrterr-WriteErrorToFlash-LLR-001

Requirement ID: H698-LLD-CMU-FNC-1095

The function shall set IWDG Key register(kr of M\_IWDG) with M\_IWDG\_KR\_RELOAD to prevent watchdog reset.

##### 10.16.5.7.2 daucmfrterr-WriteErrorToFlash-LLR-002

Requirement ID: H698-LLD-CMU-FNC-1096

The function shall write error log in flash(M\_MEMMAP\_FIRST\_ERR\_REPORT\_ADDR) to a reference of temporary report buffer from M\_ZERO to size of T\_RTERR\_REPORTSTRUCT when CheckValidErrorFlash function returns true.

##### 10.16.5.7.3 daucmfrterr-WriteErrorToFlash-LLR-003

Requirement ID: H698-LLD-CMU-FNC-1097

The function shall reset the below members of temporary report structure when CheckValidErrorFlash returns false

-Set total\_count to M\_ZERO

-Set crc\_val to M\_ZERO

-Set the all the below error\_list members (error list size is M\_FIFTYONE)

a). error\_code to M\_MINUS\_ONE

b). sub\_id to M\_ZERO

c). error\_count to M\_ZERO

##### 10.16.5.7.4 daucmfrterr-WriteErrorToFlash-LLR-004

Requirement ID: H698-LLD-CMU-FNC-1098

The function shall increment the total\_count of temporary report structure by one.

##### 10.16.5.7.5 daucmfrterr-WriteErrorToFlash-LLR-005

Requirement ID: H698-LLD-CMU-FNC-1099

The function shall store err\_code, err\_sub\_num and M\_ONE in error\_code, sub\_id and error\_count of temporary report structure respectively for all the error list (size is M\_FIFTY) and stop execution from the current condition, when error\_code from error\_list of temporary report structure is M\_MINUS\_ONE.

##### 10.16.5.7.6 daucmfrterr-WriteErrorToFlash-LLR-006

Requirement ID: H698-LLD-CMU-FNC-1100

For all the error list (size is M\_FIFTY), the function shall do the following:

a). Increment error\_count from error\_list of temporary report structure by one, when

error\_count from error\_list of temporary report structure is less than M\_ERROR\_COUNT\_999, otherwise do nothing.

b). Stop execution from the current condition

when the following conditions are satisfied

- error\_code of temporary report structure is not equal to M\_MINUS\_ONE

- error\_code and sub\_id of temporary report structure is equal to error\_code and err\_sub\_num respectively, otherwise do nothing.

##### 10.16.5.7.7 daucmfrterr-WriteErrorToFlash-LLR-007

Requirement ID: H698-LLD-CMU-FNC-1101

The function shall do the following:

-set CRC Control register (cr of M\_CRC) with M\_CRC\_RESET\_BIT.

-set buffer with address of temporary report structure.

-count value with (division of sizeof (T\_RTERR\_REPORTSTRUCT) by M\_FOUR) subtract with M\_ONE

##### 10.16.5.7.8 daucmfrterr-WriteErrorToFlash-LLR-008

Requirement ID: H698-LLD-CMU-FNC-1102

The function shall store the error log in Flash (buffer) to CRC Data register (dr of M\_CRC) from M\_ZERO to count value.

##### 10.16.5.7.9 daucmfrterr-WriteErrorToFlash-LLR-009

Requirement ID: H698-LLD-CMU-FNC-1103

The function shall store CRC Data register value (dr of M\_CRC) in check sum calculation variable and stores the check sum calculation in crc\_val of temporary report structure.

##### 10.16.5.7.10 daucmfrterr-WriteErrorToFlash-LLR-010

Requirement ID: H698-LLD-CMU-FNC-1104

The function shall configure FLASH key register by storing M\_FLASH\_KEY1

and M\_FLASH\_KEY2 in keyr of M\_FLASH to unlock FLASH control register when Bitwise AND of cr of M\_FLASH with M\_FLASH\_CR\_LOCK is not equal to RESET otherwise do nothing.

##### 10.16.5.7.11 daucmfrterr-WriteErrorToFlash-LLR-011

Requirement ID: H698-LLD-CMU-FNC-1105

The function shall configure FLASH status register (sr of M\_FLASH) with M\_FLASH\_OPERATION\_BITS and reset the count value.

##### 10.16.5.7.12 daucmfrterr-WriteErrorToFlash-LLR-012

Requirement ID: H698-LLD-CMU-FNC-1106

The function shall set register pointer with M\_MEMMAP\_FRST\_ER\_REPORT\_ADDR\_REG,load register buffer with error log in Flash register pointer for all errors from M\_ZERO to M\_TWO\_HUNDRED and then reset the count value.

##### 10.16.5.7.13 daucmfrterr-WriteErrorToFlash-LLR-013

Requirement ID: H698-LLD-CMU-FNC-1107

The function shall store the reference of temporary report structure to source pointer and return when erasing the Error Section in Memory (EraseErrSection) is equal to FALSE otherwise do nothing.

##### 10.16.5.7.14 daucmfrterr-WriteErrorToFlash-LLR-014

Requirement ID: H698-LLD-CMU-FNC-1108

The function shall set source pointer with register buffer and destination pointer with M\_MEMMAP\_FRST\_ER\_REPORT\_ADDR\_REG.

##### 10.16.5.7.15 daucmfrterr-WriteErrorToFlash-LLR-015

Requirement ID: H698-LLD-CMU-FNC-1109

The function shall stop programming the data when return value of the function ProgramWord with parameters destination pointer and dereferencing of source pointer is FALSE for calibration data from M\_ZERO to product of M\_TWO\_HUNDRED and M\_FOUR (increment the loop counter by FOUR), otherwise do nothing.

##### 10.16.5.7.16 daucmfrterr-WriteErrorToFlash-LLR-016

Requirement ID: H698-LLD-CMU-FNC-1110

The function shall increment the destination poineter by M\_FOUR and source pointer by M\_ONE for calibration data from M\_ZERO to product of M\_TWO\_HUNDRED and M\_FOUR (increment the loop counter by FOUR).

##### 10.16.5.7.17 daucmfrterr-WriteErrorToFlash-LLR-017

Requirement ID: H698-LLD-CMU-FNC-1111

The function shall set source buffer with address of temporary report structure and destination pointer with M\_MEMMAP\_FIRST\_ERR\_REPORT\_ADDR.

##### 10.16.5.7.18 daucmfrterr-WriteErrorToFlash-LLR-018

Requirement ID: H698-LLD-CMU-FNC-1112

The function shall stop programming the data when return value of the function ProgramWord with parameters destination pointer and dereferencing of source pointer is FALSE for calibration data from M\_ZERO to sizeof temporary report structure (increment the loop counter by FOUR), otherwise do nothing.

##### 10.16.5.7.19 daucmfrterr-WriteErrorToFlash-LLR-019

Requirement ID: H698-LLD-CMU-FNC-1113

The function shall increment the destination pointer by M\_FOUR and source pointer by M\_ONE for calibration data from M\_ZERO to size of temporary report structure (increment the loop counter by FOUR).

##### 10.16.5.7.20 daucmfrterr-WriteErrorToFlash-LLR-020

Requirement ID: H698-LLD-CMU-FNC-1114

The function shall lock the FLASH control register by performing Bitwise OR operation of cr of M\_FLASH with M\_FLASH\_CR\_LOCK and then return from the function.

### 10.16.6 EraseErrSection

Low Level Design Details about CSU EraseErrSection will follow in the sub sections.

#### 10.16.6.1 Brief Description

EraseErrSection function is an Erase Error Section in Memory.

#### 10.16.6.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD (H698-003-012-CMU).

#### 10.16.6.3 List of global variables accessed and modified

Accessed : None

Modified : None

#### 10.16.6.4 Parameter list (Input/Output)

Inputs : None

Outputs : None

#### 10.16.6.5 Return Value

T\_BOOL - Return the Erase Status

#### 10.16.6.6 Other CSUs called by this CSU

None

#### 10.16.6.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to EraseErrSection.

##### 10.16.6.7.1 daucmfrterr-EraseErrSection-LLR-001

Requirement ID: H698-LLD-CMU-FNC-1123

The function shall increment the loop counter until loop counter is less than M\_TENK AND ongoing programs in FLASH status register (sr of M\_FLASH) is not M\_ZERO.

##### 10.16.6.7.2 daucmfrterr-EraseErrSection-LLR-002

Requirement ID: H698-LLD-CMU-FNC-1124

The function shall erase the sector, configure and start Flash memory operations

by setting FLASH CR register(cr of M\_FLASH) as below when loop counter is less than M\_TENK :

- reset Flash CR register by setting it with (cr of M\_FLASH Bitwise AND M\_CR\_PSIZE\_MASK) to select program size x8

- set Flash CR register with cr of M\_FLASH Bitwise OR M\_FLASH\_PSIZE\_WORD to select program size x32

- reset Flash CR register sector number bits by setting it with (cr of M\_FLASH Bitwise AND M\_ERASE\_SECTOR0) to erase sector0

- activate Sector Erase for SECTOR\_4 (cr of M\_FLASH Bitwise OR M\_FLASH\_CR\_SER Bitwise OR M\_FLASH\_SECTOR\_4)

- set Flash CR register with cr of M\_FLASH Bitwise OR M\_FLASH\_CR\_STRT.

##### 10.16.6.7.3 daucmfrterr-EraseErrSection-LLR-003

Requirement ID: H698-LLD-CMU-FNC-1125

The function shall do the following when loop counter is less than M\_TENK:

-reset loop counter

-increment loop counter until loop counter is less than M\_TENK AND ongoing programs in FLASH status register (sr of M\_FLASH) is not zero.

##### 10.16.6.7.4 daucmfrterr-EraseErrSection-LLR-004

Requirement ID: H698-LLD-CMU-FNC-1126

The function shall disable Sector Erase (SER Bit) and reset Flash CR register sector by performing the following when loop counter is less than M\_TENK, otherwise do nothing:

- Set cr of M\_FLASH with Bitwise AND operation of cr of M\_FLASH with negation of M\_FLASH\_CR\_SER

- Set cr of M\_FLASH with Bitwise AND operation of cr of M\_FLASH with M\_ERASE\_SECTOR0

##### 10.16.6.7.5 daucmfrterr-EraseErrSection-LLR-005

Requirement ID: H698-LLD-CMU-FNC-1127

The function shall return TRUE when loop counter is less than M\_TENK, otherwise return FALSE.

### 10.16.7 ProgramWord

Low Level Design Details about CSU ProgramWord will follow in the sub sections.

#### 10.16.7.1 Brief Description

ProgramWord function programs a word to Flash.

#### 10.16.7.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD (H698-003-012-CMU).

#### 10.16.7.3 List of global variables accessed and modified

Accessed : None

Modified : None

#### 10.16.7.4 Parameter list (Input/Output)

Inputs : T\_UINT32 data - Program word data

Outputs : T\_UINT32 address - Program word address

#### 10.16.7.5 Return Value

T\_BOOL - return TRUE when program operation is completed

- return FALSE when program operation is not completed

#### 10.16.7.6 Other CSUs called by this CSU

None

#### 10.16.7.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to ProgramWord

##### 10.16.7.7.1 daucmfrterr-ProgramWord-LLR-001

Requirement ID: H698-LLD-CMU-FNC-1136

The function shall loop and increment loop counter until loop counter is less than M\_TENK and ongoing programs in FLASH status register (sr of M\_FLASH) is not M\_ZERO.

##### 10.16.7.7.2 daucmfrterr-ProgramWord-LLR-002

Requirement ID: H698-LLD-CMU-FNC-1137

The function shall configure CR register (cr of M\_FLASH) to program the new data when loop counter is less than M\_TENK by performing the following operations:

-reset Flash CR register by setting it with cr of M\_FLASH Bitwise AND M\_CR\_PSIZE\_MASK to select program size x8

-set Flash CR register with cr of M\_FLASH Bitwise OR M\_FLASH\_PSIZE\_WORD to select program size x32

-set Flash CR register with cr of M\_FLASH Bitwise OR M\_FLASH\_CR\_PG.

-set address with data

##### 10.16.7.7.3 daucmfrterr-ProgramWord-LLR-003

Requirement ID: H698-LLD-CMU-FNC-1138

The function shall do the following when loop counter is less than M\_TENK.:

- reset loop counter

- increment loop counter until loop counter is less than M\_TENK and ongoing programs in FLASH status register (sr of M\_FLASH) is not M\_ZERO

##### 10.16.7.7.4 daucmfrterr-ProgramWord-LLR-004

Requirement ID: H698-LLD-CMU-FNC-1139

The function shall reset Programming (PG) bit of FLASH control register by performing Bitwise AND operation of cr of M\_FLASH with negation of M\_FLASH\_CR\_PG when loop counter is less than M\_TENK, otherwise do nothing.

##### 10.16.7.7.5 daucmfrterr-ProgramWord-LLR-005

Requirement ID: H698-LLD-CMU-FNC-1140

The function shall return TRUE when loop counter is less than M\_TENK, otherwise return FALSE.

## 10.17 daucmftbase

daucmftbase CSC contains implementation of Time base routines for signalling Semaphores.

### 10.17.1 TbaseTaskSignaling

Low Level Design Details about CSU TbaseTaskSignaling will follow in the sub sections.

#### 10.17.1.1 Brief Description

TbaseTaskSignaling function installs the following task signalling parameters into the time base to be serviced

- task ticks

- Semaphore of the task.

#### 10.17.1.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-012-CMU).

#### 10.17.1.3 List of global variables accessed and modified

Accessed : None

Modified : None

#### 10.17.1.4 Parameter list (Input/Output)

Inputs : T\_UINT16 u16\_task\_ticks – Task ticks to delay this thread

T\_OS\_EVENT \*ps\_semaphore - pointer to Semaphore to be posted for thread

Outputs : None

#### 10.17.1.5 Return Value

None

#### 10.17.1.6 Other CSUs called by this CSU

RterrDisplayErrorId

RterrForever

#### 10.17.1.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to TbaseTaskSignaling

##### 10.17.1.7.1 daucmftbase-TbaseTaskSignaling-LLR-001

Requirement ID: H698-LLD-CMU-FNC-1217

The function shall call the function RterrDisplayErrorId with parameters (M\_APP\_ID, RterrForever function, TBASE\_ERROR Bitwise OR with RT\_ERROR\_1) and returns when the total number of tasks installed has reached the maximum limit (>=) of M\_OS\_MAX\_TASKS. otherwise do nothing.

##### 10.17.1.7.2 daucmftbase-TbaseTaskSignaling-LLR-002

Requirement ID: H698-LLD-CMU-FNC-1218

The function shall reset the tick counter of the time base task list to M\_ZERO.

##### 10.17.1.7.3 daucmftbase-TbaseTaskSignaling-LLR-003

Requirement ID: H698-LLD-CMU-FNC-1219

The function shall set u16\_task\_ticks of the time base task list to the task ticks

##### 10.17.1.7.4 daucmftbase-TbaseTaskSignaling-LLR-004

Requirement ID: H698-LLD-CMU-FNC-1220

The function shall set ps\_semaphore of time base task list.to the Semaphore.

##### 10.17.1.7.5 daucmftbase-TbaseTaskSignaling-LLR-005

Requirement ID: H698-LLD-CMU-FNC-1221

The function shall increment number of installed tasks by M\_ONE.

### 10.17.2 TbaseIntrHandler

Low Level Design Details about CSU TbaseIntrHandler shall follow in the sub sections.

#### 10.17.2.1 Brief Description

#### 10.17.2.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-012-CMU).

#### 10.17.2.3 List of global variables accessed and modified

Accessed : None

Modified : None

#### 10.17.2.4 Parameter list (Input/Output)

Inputs : None

Outputs : None

#### 10.17.2.5 Return Value

None

#### 10.17.2.6 Other CSUs called by this CSU

OsIntEnter

OsTimeTick

OsSemPost

OsIntExit

RterrDisplayErrorId

RterrForever

#### 10.17.2.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to TbaseIntrHandler

##### 10.17.2.7.1 daucmftbase-TbaseIntrHandler-LLR-001

Requirement ID: H698-LLD-CMU-FNC-1230

The function shall inform the Micro C OS that, it is entering the ISR by calling OsIntEnter.

##### 10.17.2.7.2 daucmftbase-TbaseIntrHandler-LLR-002

Requirement ID: H698-LLD-CMU-FNC-1231

The function shall update the time tick by calling OsTimeTick.

##### 10.17.2.7.3 daucmftbase-TbaseIntrHandler-LLR-003

Requirement ID: H698-LLD-CMU-FNC-1232

The function shall loop through the time base installed task and perform the following for each task in the list. when the task tick counter (incremented by 1) of task list is greater than or equal to the task ticks of the corresponding task. Otherwise do nothing.

- Reset the task tick counter to M\_ZERO.

- Signal the Semaphore of the task (M\_OS\_NO\_ERR) by calling OsSemPost with Semaphore of the corresponding task as parameter

and calls the function RterrDisplayErrorId with parameter (M\_APP\_ID, RterrForever, OS\_KERNEL\_ERR Bitwise OR with RT\_ERROR\_9) when the function OsSemPost returns other than M\_OS\_NO\_ERR, otherwise do nothing.

##### 10.17.2.7.4 daucmftbase-TbaseIntrHandler-LLR-004

Requirement ID: H698-LLD-CMU-FNC-1233

The function shall inform the Micro C OS, that it is leaving the ISR by calling OsIntExit.

### 10.17.3 TbaseInit

Low Level Design Details about CSU TbaseInit shall follow in the sub sections.

#### 10.17.3.1 Brief Description

The function TbaseInit initializes the Time base.

#### 10.17.3.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-012-CMU).

#### 10.17.3.3 List of global variables accessed and modified

Accessed : None

Modified : None

#### 10.17.3.4 Parameter list (Input/Output)

Inputs : None

Outputs : None

#### 10.17.3.5 Return Value

None

#### 10.17.3.6 Other CSUs called by this CSU

SysTickConfig

#### 10.17.3.7 Description of list of LLRs allocated

The following section shall list the LLRs allocated to TbaseInit.

##### 10.17.3.7.1 daucmftbase-TbaseInit-LLR-001

Requirement ID: H698-LLD-CMU-FNC-1242

The function shall initialize the system tick interrupt period by calling SysTickConfig with number of ticks between two interrupts as parameter (division of M\_HW\_SYSTEM\_CLOCK by M\_OS\_TICKS\_PER\_SEC).

## 10.18 daucmfucos

daucmfucos CSC defines the implementation of uC/OS routines for the real-time kernel.

### 10.18.1 OsInit

Low Level Design Details about CSU OsInit will follow in the sub sections.

#### 10.18.1.1 Brief Description

OsInit is the uC/OS initialization function.

The function initializes the Os ready to run task list, TCB priority table list, list of free TCBs, list of free Event Control Blocks to default values. The function creates the OsTaskIdle function.

(Note : OSInit must be called before OSStart).

#### 10.18.1.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-012-CMU).

#### 10.18.1.3 List of global variables accessed and modified

Accessed : Os\_task\_idle\_stk

Modified : Os\_tcb\_high\_rdy,

Os\_tcb\_cur,

Os\_running,

Os\_tcb\_prio\_tbl

#### 10.18.1.4 Parameter list (Input/Output)

Inputs: None

Outputs: None

#### 10.18.1.5 Return Value

None

#### 10.18.1.6 Other CSUs called by this CSU

OsTaskCreate

OsTaskIdle

#### 10.18.1.7 Description of list of LLRs allocated

The following section shall list the LLRs allocated to OsInit.

##### 10.18.1.7.1 daucmfucos-OsInit-LLR-001

Requirement ID: H698-LLD-CMU-FNC-1252

The function shall initialize the following:

- Current value of System Time to M\_ZERO

- Pointer to highest priority TCB ready to run (Os\_tcb\_high\_rdy) to M\_NULL.

- Pointer to currently running TCB (Os\_tcb\_cur) to M\_NULL.

- Pointer to doubly linked list of TCBs to M\_NULL

- Interrupt nesting level, Multitasking lock nesting level to M\_ZERO.

- Flag indicating that Kernel is running (Os\_running) to FALSE.

- Counter of number of context switches to M\_ZERO.

- Ready list group to M\_ZERO.

- all the indices of the Os ready to run list (size is M\_RDY\_LST\_SIZE) to M\_ZERO

- all the indices of the Os\_tcb\_prio\_tbl (size is M\_MAX\_TCB\_PRIO) to M\_NULL.

- os\_tcb\_next of each index of OS TCB list (size is M\_OS\_MAX\_TASKS) to the next element of TCB list (except the last element).

- os\_tcb\_next of last index of OS TCB list to M\_NULL.

- OS TCB free list to the base address of OS TCB list.

- os\_eventptr of each index of free Event Control Blocks list (size is M\_OS\_MAX\_EVENTS minus 1) to the next element Event Control Blocks list(except the last element).

- os\_eventptr of last index of free Event Control Blocks list to M\_NULL.

- OS EVENT free list to the base address of free Event Control Blocks list.

##### 10.18.1.7.2 daucmfucos-OsInit-LLR-002

Requirement ID: H698-LLD-CMU-FNC-1253

The function shall do the following when M\_OS\_NO\_ERR is not equal to Os Idle Task by calling 'OsTaskCreate' with parameters

a) pointer to function OsTaskIdle,

b) M\_NULL as task entry point,

c) Reference of Os\_task\_idle\_stk with index M\_OS\_IDLE\_TASK\_STK\_SIZE

d) M\_OS\_LOWEST\_PRIO as task priority

Set the function RterrDisplayErrorId with parameters as (M\_APP\_ID,RterrForever,OS\_KERNEL\_ERR Bitwise OR with RT\_ERROR\_12). otherwise does nothing.

### 10.18.2 OsTaskIdle

Low Level Design Details about CSU OsTaskIdle will follow in the sub sections.

#### 10.18.2.1 Brief Description

The OsTaskIdle function is the Os Idle task to keep track of the CPU idle time and reset the watch dog counter.

#### 10.18.2.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-012-CMU).

#### 10.18.2.3 List of global variables accessed and modified

Accessed : U32\_critical\_sr

Modified : U32\_critical\_sr

#### 10.18.2.4 Parameter list (Input/Output)

Inputs: void \*p\_data - Not used, always executed with null reference

Outputs: None

#### 10.18.2.5 Return Value

None

#### 10.18.2.6 Other CSUs called by this CSU

SaveStatusReg

RestoreStatusReg

WdogKickWatchDog

#### 10.18.2.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to OsTaskIdle.

##### 10.18.2.7.1 daucmfucos-OsTaskIdle-LLR-001

Requirement ID: H698-LLD-CMU-FNC-1262

The function shall loop infinitely and do the following:

-Call WdogKickWatchDog to reset the watchdog timer.

### 10.18.3 OsStart

Low Level Design Details about CSU OsStart will follow in the sub sections.

#### 10.18.3.1 Brief Description

OsStart function starts the multitasking process, allowing uC/OS to manage the tasks that have been created.

(NOTE :

a)Before OsStart is called ,OsInit() must have been called and at least one

task must have been created.

b)OsStart sets Os\_tcb\_high\_rdy to point to the Os\_tcb of the highest priority

task.).

#### 10.18.3.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-012-CMU).

#### 10.18.3.3 List of global variables accessed and modified

Accessed : Os\_tcb\_high\_rdy,

Os\_tcb\_prio\_tbl

Modified : Os\_tcb\_high\_rdy,

Os\_tcb\_cur.

Os\_running

#### 10.18.3.4 Parameter list (Input/Output)

Inputs: None

Outputs: None

#### 10.18.3.5 Return Value

None

#### 10.18.3.6 Other CSUs called by this CSU

OsStartHighRdy

#### 10.18.3.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to OsStart.

##### 10.18.3.7.1 daucmfucos-OsStart-LLR-001

Requirement ID: H698-LLD-CMU-FNC-1271

The function shall do the following:

- Set highest priority’s task number y with Os unmap table of index ready group list

- Set highest priority’s task number x with Os unmap table of index (Os ready table of index highest priority’s task number y)

- Set priority with (addition of (bit shift of highest priority’s task number y to left by M\_THREE) and highest priority’s task number x )

- Set Os\_tcb\_high\_rdy with Os\_tcb\_prio\_tbl with index priority

##### 10.18.3.7.2 daucmfucos-OsStart-LLR-002

Requirement ID: H698-LLD-CMU-FNC-1272

The function shall set the pointer to currently running TCB (Os\_tcb\_cur) to pointer to highest priority TCB ready to run (Os\_tcb\_high\_rdy).

##### 10.18.3.7.3 daucmfucos-OsStart-LLR-003

Requirement ID: H698-LLD-CMU-FNC-1273

The function shall set the Os\_running to TRUE.

##### 10.18.3.7.4 daucmfucos-OsStart-LLR-004

Requirement ID: H698-LLD-CMU-FNC-1274

The function shall call 'OsStartHighRdy' to start running the highest priority task.

### 10.18.4 OsSched

Low Level Design Details about CSU OsSched will follow in the sub sections.

#### 10.18.4.1 Brief Description

OsSched function is the uC/OS scheduler which determines which task has the highest priority and thus will be next to run.

#### 10.18.4.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-012-CMU).

#### 10.18.4.3 List of global variables accessed and modified

Accessed : Os\_tcb\_high\_rdy

Os\_tcb\_cur

Os\_tcb\_prio\_tbl

U32\_critical\_sr

Modified : Os\_tcb\_high\_rdy

U32\_critical\_sr

#### 10.18.4.4 Parameter list (Input/Output)

Inputs: None

Outputs: None

#### 10.18.4.5 Return Value

None

#### 10.18.4.6 Other CSUs called by this CSU

SaveStatusReg

RestoreStatusReg

OsCtxSw

#### 10.18.4.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to OsSched.

##### 10.18.4.7.1 daucmfucos-OsSched-LLR-001

Requirement ID: H698-LLD-CMU-FNC-1283

The function shall set U32\_critical\_sr to return value of ‘SaveStatusReg’ (M\_OS\_ENTER\_CRITICAL) to disable interrupts before execution of the Scheduler begins.

##### 10.18.4.7.2 daucmfucos-OsSched-LLR-002

Requirement ID: H698-LLD-CMU-FNC-1284

The function shall do the following when the Interrupt nesting level Bitwise OR Multitasking lock nesting level is equal to M\_ZERO, Otherwise do nothing.

- Set the highest priority with OS unmap table of index ready group

- Set Os\_tcb\_high\_rdy with Os\_tcb\_prio\_tbl of index (addition of (bit shift of highest priority to left by M\_THREE) and Os unmap table of index (Os ready table of index highest priority))

##### 10.18.4.7.3 daucmfucos-OsSched-LLR-003

Requirement ID: H698-LLD-CMU-FNC-1285

The function shall perform the following operations when the task scheduling is enabled and the task to be scheduled next 'Os\_tcb\_high\_rdy' is not the current running task 'Os\_tcb\_cur'. Otherwise do nothing.

a) Increment context switch counter

b) call OsCtxSw to perform context switch using M\_OS\_TASK\_SW.

##### 10.18.4.7.4 daucmfucos-OsSched-LLR-004

Requirement ID: H698-LLD-CMU-FNC-1286

The function shall Call RestoreStatusReg (M\_OS\_EXIT\_CRITICAL) with parameter U32\_critical\_sr

to restore the priority mask register after the highest priority task is scheduled or if called from an ISR (i.e. Interrupt nesting level is greater than 0 ) or Scheduling has been disabled (i.e. Multitasking lock nesting level is greater than 0).

### 10.18.5 OsTcbInit

Low Level Design Details about CSU OsTcbInit will follow in the sub sections.

#### 10.18.5.1 Brief Description

OsTcbInit function initializes the task control block (TCB).

#### 10.18.5.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-012-CMU).

#### 10.18.5.3 List of global variables accessed and modified

Accessed : U32\_critical\_sr

Modified : U32\_critical\_sr

Os\_tcb\_prio\_tbl

#### 10.18.5.4 Parameter list (Input/Output)

Inputs: T\_UBYTE prio - Task priority

void \*stck - pointer to task stack.

Outputs: None

#### 10.18.5.5 Return Value

T\_UBYTE - returns error status

M\_OS\_NO\_ERR - TCB created successfully.

M\_OS\_NO\_MORE\_TCB - No more free TCB in the TCB list

#### 10.18.5.6 Other CSUs called by this CSU

RterrDisplayErrorId

SaveStatusReg

RestoreStatusReg

RterrForever

#### 10.18.5.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to OsTcbInit.

##### 10.18.5.7.1 daucmfucos-OsTcbInit-LLR-001

Requirement ID: H698-LLD-CMU-FNC-1295

The function shall set U32\_critical\_sr to return value of ‘SaveStatusReg’ (M\_OS\_ENTER\_CRITICAL) to disable interrupts before creation of TCB.

##### 10.18.5.7.2 daucmfucos-OsTcbInit-LLR-002

Requirement ID: H698-LLD-CMU-FNC-1296

The function shall get a free TCB from the free TCB list.

##### 10.18.5.7.3 daucmfucos-OsTcbInit-LLR-003

Requirement ID: H698-LLD-CMU-FNC-1297

The function shall perform following operations when free TCB is obtained successfully from the free TCB list.

a) Update the free TCB list with the pointer to the next TCB (os\_tcb\_next) in the free TCB list

b) Call RestoreStatusReg (M\_OS\_EXIT\_CRITICAL) with parameter U32\_critical\_sr to restore the priority mask register

c) set the following members of the free TCB list

- pointer to current stack(os\_tcb\_stkptr) to stack pointer 'stck'

- task priority (os\_tcb\_prio) to the current task priority 'prio'

- task status (os\_tcb\_stat) to M\_OS\_STAT\_RDY

- task delay (os\_tcb\_dly) to M\_ZERO

- os\_tcb\_y to current task priority 'prio' right shifted by M\_THREE

- os\_tcb\_bity to value of Os map table with index as os\_tcb\_y of free TCB list structure free

- os\_tcb\_x to current task priority 'prio' bitwise AND of M\_HEX\_SEVEN

- os\_tcb\_bitx to value of Os map table with index as os\_tcb\_x of free TCB list structure

- os\_tcb\_eventptr to M\_NULL

- set U32\_critical\_sr to return value of ‘SaveStatusReg’ (M\_OS\_ENTER\_CRITICAL) to disable interrupts

d) set Os\_tcb\_prio\_tbl with index 'prio' to free TCB

e) set the free TCB member

- os\_tcb\_next to Os tcb list

- os\_tcb\_prev to M\_NULL

f) Set os\_tcb\_prev of Os tcb list to free TCB list when Os tcb list is not M\_NULL, Otherwise do nothing.

g) set the Pointer to doubly linked list of TCBs to free TCB list

h) Make the task ready to run by updating the following values

- Ready group list Bitwise OR with os\_tcb\_bity of free TCB

- Os ready table index os\_tcb\_y of free TCB to Os ready table Bitwise OR with os\_tcb\_bitx of free TCB

i) Send a request to uC/OS to enable all the interrupt by calling RestoreStatusReg (M\_OS\_EXIT\_CRITICAL) with parameter U32\_critical\_sr

j) Return M\_OS\_NO\_ERR.

##### 10.18.5.7.4 daucmfucos-OsTcbInit-LLR-004

Requirement ID: H698-LLD-CMU-FNC-1298

The function shall perform following operations when free TCB is not obtained successfully from the free TCB list

a) Send a request to uC/OS to enable all the interrupt by calling RestoreStatusReg (M\_OS\_EXIT\_CRITICAL) with parameter U32\_critical\_sr

b) Call the function RterrDisplayErrorId with parameter M\_APP\_ID, RterrForever, OS\_KERNEL\_ERR Bitwise OR with RT\_ERROR\_10 to display the input error

c) Return 'M\_OS\_NO\_MORE\_TCB'.

### 10.18.6 OsIntEnter

Low Level Design Details about CSU OsIntEnter will follow in the sub sections.

#### 10.18.6.1 Brief Description

OsIntEnter function increments the ISR nesting level

The function is called to inform the uC/OS on entering the ISR.

#### 10.18.6.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-012-CMU).

#### 10.18.6.3 List of global variables accessed and modified

Accessed : U32\_critical\_sr

Modified : U32\_critical\_sr

#### 10.18.6.4 Parameter list (Input/Output)

Inputs: None

Outputs: None

#### 10.18.6.5 Return Value

None

#### 10.18.6.6 Other CSUs called by this CSU

SaveStatusReg

RestoreStatusReg

#### 10.18.6.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to OsIntEnter.

##### 10.18.6.7.1 daucmfucos-OsIntEnter-LLR-001

Requirement ID: H698-LLD-CMU-FNC-1307

The function shall do the following on entering an ISR

- Send a request to uC/OS to disable all the interrupt by setting U32\_critical\_sr to return value of ‘SaveStatusReg’ (M\_OS\_ENTER\_CRITICAL)

- Increment ISR nesting level counter

- Send a request to uC/OS to restore the priority mask register by calling RestoreStatusReg (M\_OS\_EXIT\_CRITICAL) with parameter U32\_critical\_sr.

### 10.18.7 OsIntExit

Low Level Design Details about CSU OsIntExit will follow in the sub sections

#### 10.18.7.1 Brief Description

OsIntExit function will decrement ISR nesting level and perform interrupt level context switching.

#### 10.18.7.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-012-CMU).

#### 10.18.7.3 List of global variables accessed and modified

Accessed : Os\_tcb\_high\_rdy

Os\_tcb\_cur

Os\_tcb\_prio\_tbl

U32\_critical\_sr

Modified : Os\_tcb\_high\_rdy

U32\_critical\_sr

#### 10.18.7.4 Parameter list (Input/Output)

Inputs: None

Outputs: None

#### 10.18.7.5 Return Value

None

#### 10.18.7.6 Other CSUs called by this CSU

SaveStatusReg

RestoreStatusReg

OsIntCtxSw

#### 10.18.7.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to OsIntExit.

##### 10.18.7.7.1 daucmfucos-OsIntExit-LLR-001

Requirement ID: H698-LLD-CMU-FNC-1316

The function shall set U32\_critical\_sr to return value of ‘SaveStatusReg’ (M\_OS\_ENTER\_CRITICAL) to disable interrupts before execution of the function.

##### 10.18.7.7.2 daucmfucos-OsIntExit-LLR-002

Requirement ID: H698-LLD-CMU-FNC-1317

The function shall do the following when the Interrupt nesting level(which is pre-decremented) Bitwise OR Multitasking lock nesting level is equal to M\_ZERO, Otherwise do nothing.

- Set os interrupt y with Os unmap table of index ready group list

- Os\_tcb\_high\_rdy with Os\_tcb\_prio\_tbl of index (addition of (bit shift of os interrupt y to left by M\_THREE) and Os unmap table of index (Os ready table of index os interrupt y))

##### 10.18.7.7.3 daucmfucos-OsIntExit-LLR-003

Requirement ID: H698-LLD-CMU-FNC-1318

The function shall perform the following operations when the ISR is complete and not locked and the task to be run next 'Os\_tcb\_high\_rdy' is not the current running task 'Os\_tcb\_cur'. Otherwise do nothing.

a) Increment context switch counter

b) call OsIntCtxSw to perform interrupt level context switch.

##### 10.18.7.7.4 daucmfucos-OsIntExit-LLR-004

Requirement ID: H698-LLD-CMU-FNC-1319

The function shall Call RestoreStatusReg (M\_OS\_EXIT\_CRITICAL) with parameter U32\_critical\_sr to restore the priority mask register after the highest priority task is ready to run or if ISR not complete( i.e. Interrupt nesting level is greater than 0 ) or ISR locked(i.e. Multitasking lock nesting level is greater than 0).

### 10.18.8 OsTimeDly

Low Level Design Details about CSU OsTimeDly will follow in the sub sections.

#### 10.18.8.1 Brief Description

OsTimeDly function allows the calling task to delay for a specific number of Clock ticks. DELAY TASK 'n' TICKS (n from 1 to 65535).

#### 10.18.8.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-012-CMU).

#### 10.18.8.3 List of global variables accessed and modified

Accessed : Os\_tcb\_cur

U32\_critical\_sr

Modified:Os\_tcb\_cur  
 U32\_critical\_sr

#### 10.18.8.4 Parameter list (Input/Output)

Inputs: T\_UWORD ticks - No of Delay Clock ticks

Outputs: None

#### 10.18.8.5 Return Value

None

#### 10.18.8.6 Other CSUs called by this CSU

OsSched

SaveStatusReg

RestoreStatusReg

#### 10.18.8.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to OsTimeDly.

##### 10.18.8.7.1 daucmfucos-OsTimeDly-LLR-001

Requirement ID: H698-LLD-CMU-FNC-1328

The function shall do the following when 'ticks' is greater than M\_ZERO.

- Send a request to uC/OS to disable all the interrupts by setting U32\_critical\_sr to return value of ‘SaveStatusReg’ (M\_OS\_ENTER\_CRITICAL)

- Set ready group list with (ready group list Bitwise AND (negation of os\_tcb\_bity of Os\_tcb\_cur)) when (Os ready table of index (os\_tcb\_y of Os\_tcb\_cur) set with Os ready table of index (os\_tcb\_y of Os\_tcb\_cur)Bitwise AND (negation of os\_tcb\_bitx of Os\_tcb\_cur)) returns M\_ZERO, otherwise do nothing.

- set os\_tcb\_dly of the TCB current task (Os\_tcb\_cur) to 'ticks' to indicate that the task is delayed.

- Send a request to uC/OS to enable all the interrupts by calling RestoreStatusReg (M\_OS\_EXIT\_CRITICAL) with parameter U32\_critical\_sr

- Call the function 'OsSched' to run the next high priority task as this task has been delayed.

##### 10.18.8.7.2 daucmfucos-OsTimeDly-LLR-002

Requirement ID: H698-LLD-CMU-FNC-1329

The function shall do nothing when 'ticks' is less than M\_ZERO.

### 10.18.9 OsTimeTick

Low Level Design Details about CSU OsTimeTick will follow in the sub sections.

#### 10.18.9.1 Brief Description

OsTimeTick function processes the Clock tick for the task or the ISR. .

Checks all the task to see if they are either waiting for time to expire(Call to OSTimeDly()) or waiting for events to occur until they timeout.

#### 10.18.9.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-012-CMU).

#### 10.18.9.3 List of global variables accessed and modified

Accessed : U32\_critical\_sr

Modified : U32\_critical\_sr

#### 10.18.9.4 Parameter list (Input/Output)

Inputs: None

Outputs: None

#### 10.18.9.5 Return Value

None

#### 10.18.9.6 Other CSUs called by this CSU

SaveStatusReg

RestoreStatusReg

#### 10.18.9.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to OsTimeTick.

##### 10.18.9.7.1 daucmfucos-OsTimeTick-LLR-001

Requirement ID: H698-LLD-CMU-FNC-1338

The function shall loop through all TCBs in TCB list till the task priority for the TCB is not equal to M\_OS\_LOWEST\_PRIO and do the following

- Send a request to uC/OS to disable all the interrupts by setting U32\_critical\_sr to return value of ‘SaveStatusReg’ (M\_OS\_ENTER\_CRITICAL).

- When os\_tcb\_dly for the TCB is not M\_ZERO do the following, otherwise do nothing.

a) Decrement the os\_tcb\_dly for the TCB.

b) Set the Ready list group with (Ready list group Bitwise OR os\_tcb\_bity of TCB) and Os ready table of index (os\_tcb\_y of TCB) with (Os ready table of index (os\_tcb\_y of TCB) Bitwise OR os\_tcb\_bitx of TCB) when

i) os\_tcb\_dly for the TCB is M\_ZERO, Otherwise do nothing

ii) os\_tcb\_stat for the TCB Bitwise AND M\_OS\_STAT\_SUSPEND is not equal to M\_OS\_STAT\_SUSPEND otherwise set os\_tcb\_dly for the TCB to M\_ONE

- Send a request to uC/OS to enable all the interrupts by calling RestoreStatusReg (M\_OS\_EXIT\_CRITICAL) with parameter U32\_critical\_sr.

- Set TCB with os\_tcb\_next of TCB.

##### 10.18.9.7.2 daucmfucos-OsTimeTick-LLR-002

Requirement ID: H698-LLD-CMU-FNC-1339

The function shall send a request to uC/OS to disable all the interrupts by setting U32\_critical\_sr to return value of ‘SaveStatusReg’ (M\_OS\_ENTER\_CRITICAL).

##### 10.18.9.7.3 daucmfucos-OsTimeTick-LLR-003

Requirement ID: H698-LLD-CMU-FNC-1340

The function shall increment the Current value of System time by one.

##### 10.18.9.7.4 daucmfucos-OsTimeTick-LLR-004

Requirement ID: H698-LLD-CMU-FNC-2056

The function shall send a request to uC/OS to enable all the interrupts by calling RestoreStatusReg (M\_OS\_EXIT\_CRITICAL) with parameter U32\_critical\_sr.

### 10.18.10 OsSemCreate

Low Level Design Details about CSU OsSemCreate will follow in the sub sections.

#### 10.18.10.1 Brief Description

OsSemCreate function creates and initializes the Semaphore.

NOTE: The initial value of Semaphore can be between 0 and 65535.

#### 10.18.10.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-012-CMU).

#### 10.18.10.3 List of global variables accessed and modified

Accessed : U32\_critical\_sr

Modified : U32\_critical\_sr

#### 10.18.10.4 Parameter list (Input/Output)

Inputs: T\_UWORD count - Semaphore count value

Outputs: None

#### 10.18.10.5 Return Value

T\_OS\_EVENT \* - returns reference to the obtained ECB on successful

creation of Semaphore otherwise returns M\_NULL.

#### 10.18.10.6 Other CSUs called by this CSU

SaveStatusReg

RestoreStatusReg

#### 10.18.10.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to OsSemCreate.

##### 10.18.10.7.1 daucmfucos-OsSemCreate-LLR-001

Requirement ID: H698-LLD-CMU-FNC-1349

The function shall set U32\_critical\_sr to return value of ‘SaveStatusReg’ (M\_OS\_ENTER\_CRITICAL) to disable interrupts.

##### 10.18.10.7.2 daucmfucos-OsSemCreate-LLR-002

Requirement ID: H698-LLD-CMU-FNC-1350

The function shall obtain an ECB from free list of ECBs.

##### 10.18.10.7.3 daucmfucos-OsSemCreate-LLR-003

Requirement ID: H698-LLD-CMU-FNC-1351

The function shall set the linked list of free ECB to point to next free list of ECB (os\_eventptr) when the free ECB pool is not empty. Otherwise do nothing.

##### 10.18.10.7.4 daucmfucos-OsSemCreate-LLR-004

Requirement ID: H698-LLD-CMU-FNC-1352

The function shall enable all the interrupts by calling RestoreStatusReg (M\_OS\_EXIT\_CRITICAL) with parameter U32\_critical\_sr.

##### 10.18.10.7.5 daucmfucos-OsSemCreate-LLR-005

Requirement ID: H698-LLD-CMU-FNC-1353

The function shall do the following when the reference to the selected ECB is not M\_NULL, otherwise return M\_NULL

- Set the desired initial count (count) for the Semaphore into os\_eventcnt of obtained ECB.

- Set the os\_eventgrp with M\_HEX2\_ZERO

- Set all the indices of os\_event\_tbl of the obtained ECB (size is M\_MAX\_TSK\_WAIT\_EVNT) to M\_HEX2\_ZERO .

- Return the obtained ECB.

### 10.18.11 OsSemPend

Low Level Design Details about CSU OsSemPend will follow in the sub sections.

#### 10.18.11.1 Brief Description

OsSemPend function waits on a Semaphore.

#### 10.18.11.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-012-CMU).

#### 10.18.11.3 List of global variables accessed and modified

Accessed : Os\_tcb\_cur

U32\_critical\_sr

Modified : Os\_tcb\_cur

U32\_critical\_sr

#### 10.18.11.4 Parameter list (Input/Output)

Inputs: T\_OS\_EVENT \*p\_event - Pointer to the desired Semaphore's Event Control Block

T\_UWORD timeout - Time in clock ticks to wait for the resource.

If 0, the task will wait until the resource becomes available or the event occurs.

Outputs: T\_OS\_EVENT \*p\_event - Pointer to the desired Semaphore's Event Control Block

T\_UBYTE \* error - Pointer to error message.

- Set to M\_OS\_NO\_ERR when the Semaphore was available.

- Set to M\_OS\_TIMEOUT when the Semaphore was not signaled within the specified timeout

#### 10.18.11.5 Return Value

None

#### 10.18.11.6 Other CSUs called by this CSU

OsSched

SaveStatusReg

RestoreStatusReg

#### 10.18.11.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to OsSemPend.

##### 10.18.11.7.1 daucmfucos-OsSemPend-LLR-001

Requirement ID: H698-LLD-CMU-FNC-1362

The function shall set U32\_critical\_sr to return value of ‘SaveStatusReg’ (M\_OS\_ENTER\_CRITICAL) to disable interrupts.

##### 10.18.11.7.2 daucmfucos-OsSemPend-LLR-002

Requirement ID: H698-LLD-CMU-FNC-1363

The function shall do the following when os\_eventcnt of p\_event is greater than M\_ZERO(Semaphore is positive, resource is available)

- Decrement the os\_eventcnt of p\_event by one

- Send a request to uC/OS to enable all the interrupts by calling RestoreStatusReg (M\_OS\_EXIT\_CRITICAL) with parameter U32\_critical\_sr

- Set the error message error to M\_OS\_NO\_ERR.

##### 10.18.11.7.3 daucmfucos-OsSemPend-LLR-003

Requirement ID: H698-LLD-CMU-FNC-1364

The function shall do the following when os\_eventcnt of p\_event is less than or equal to M\_ZERO (Semaphore is negative, resource is not available)

- Set os\_tcb\_stat of Os\_tcb\_cur with os\_tcb\_stat of Os\_tcb\_cur Bitwise OR M\_OS\_STAT\_SEM

- Set os\_tcb\_dly of Os\_tcb\_cur with time out value

- Set os\_tcb\_eventptr of Os\_tcb\_cur with p\_event

- Set Os ready group with Os ready group Bitwise AND (negation of os\_tcb\_bity of Os\_tcb\_cur) when (Os ready table of index (os\_tcb\_y of Os\_tcb\_cur) is set with Os ready table of index (os\_tcb\_y of Os\_tcb\_cur) Bitwise AND(negation of os\_tcb\_bitx of Os\_tcb\_cur)) returns M\_ZERO, Otherwise do nothing

- Put the task in waiting list by doing the following

a. Set os\_event\_tbl of p\_event for index (os\_tcb\_y of Os\_tcb\_cur) with os\_event\_tbl of p\_event

for index (os\_tcb\_y of Os\_tcb\_cur)Bitwise OR os\_tcb\_bitx of Os\_tcb\_cur.

b. Set os\_eventgrp of p\_event with os\_eventgrp of p\_event Bitwise OR os\_tcb\_bity of

Os\_tcb\_cur.

- Send a request to uC/OS to enable all the interrupts by calling RestoreStatusReg (M\_OS\_EXIT\_CRITICAL) with parameter U32\_critical\_sr

- Call the function 'OsSched' to ready the next highest priority task

- Send a request to uC/OS to disable all the interrupts by calling SaveStatusReg (M\_OS\_ENTRY\_CRITICAL)

- Do the following when (os\_tcb\_stat of Os\_tcb\_cur Bitwise AND M\_OS\_STAT\_SEM) returns M\_OS\_STAT\_SEM

1. Set os\_eventgrp of p\_event with os\_eventgrp of p\_event Bitwise AND (negation of os\_tcb\_bity of Os\_tcb\_cur) when (os\_event\_tbl of p\_event for index (os\_tcb\_y of Os\_tcb\_cur)is set with os\_event\_tbl of p\_event for index (os\_tcb\_y of Os\_tcb\_cur)

Bitwise AND (negation of os\_tcb\_bitx of Os\_tcb\_cur)) returns M\_ZERO, Otherwise do nothing

b. Set os\_tcb\_stat of Os\_tcb\_cur with 'M\_OS\_STAT\_RDY'

c. Set os\_tcb\_eventptr Os\_tcb\_cur to M\_NULL

d. Send a request to uC/OS to enable all the interrupts by calling

RestoreStatusReg (M\_OS\_EXIT\_CRITICAL) with parameter U32\_critical\_sr

e. Set the error message pointer to M\_OS\_TIMEOUT

- Do the following when (os\_tcb\_stat of Os\_tcb\_cur Bitwise AND M\_OS\_STAT\_SEM) returns other than

M\_OS\_STAT\_SEM

a. Set os\_tcb\_eventptr of Os\_tcb\_cur to M\_NULL

b. Send a request to uC/OS to enable all the interrupts by calling RestoreStatusReg (M\_OS\_EXIT\_CRITICAL) with parameter U32\_critical\_sr

c. Set the error message error to M\_OS\_NO\_ERR.

### 10.18.12 OsSemPost

Low Level Design Details about CSU OsSemPost will follow in the sub sections.

#### 10.18.12.1 Brief Description

OsSemPost function posts a Semaphore.

#### 10.18.12.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-012-CMU).

#### 10.18.12.3 List of global variables accessed and modified

Accessed : U32\_critical\_sr

Os\_tcb\_prio\_tbl

Modified : U32\_critical\_sr

#### 10.18.12.4 Parameter list (Input/Output)

Inputs: T\_OS\_EVENT \*p\_event - Pointer to the desired Semaphore's Event

Control Block

Outputs: T\_OS\_EVENT \*p\_event - Pointer to the desired Semaphore's Event

Control Block

#### 10.18.12.5 Return Value

T\_UBYTE - returns the error code

M\_OS\_NO\_ERR - a) Semaphore posted successfully.

b) Semaphore value within limit and no task waiting

for Semaphore.

M\_OS\_SEM\_OVF - overflow of Semaphore value

#### 10.18.12.6 Other CSUs called by this CSU

OsSched

RterrDisplayErrorId

SaveStatusReg

RestoreStatusReg

RterrForever

#### 10.18.12.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to OsSemPost.

##### 10.18.12.7.1 daucmfucos-OsSemPost-LLR-001

Requirement ID: H698-LLD-CMU-FNC-1373

The function shall set U32\_critical\_sr to return value of ‘SaveStatusReg’ (M\_OS\_ENTER\_CRITICAL) to disable interrupts.

##### 10.18.12.7.2 daucmfucos-OsSemPost-LLR-002

Requirement ID: H698-LLD-CMU-FNC-1374

The function shall do the following when os\_eventgrp of p\_event not equal to M\_ZERO

- Get the index of highest priority task pending on event.i,e.

Set the event table y to Os unmap table with index as os\_eventgrp of p\_event

Set the bit y to Os map table with index as event table y

Set the unmap table x to Os unmap table with index as (os\_event\_tbl of p\_event with index event table y)

Set the bit x to Os map table with index unmap table x

Set the TCB index to (addition of (bit shift of event table y to left by M\_THREE) and unmap table x )

- Remove the pending task from the list of tasks waiting for event to occur by setting the os\_eventgrp of p\_event to os\_eventgrp of p\_event Bitwise AND (negation of bit y) when (os\_event\_tbl of p\_event with index event table y is set to os\_event\_tbl of p\_event with index event table y Bitwise AND (negation of bit x)) returns M\_ZERO, otherwise do nothing

- Set TCB pointer to Os\_tcb\_prio\_tbl of index (TCB index)

- Set os\_tcb\_dly of TCB pointer to M\_ZERO to prevent OsTimeTick() from readying this task

- Set Event ptr os\_tcb\_eventptr of the TCB pointer to M\_NULL

- Set os\_tcb\_stat of the TCB pointer to os\_tcb\_stat of the TCB pointer Bitwise AND ( negation of M\_OS\_STAT\_SEM)

- Set the following when os\_tcb\_stat of TCB pointer is M\_OS\_STAT\_RDY,otherwise do nothing.

a. Set ready group list to ready group list Bitwise OR bit y

b. Set Os ready table of index event table y to Os ready table of index event table y Bitwise OR bit x

- Send a request to uC/OS to enable all the interrupts by calling RestoreStatusReg (M\_OS\_EXIT\_CRITICAL) with parameter U32\_critical\_sr

- Call the function 'OsSched' to ready the next highest priority task

- Return with M\_OS\_NO\_ERR.

##### 10.18.12.7.3 daucmfucos-OsSemPost-LLR-003

Requirement ID: H698-LLD-CMU-FNC-1375

The function shall do the following when os\_eventgrp of p\_event is M\_ZERO and os\_eventcnt of p\_event is less than M\_MAX\_SEMPHORE

a) Increment os\_eventcnt of p\_event by one

b) Send a request to uC/OS to enable all the interrupts by calling RestoreStatusReg (M\_OS\_EXIT\_CRITICAL) with parameter U32\_critical\_sr

c) Return with M\_OS\_NO\_ERR.

##### 10.18.12.7.4 daucmfucos-OsSemPost-LLR-004

Requirement ID: H698-LLD-CMU-FNC-1376

The function shall do the following when os\_eventgrp of p\_event is M\_ZERO and os\_eventcnt of p\_event is greater than or equal to M\_MAX\_SEMPHORE

a) Send a request to uC/OS to enable all the interrupts by calling RestoreStatusReg (M\_OS\_EXIT\_CRITICAL) with parameter U32\_critical\_sr

b) Call the RterrDisplayErrorId with parameter M\_APP\_ID, RterrForever, OS\_KERNEL\_ERR Bitwise OR with RT\_ERROR\_11 to display the input error

c) Return with M\_OS\_SEM\_OVF.

## 10.19 daucmfwdog

daucmfwdog CSC contains implementation of routines for accessing the Watch dog Timer.

### 10.19.1 WdogInit

Low Level Design Details about CSU WdogInit will follow in the sub sections.

#### 10.19.1.1 Brief Description

WdogInit function calculates the reload value for the Independent Watchdog timer, initializes the prescaler divider to 64, writes reload value to the reload register,

enables the Watchdog register and reloads Watchdog Counter.

#### 10.19.1.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-012-CMU).

#### 10.19.1.3 List of global variables accessed and modified

Accessed : None

Modified : None

#### 10.19.1.4 Parameter list (Input/Output)

Inputs : None

Outputs : None

#### 10.19.1.5 Return Value

None

#### 10.19.1.6 Other CSUs called by this CSU

IwdgWriteAccessCmd

IwdgSetPrescaler

IwdgSetReload

IwdgEnable

IwdgReloadCounter

#### 10.19.1.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to WdogInit

##### 10.19.1.7.1 daucmfwdog-WdogInit-LLR-001

Requirement ID: H698-LLD-CMU-FNC-1386

The function shall calculate the reload value for the Independent Watchdog timer as follows, the product of the Watchdog Timeout (M\_WDOG\_TIMEOUT\_MS) and Watchdog Frequency(M\_WDOG\_FREQ) scaled by the product of Watchdog prescaler (M\_GET\_PRESCALER\_4\_FOR\_WD left shift by M\_WDOG\_PRESCALER) and M\_WATCHDOG\_FREQ\_KHZ\_SCALE.

##### 10.19.1.7.2 daucmfwdog-WdogInit-LLR-002

Requirement ID: H698-LLD-CMU-FNC-1387

The function shall set the calculated reload value to M\_MAX\_RELOAD\_VALUE when the reload value is greater than M\_MAX\_RELOAD\_VALUE. Otherwise do nothing.

##### 10.19.1.7.3 daucmfwdog-WdogInit-LLR-003

Requirement ID: H698-LLD-CMU-FNC-1388

The function shall Call IwdgWriteAccessCmd with parameter M\_IWDG\_WRITEACCESS\_ENABLE to enable write access to the prescaler and reload registers, before writing into the registers

##### 10.19.1.7.4 daucmfwdog-WdogInit-LLR-004

Requirement ID: H698-LLD-CMU-FNC-1389

The function shall call IwdgSetPrescaler with parameter M\_WDOG\_PRESCALER to set the IWDG Prescaler value.

##### 10.19.1.7.5 daucmfwdog-WdogInit-LLR-005

Requirement ID: H698-LLD-CMU-FNC-1390

The function shall call IwdgSetReload with the calculated reload value as parameter to write reload value to the reload register.

##### 10.19.1.7.6 daucmfwdog-WdogInit-LLR-006

Requirement ID: H698-LLD-CMU-FNC-1391

The function shall Call IwdgWriteAccessCmd with parameter M\_IWDG\_WRITEACCESS\_DISABLE to disable write access of prescaler and reload registers, after write has been completed.

##### 10.19.1.7.7 daucmfwdog-WdogInit-LLR-007

Requirement ID: H698-LLD-CMU-FNC-1392

The function shall call IwdgEnable to enable Watchdog Timer.

##### 10.19.1.7.8 daucmfwdog-WdogInit-LLR-008

Requirement ID: H698-LLD-CMU-FNC-1393

The function shall call IwdgReloadCounter to reload Watchdog Counter.

### 10.19.2 WdogKickWatchDog

Low Level Design Details about CSU WdogKickWatchDog will follow in the sub sections.

#### 10.19.2.1 Brief Description

WdogKickWatchDog function reloads Watchdog Counter.

#### 10.19.2.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-012-CMU).

#### 10.19.2.3 List of global variables accessed and modified

Accessed : None

Modified : None

#### 10.19.2.4 Parameter list (Input/Output)

Inputs : None

Outputs : None

#### 10.19.2.5 Return Value

None.

#### 10.19.2.6 Other CSUs called by this CSU

IwdgReloadCounter

#### 10.19.2.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to WdogKickWatchDog

##### 10.19.2.7.1 daucmfwdog-WdogKickWatchDog-LLR-001

Requirement ID: H698-LLD-CMU-FNC-1402

The function shall call IwdgReloadCounter to reload the Watchdog Counter.

## 10.20 daucmfxram

daucmfxram CSC contains controls access to NOR and NVRAM devices. And Initialization of NOR and XRAM.

### 10.20.1 XramInit

Low Level Design Details about CSU XramInit will follow in the sub sections.

#### 10.20.1.1 Brief Description

XramInit function Initialize NOR and XRAM.

#### 10.20.1.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-012-CMU).

#### 10.20.1.3 List of global variables accessed and modified

Accessed : None

Modified : None

#### 10.20.1.4 Parameter list (Input/Output)

Inputs : None

Outputs : None

#### 10.20.1.5 Return Value

None

#### 10.20.1.6 Other CSUs called by this CSU

RccAhb1PeriphClockCmd

RccAhb3PeriphClockCmd

GpioPinAFConfig

GpioInit

GpioResetBits

FsmcNorSramInit

FsmcNorSramCmd

#### 10.20.1.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to XramInit.

##### 10.20.1.7.1 daucmfxram-XramInit-LLR-001

Requirement ID: H698-LLD-CMU-FNC-1412

The XramInit function shall call RccAhb1PeriphClockCmd function with parameters (M\_RCC\_AHB1PERIPH\_GPIOD Bitwise OR M\_RCC\_AHB1PERIPH\_GPIOE Bitwise OR M\_RCC\_AHB1PERIPH\_GPIOF Bitwise OR M\_RCC\_AHB1PERIPH\_GPIOG, ENABLE) to enable Enable GPIOs clock .Any of the following clock is enabled.

* GPIOD
* GPIOE
* GPIOF
* GPIOG

##### 10.20.1.7.2 daucmfxram-XramInit-LLR-002

Requirement ID: H698-LLD-CMU-FNC-1413

The XramInit function shall call RccAhb3PeriphClockCmd function with parameters (M\_RCC\_AHB3PERIPH\_FSMC, ENABLE) to enable FSMC clock.

##### 10.20.1.7.3 daucmfxram-XramInit-LLR-003

Requirement ID: H698-LLD-CMU-FNC-1414

The XramInit function shall call GpioPinAFConfig function to enable M\_GPIOD configuration (Config for the Data bus (D2, D3,D13,D14,D15,D0,D1),(A16,A17,A18),clock, Output Enable, Write Enable, Wait, Not Enable1). The following Pins are enabled.

- Call the function GpioPinAFConfig with parameter

(M\_GPIOD, M\_GPIO\_PINSOURCE0, M\_GPIO\_AF\_FSMC)

- Call the function GpioPinAFConfig with parameter

(M\_GPIOD, M\_GPIO\_PINSOURCE1, M\_GPIO\_AF\_FSMC)

- Call the function GpioPinAFConfig with parameter

(M\_GPIOD, M\_GPIO\_PINSOURCE3, M\_GPIO\_AF\_FSMC)

- Call the function GpioPinAFConfig with parameter

(M\_GPIOD, M\_GPIO\_PINSOURCE4, M\_GPIO\_AF\_FSMC)

- Call the function GpioPinAFConfig with parameter

(M\_GPIOD, M\_GPIO\_PINSOURCE5, M\_GPIO\_AF\_FSMC)

- Call the function GpioPinAFConfig with parameter

(M\_GPIOD, M\_GPIO\_PINSOURCE6, M\_GPIO\_AF\_FSMC)

- Call the function GpioPinAFConfig with parameter

(M\_GPIOD, M\_GPIO\_PINSOURCE7, M\_GPIO\_AF\_FSMC)

- Call the function GpioPinAFConfig with parameter

(M\_GPIOD, M\_GPIO\_PINSOURCE8, M\_GPIO\_AF\_FSMC)

- Call the function GpioPinAFConfig with parameter

(M\_GPIOD, M\_GPIO\_PINSOURCE9, M\_GPIO\_AF\_FSMC)

- Call the function GpioPinAFConfig with parameter

(M\_GPIOD, M\_GPIO\_PINSOURCE10, M\_GPIO\_AF\_FSMC)

- Call the function GpioPinAFConfig with parameter

(M\_GPIOD, M\_GPIO\_PINSOURCE11, M\_GPIO\_AF\_FSMC)

- Call the function GpioPinAFConfig with parameter

(M\_GPIOD, M\_GPIO\_PINSOURCE12, M\_GPIO\_AF\_FSMC)

- Call the function GpioPinAFConfig with parameter

(M\_GPIOD, M\_GPIO\_PINSOURCE13, M\_GPIO\_AF\_FSMC)

- Call the function GpioPinAFConfig with parameter

(M\_GPIOD, M\_GPIO\_PINSOURCE14, M\_GPIO\_AF\_FSMC)

- Call the function GpioPinAFConfig with parameter

(M\_GPIOD, M\_GPIO\_PINSOURCE15, M\_GPIO\_AF\_FSMC)

##### 10.20.1.7.4 daucmfxram-XramInit-LLR-004

Requirement ID: H698-LLD-CMU-FNC-1415

The function shall Set the members of Init structure to configure GPIO pins as alternate function output pins as follows:

- Set the gpio\_pin of Init structure to bitwise OR of M\_GPIO\_PIN\_0, M\_GPIO\_PIN\_1, M\_GPIO\_PIN\_3,

M\_GPIO\_PIN\_4, M\_GPIO\_PIN\_5, M\_GPIO\_PIN\_6, M\_GPIO\_PIN\_7, M\_GPIO\_PIN\_8,

M\_GPIO\_PIN\_9, M\_GPIO\_PIN\_10, M\_GPIO\_PIN\_11, M\_GPIO\_PIN\_12,

M\_GPIO\_PIN\_13, M\_GPIO\_PIN\_14, M\_GPIO\_PIN\_15

##### 10.20.1.7.5 daucmfxram-XramInit-LLR-005

Requirement ID: H698-LLD-CMU-FNC-1416

The XramInit function shall enable the following by accessing members of GpioInit.

- Set the gpio\_mode of Init structure to GPIO\_MODE\_AF

- Set the gpio\_speed of Init structure to GPIO\_SPEED\_100MHZ

- Set the gpio\_otype of Init structure to GPIO\_OTYPE\_PP

- Set the gpio\_pupd of Init structure to GPIO\_PUPD\_NOPULL

##### 10.20.1.7.6 daucmfxram-XramInit-LLR-006

Requirement ID: H698-LLD-CMU-FNC-1417

The XramInit function shall call GpioInit function with parameters (M\_GPIOD, reference of gpio Init structure) to enable M\_GPIOD pin.

##### 10.20.1.7.7 daucmfxram-XramInit-LLR-007

Requirement ID: H698-LLD-CMU-FNC-1418

The XramInit function shall call GpioPinAFConfig function to enable M\_GPIOE configuration. The following Pins are enabled(BL0,BL1,A19 to A23, D4 to D12).

- Call the function GpioPinAFConfig with parameter

(M\_GPIOE, M\_GPIO\_PINSOURCE0, M\_GPIO\_AF\_FSMC)

- Call the function GpioPinAFConfig with parameter

(M\_GPIOE, M\_GPIO\_PINSOURCE1, M\_GPIO\_AF\_FSMC)

- Call the function GpioPinAFConfig with parameter

(M\_GPIOE, M\_GPIO\_PINSOURCE2, M\_GPIO\_AF\_FSMC)

- Call the function GpioPinAFConfig with parameter

(M\_GPIOE, M\_GPIO\_PINSOURCE3, M\_GPIO\_AF\_FSMC)

- Call the function GpioPinAFConfig with parameter

(M\_GPIOE, M\_GPIO\_PINSOURCE4, M\_GPIO\_AF\_FSMC)

- Call the function GpioPinAFConfig with parameter

(M\_GPIOE, M\_GPIO\_PINSOURCE5, M\_GPIO\_AF\_FSMC)

- Call the function GpioPinAFConfig with parameter

(M\_GPIOE, M\_GPIO\_PINSOURCE6, M\_GPIO\_AF\_FSMC)

- Call the function GpioPinAFConfig with parameter

(M\_GPIOE, M\_GPIO\_PINSOURCE7, M\_GPIO\_AF\_FSMC)

- Call the function GpioPinAFConfig with parameter

(M\_GPIOE, M\_GPIO\_PINSOURCE8, M\_GPIO\_AF\_FSMC)

- Call the function GpioPinAFConfig with parameter

(M\_GPIOE, M\_GPIO\_PINSOURCE9, M\_GPIO\_AF\_FSMC)

- Call the function GpioPinAFConfig with parameter

(M\_GPIOE, M\_GPIO\_PINSOURCE10, M\_GPIO\_AF\_FSMC)

- Call the function GpioPinAFConfig with parameter

(M\_GPIOE, M\_GPIO\_PINSOURCE11, M\_GPIO\_AF\_FSMC)

- Call the function GpioPinAFConfig with parameter

(M\_GPIOE, M\_GPIO\_PINSOURCE12, M\_GPIO\_AF\_FSMC)

- Call the function GpioPinAFConfig with parameter

(M\_GPIOE, M\_GPIO\_PINSOURCE13, M\_GPIO\_AF\_FSMC)

- Call the function GpioPinAFConfig with parameter

(M\_GPIOE, M\_GPIO\_PINSOURCE14, M\_GPIO\_AF\_FSMC)

- Call the function GpioPinAFConfig with parameter

(M\_GPIOE, M\_GPIO\_PINSOURCE15, M\_GPIO\_AF\_FSMC)

- Set the gpio\_pin of Init structure to bitwise OR of M\_GPIO\_PIN\_0, M\_GPIO\_PIN\_1, M\_GPIO\_PIN\_2,

M\_GPIO\_PIN\_3,M\_GPIO\_PIN\_4,M\_GPIO\_PIN\_5,M\_GPIO\_PIN\_6,M\_GPIO\_PIN\_7, M\_GPIO\_PIN\_8, M\_GPIO\_PIN\_9, M\_GPIO\_PIN\_10, M\_GPIO\_PIN\_11, M\_GPIO\_PIN\_12,

M\_GPIO\_PIN\_13, M\_GPIO\_PIN\_14, M\_GPIO\_PIN\_15

##### 10.20.1.7.8 daucmfxram-XramInit-LLR-008

Requirement ID: H698-LLD-CMU-FNC-1419

The XramInit function shall call GpioInit function with parameters (M\_GPIOE , reference of gpio Init structure) to enable M\_GPIOE pin.

##### 10.20.1.7.9 daucmfxram-XramInit-LLR-009

Requirement ID: H698-LLD-CMU-FNC-1420

The XramInit function shall call GpioPinAFConfig function to enable M\_GPIOF configuration. The following Pins are enabled.(A0 to A9)

- Call the function GpioPinAFConfig with parameter

(M\_GPIOF, M\_GPIO\_PINSOURCE0, M\_GPIO\_AF\_FSMC)

- Call the function GpioPinAFConfig with parameter

(M\_GPIOF, M\_GPIO\_PINSOURCE1, M\_GPIO\_AF\_FSMC)

- Call the function GpioPinAFConfig with parameter

(M\_GPIOF, M\_GPIO\_PINSOURCE2, M\_GPIO\_AF\_FSMC)

- Call the function GpioPinAFConfig with parameter

(M\_GPIOF, M\_GPIO\_PINSOURCE3, M\_GPIO\_AF\_FSMC)

- Call the function GpioPinAFConfig with parameter

(M\_GPIOF, M\_GPIO\_PINSOURCE4, M\_GPIO\_AF\_FSMC)

- Call the function GpioPinAFConfig with parameter

(M\_GPIOF, M\_GPIO\_PINSOURCE5, M\_GPIO\_AF\_FSMC)

- Call the function GpioPinAFConfig with parameter

(M\_GPIOF, M\_GPIO\_PINSOURCE12, M\_GPIO\_AF\_FSMC)

- Call the function GpioPinAFConfig with parameter

(M\_GPIOF, M\_GPIO\_PINSOURCE13, M\_GPIO\_AF\_FSMC)

- Call the function GpioPinAFConfig with parameter

(M\_GPIOF, M\_GPIO\_PINSOURCE14, M\_GPIO\_AF\_FSMC)

- Call the function GpioPinAFConfig with parameter

(M\_GPIOF, M\_GPIO\_PINSOURCE915 M\_GPIO\_AF\_FSMC)

##### 10.20.1.7.10 daucmfxram-XramInit-LLR-010

Requirement ID: H698-LLD-CMU-FNC-1421

The XramInit function shall enable Gpio Pin by accessing members of GpioInit. Any of the following pins are enabled.

- Set the gpio\_pin of Init structure to bitwise OR of M\_GPIO\_PIN\_0, M\_GPIO\_PIN\_1, M\_GPIO\_PIN\_2,

M\_GPIO\_PIN\_3,M\_GPIO\_PIN\_4,M\_GPIO\_PIN\_5,M\_GPIO\_PIN\_12,M\_GPIO\_PIN\_13,M\_GPIO\_PIN\_14, M\_GPIO\_PIN\_15

##### 10.20.1.7.11 daucmfxram-XramInit-LLR-011

Requirement ID: H698-LLD-CMU-FNC-1422

The XramInit function shall call GpioInit function with parameters (M\_GPIOF , reference of gpio Init structure) to enable M\_GPIOF pin.

##### 10.20.1.7.12 daucmfxram-XramInit-LLR-012

Requirement ID: H698-LLD-CMU-FNC-1423

The XramInit function shall call GpioPinAFConfig function to enable M\_GPIOG configuration pins.(A10 to A15,A24,NE2)

- Call the function GpioPinAFConfig with parameter

(M\_GPIOG, M\_GPIO\_PINSOURCE0, M\_GPIO\_AF\_FSMC)

- Call the function GpioPinAFConfig with parameter

(M\_GPIOG, M\_GPIO\_PINSOURCE1, M\_GPIO\_AF\_FSMC)

- Call the function GpioPinAFConfig with parameter

(M\_GPIOG, M\_GPIO\_PINSOURCE2, M\_GPIO\_AF\_FSMC)

- Call the function GpioPinAFConfig with parameter

(M\_GPIOG, M\_GPIO\_PINSOURCE3, M\_GPIO\_AF\_FSMC)

- Call the function GpioPinAFConfig with parameter

(M\_GPIOG, M\_GPIO\_PINSOURCE4, M\_GPIO\_AF\_FSMC)

- Call the function GpioPinAFConfig with parameter

(M\_GPIOG, M\_GPIO\_PINSOURCE5, M\_GPIO\_AF\_FSMC)

- Call the function GpioPinAFConfig with parameter

(M\_GPIOG, M\_GPIO\_PINSOURCE9, M\_GPIO\_AF\_FSMC)

- Call the function GpioPinAFConfig with parameter

(M\_GPIOG, M\_GPIO\_PINSOURCE13, M\_GPIO\_AF\_FSMC)

##### 10.20.1.7.13 daucmfxram-XramInit-LLR-013

Requirement ID: H698-LLD-CMU-FNC-1424

The XramInit function shall enable Gpio Pin by accessing members of GpioInit. Any of the following pins are enabled.

- Set the gpio\_pin of Init structure to bitwise OR of M\_GPIO\_PIN\_0, M\_GPIO\_PIN\_1, M\_GPIO\_PIN\_2,

M\_GPIO\_PIN\_3,M\_GPIO\_PIN\_4,M\_GPIO\_PIN\_5,M\_GPIO\_PIN\_9,M\_GPIO\_PIN\_13

##### 10.20.1.7.14 daucmfxram-XramInit-LLR-014

Requirement ID: H698-LLD-CMU-FNC-1425

The XramInit function shall enable the following by accessing members of GpioInit.

* Gpioint is called with parameters (M\_GPIOG, reference of gpio Init structure) by the function to enable M\_GIPOG.
* Gpio pin is enabled (M\_GPIO\_PIN\_14).
* Gpio speed is GPIO\_SPEED\_100MHZ
* Gpio mode is GPIO\_MODE\_OUT.
* Gpio operational type is GPIO\_OTYPE\_PP.
* Gpio pupd is GPIO\_PUPD\_NOPULL
* Gpioint is called with parameters (M\_GPIOG, reference of GpioInit structure) by the function to enable M\_GIPOG.

##### 10.20.1.7.15 daucmfxram-XramInit-LLR-015

Requirement ID: H698-LLD-CMU-FNC-1426

The XramInit function shall call GpioResetBits function to reset the 14th bit(M\_GPIO\_PIN\_14) of M\_GPIOG.

##### 10.20.1.7.16 daucmfxram-XramInit-LLR-016

Requirement ID: H698-LLD-CMU-FNC-1427

The XramInit function shall do the following :

- Set the following for the configuration of FSMC for NVRAM

* Address setup time to ” M\_NVRAM\_ADDR\_SETUPTIME” .
* Address Hold time to “M\_ZERO”.
* Data setuptime to “M\_NVRAM\_DATA\_SETUPTIME”.
* Bus turn around duration to “M\_BUS\_TURNAROUND\_DURATION”.
* Clock division to “M\_ZERO”.
* Data latency to “M\_ZERO”.
* Access mode to “M\_FSMC\_ACCESSMODE\_A”.

- Set the following for the sram init structure

* FSMC bank to “M\_FSMC\_BANK1\_NORSRAM1”.
* FSMC data address mux to “M\_FSMC\_DATAADDRESSMUX\_DISABLE”.
* FSMC memory type to “M\_FSMC\_MEMORYTYPE\_PSRAM”.
* FSMC memory data Width to “M\_FSMC\_MEMORYDATAWIDTH\_16B”.
* FSMC Burst access mode to “M\_FSMC\_BURSTACCESSMODE\_DISABLE”.
* FSMC asynchronous wait to “M\_FSMC\_ASYNCHRONOUSWAIT\_DISABLE”.
* FSMC wait signal polarity to “M\_FSMC\_WAITSIGNALPOLARITY\_LOW”.
* FSMC wrap mode to “M\_FSMC\_WRAPMODE\_DISABLE”.
* FSMC wait signal active to “M\_FSMC\_WAIT\_SIG\_ACTIVE\_BEF\_WAIT”.
* FSMC write operation to “M\_FSMC\_WRITEOPERATION\_ENABLE”.
* FSMC wait signal to “M\_FSMC\_WAITSIGNAL\_DISABLE”
* FSMC Extend Mode to “M\_FSMC\_EXTENDEDMODE\_DISABLE”.
* FSMC Write Burst to “M\_FSMC\_WRITEBURST\_DISABLE”.
* FSMC read write timing to address of NVRAM structure
* FSMC write timing to address of NVRAM structure

##### 10.20.1.7.17 daucmfxram-XramInit-LLR-017

Requirement ID: H698-LLD-CMU-FNC-1428

The XramInit function shall call FsmcNorSramInit function with parameter as address of sram init structure

##### 10.20.1.7.18 daucmfxram-XramInit-LLR-018

Requirement ID: H698-LLD-CMU-FNC-1429

The XramInit function shall call FsmcNorSramCmd function with parameters (M\_FSMC\_BANK1\_NORSRAM1, ENABLE) to Enable first block of bank 1 and call PbitNvmReadWriteTest function to perform NVM read/write test.

##### 10.20.1.7.19 daucmfxram-XramInit-LLR-019

Requirement ID: H698-LLD-CMU-FNC-2327

The function shall do the following:

1.Calls InitNvram function when b\_nvm\_read\_write\_failed of bitwise\_status of power\_on of Bit\_status is equal to FALSE

##### 10.20.1.7.20 IsOlder

Low Level Design Details about CSU IsOlder will follow in the sub sections.

###### 10.20.1.7.20.1 Brief Description

Compares two timestamps and returns TRUE if time\_1 is older than time\_2.

###### 10.20.1.7.20.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-012-CMU).

###### 10.20.1.7.20.3 List of global variables accessed and modified

Accessed : None

Modified : None

###### 10.20.1.7.20.4 Parameter list (Input/Output)

Inputs : T\_TIMESTAMP \*p\_time\_1 - IN timestamp1.

T\_TIMESTAMP \*p\_time\_2 - IN timestamp2.

Outputs : None

###### 10.20.1.7.20.5 Return Value

None

###### 10.20.1.7.20.6 Other CSUs called by this CSU

None

###### 10.20.1.7.20.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to IsOlder.

10.20.1.7.20.7.1 daucmfxram-IsOlder-LLR-001

Requirement ID: H698-LLD-CMU-FNC-2336

The function shall do the following:

a)when u8\_year of p\_time\_1 is  not equal to u8\_year of p\_time\_2.

Set bt1\_is\_older to  (substraction of p\_time\_1 of u8\_year and  p\_time\_2 of u8\_year) greater than equal to MAX\_YEARS

b)Set bt1\_is\_older is not equal to bt1\_is\_older

when u8\_year of p\_time\_1  is equal to M\_ZERO AND  u8\_year of p\_time\_2  is equal to MAX\_YEARS

10.20.1.7.20.7.2 daucmfxram-IsOlder-LLR-002

Requirement ID: H698-LLD-CMU-FNC-2337

The function shall do the following:

a) Set bt1\_is\_older to  (substraction of u8\_month of p\_time\_1  and u8\_month of p\_time\_2 ) greater than  MAX\_MONTHS when u8\_month of p\_time\_1 is not equal to u8\_month of p\_time\_2 .

10.20.1.7.20.7.3 daucmfxram-IsOlder-LLR-003

Requirement ID: H698-LLD-CMU-FNC-2338

The function shall do the following:

a) Set bt1\_is\_older to (substraction of u8\_day of p\_time\_1  and u8\_day of p\_time\_2 ) greater than MAX\_DAYS when  u8\_day of p\_time\_1 is not equal to u8\_day of p\_time\_2 .

10.20.1.7.20.7.4 daucmfxram-IsOlder-LLR-004

Requirement ID: H698-LLD-CMU-FNC-2339

The function shall do the following:

a)Set bt1\_is\_older to (substraction of u8\_hour of p\_time\_1  and u8\_hour of p\_time\_2) greater than MAX\_HOURS when u8\_hour of p\_time\_1 is not equal to  u8\_hour of p\_time\_2.

10.20.1.7.20.7.5 daucmfxram-IsOlder-LLR-005

Requirement ID: H698-LLD-CMU-FNC-2340

The function shall do the following:

a)Set bt1\_is\_older to (substraction of u8\_minute of p\_time\_1  and u8\_minute of p\_time\_2 ) greater than MAX\_MINUTES when  u8\_minute of p\_time\_1 is not equal to u8\_minute of p\_time\_2.

10.20.1.7.20.7.6 daucmfxram-IsOlder-LLR-006

Requirement ID: H698-LLD-CMU-FNC-2341

The function shall do the following:

a)Set bt1\_is\_older to (substraction of u8\_second of p\_time\_1 and u8\_second of p\_time\_2) greater than MAX\_SECONDS when u8\_second of p\_time\_1 is not equal to u8\_second of p\_time\_2 .

10.20.1.7.20.7.7 daucmfxram-IsOlder-LLR-007

Requirement ID: H698-LLD-CMU-FNC-2342

The function shall return bt1\_is\_older

##### 10.20.1.7.21 InitNvram

Low Level Design Details about CSU InitNvram will follow in the sub sections.

###### 10.20.1.7.21.1 Brief Description

Initialize the NVM area calculated CRC, time, and validity status.

###### 10.20.1.7.21.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-012-CMU).

###### 10.20.1.7.21.3 List of global variables accessed and modified

Accessed : None

Modified : None

###### 10.20.1.7.21.4 Parameter list (Input/Output)

Inputs : None

Outputs : None

###### 10.20.1.7.21.5 Return Value

None

###### 10.20.1.7.21.6 Other CSUs called by this CSU

CrcResetDr

CrcCalcBlockCrc

IsOlder

HwCopy

RterrDisplayErrorId

RterrForever

###### 10.20.1.7.21.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to XramInit InitNvram.

10.20.1.7.21.7.1 daucmfxram-InitNvram-LLR-001

Requirement ID: H698-LLD-CMU-FNC-2359

The function shall loop WORKING to AREA\_COUNT minus 1 and pre increment loop counter

1) Set parea to address of nvm\_data with index loop counter

2) Set u32\_crc\_actual of parea to (pu8\_crc\_address of parea)

3) Call function CrcResetDr

4)Set u32\_crc\_calc of parea to call CrcCalcBlockCrc function with parameters pu8\_address of parea and M\_MEMMAP\_NVRAM\_WORKING\_CRC\_CNT

5)Set timestamp of parea to (Addition of pu8\_address of parea and M\_MEMMAP\_TIMESTAMP\_OFFSET).

6) Set b\_valid of parea to (u32\_crc\_actual of parea is equal to u32\_crc\_calc of parea).

10.20.1.7.21.7.2 daucmfxram-InitNvram-LLR-002

Requirement ID: H698-LLD-CMU-FNC-2360

The function shall

a) Call HwCopy with parameters (pu8\_address of nvmdata with index WORKING, pu8\_address of nvmdata with index BACKUP\_2,M\_MEMMAP\_30K), When

- b\_valid of nvmdata with index WORKING is equal to TRUE.

- b\_valid of nvmdata with index BACKUP\_1 AND b\_valid of nvmdata with index BACKUP\_2 to TRUE

- return value of IsOlder function with parameters( timestamp of address of nvmdata with index BACKUP\_1 , timestamp of address of nvmdata with index BACKUP\_2) returns TRUE

- return value of IsOlder function with parameters (timestamp of address of nvmdata with index WORKING, timestamp of address of nvmdata with index BACKUP\_2) returns TRUE

b) Call HwCopy with parameters (pu8\_address of nvmdata with index WORKING, pu8\_address of nvmdata with index BACKUP\_1,M\_MEMMAP\_30K)

when return value of IsOlder function with parameters (timestamp of address of nvmdata with index WORKING, timestamp of address of nvmdata with index BACKUP\_1 returns TRUE

10.20.1.7.21.7.3 daucmfxram-InitNvram-LLR-003

Requirement ID: H698-LLD-CMU-FNC-2361

The function shall

a) Call HwCopy with parameters (pu8\_address of nvmdata with index WORKING, pu8\_address of nvmdata with index BACKUP\_1, M\_MEMMAP\_30K),when

b\_valid of nvmdata with index BACKUP\_1 is equal to TRUE.

return value of IsOlder function with parameters(timestamp of address of nvmdata with index WORKING, timestamp of address of nvmdata with index BACKUP\_1) to TRUE

b) Call HwCopy with parameters (pu8\_address of nvmdata with index WORKING, pu8\_address of nvmdata with index BACKUP\_2, M\_MEMMAP\_30K), when

b\_valid of nvmdata with index BACKUP\_2 is equal to TRUE.

return value of IsOlder with parameters(timestamp of address of nvmdata with index WORKING, timestamp of address of nvmdata with index BACKUP\_2) to TRUE.

10.20.1.7.21.7.4 daucmfxram-InitNvram-LLR-004

Requirement ID: H698-LLD-CMU-FNC-2362

The function shall

a) b\_valid of nvmdata with index BACKUP\_1 is equal to TRUE

- Call HwCopy with parameters (pu8\_address of nvm\_data with index WORKING, pu8\_address of nvmdata with index BACKUP\_2, M\_MEMMAP\_30K),when

return value of b\_valid of nvmdata with index BACKUP\_2 AND IsOlder with parameters(timestamp of address of nvmdata with index BACKUP\_1, timestamp of address of nvmdata with index BACKUP\_2) to TRUE.

- Otherwise, Call HwCopy with parameters(pu8\_address of nvmdata with index WORKING, pu8\_address of nvmdata with BACKUP\_1,M\_MEMMAP\_30K)

10.20.1.7.21.7.5 daucmfxram-InitNvram-LLR-005

Requirement ID: H698-LLD-CMU-FNC-2363

The function shall

a) Call HwCopy with parameters (pu8\_address of nvmdata with index WORKING, pu8\_address of nvmdata with index BACKUP\_2,M\_MEMMAP\_30K), when

b\_valid of nvmdata with index BACKUP\_2 is equal to TRUE.

b) Otherwise, Call RterrDisplayErrorId with parameters (M\_APP\_ID,RterrForever,MEMORY\_ERROR bitwise OR RT\_ERROR\_2)

# 11 Software Low Level Requirements-CMU Flight Module Library

This section specifies the Software Low Level Requirements for CMU Flight-STM Library.

## 11.1 daulibmisc

This module provides all the miscellaneous firmware functions (add-on to CMSIS functions).

### 11.1.1 NvicPriorityGroupConfig

Low Level Design Details about CSU NvicPriorityGroupConfig will follow in the sub sections.

#### 11.1.1.1 Brief Description

The function NvicPriorityGroupConfig configures the priority grouping: pre-emption priority and sub priority.

#### 11.1.1.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-012-CMU).

#### 11.1.1.3 List of global variables accessed and modified

Accessed: None

Modified: None

#### 11.1.1.4 Parameter list (Input/Output)

Inputs : T\_UINT32 nvic\_priority\_group - specifies the priority grouping bits length.

Outputs : None

#### 11.1.1.5 Return Value

None

#### 11.1.1.6 Other CSUs called by this CSU

None

#### 11.1.1.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to NvicPriorityGroupConfig

##### 11.1.1.7.1 daulibmisc-NvicPriorityGroupConfig-LLR-001

Requirement ID: H698-LLD-CMU-FNC-1442

The function shall set the priority group bits [10:8] in aircr (Application interrupt and reset control register) of M\_SCB(System control block) by performing logical OR of M\_AIRCR\_VECTKEY\_MASK and nvic\_priority\_group value.

### 11.1.2 NvicInit

Low Level Design Details about CSU NvicInit will follow in the sub sections.

#### 11.1.2.1 Brief Description

The function NvicInit initializes the NVIC peripheral according to the specifiedparameters in the nvic\_init\_struct.

#### 11.1.2.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-012-CMU).

#### 11.1.2.3 List of global variables accessed and modified

Accessed: None

Modified: None

#### 11.1.2.4 Parameter list (Input/Output)

Inputs: T\_NVIC\_INIT \* nvic\_init\_struct - pointer to a T\_NVIC\_INIT structure that contains the configuration information for the specified NVIC peripheral.

Outputs: None

#### 11.1.2.5 Return Value

None

#### 11.1.2.6 Other CSUs called by this CSU

None

#### 11.1.2.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to NvicInit.

##### 11.1.2.7.1 daulibmisc-NvicInit-LLR-001

Requirement ID: H698-LLD-CMU-FNC-1451

The function shall compute the Interrupt Request (IRQ) When DISABLE is not equal to nvic\_irq\_channel\_cmd of nvic\_init\_struct then compute the Corresponding IRQ Priority as per below logic:

1. Extracted Priority Group bits value (Priority group bits [10:8] of Application interrupt and reset control register) is extracted as M\_PRIORITY\_GROUP subtracted by ((aircr of M\_SCB) bitwise AND with M\_PRIORITY\_GROUP)) right shifted by M\_SHIFT\_BY\_8).
2. Preempt Priority value  is set to M\_FOUR subtracted by Extracted Priority Group bits value.
3. Sub Priority value is set to Sub priority right shift by Extracted Priority Group bits value
4. NVIC IRQ channel preemption priority value is set to (nvic\_irq\_channel\_preemption\_priority of nvic\_init\_struct LEFT\_SHIFT Preempt Priority value) .
5. NVIC IRQ channel subpriority value is set to NVIC IRQ channel preemption priority value bitwise OR with (nvic\_irq\_channel\_subpriority of nvic\_init\_struct bitwise AND with Sub Priority value) .
6. Priority Value is set to NVIC IRQ channel subpriority value left shift by M\_SHIFT\_BY\_4.
7. ip\_reg of index (nvic\_irq\_channel of nvic\_init\_struct) of M\_NVIC is set to Priority Value.

             Note: Refer DM00046982-with FPU-ref-manual.pdf for computing interrupt priority

##### 11.1.2.7.2 daulibmisc-NvicInit-LLR-002

Requirement ID: H698-LLD-CMU-FNC-1452

The function shall configure interrupt set-enable register of NVIC to enable the selected interrupt channel by setting iser of M\_NVIC with index value (nvic\_irq\_channel of nvic\_init\_struct right shift by M\_SHIFT\_BY\_5) to (M\_SHIFT\_BY\_1 left shift by (nvic\_irq\_channel of nvic\_init\_struct   bitwise AND with M\_THIRTY\_ONE)).

##### 11.1.2.7.3 daulibmisc-NvicInit-LLR-003

Requirement ID: H698-LLD-CMU-FNC-1453

The function shall disable the selected interrupt channels by setting Interrupt clear-enable register for selected interrupt channel when NVIC interrupt channel command is disabled i.e,

Set icer of M\_NVIC with index value (nvic\_irq\_channel of nvic\_init\_struct right shift by M\_SHIFT\_BY\_5) to M\_SHIFT\_BY\_1 left shift by (nvic\_irq\_channel of nvic\_init\_struct bitwise AND with M\_THIRTY\_ONE) when DISABLE is equal to nvic\_irq\_channel\_cmd of nvic\_init\_struct.

### 11.1.3 NvicSetVectorTable

Low Level Design Details about CSU NvicSetVectorTable will follow in the sub sections.

#### 11.1.3.1 Brief Description

The function NvicSetVectorTable sets the vector table location and offset.

#### 11.1.3.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-012-CMU).

#### 11.1.3.3 List of global variables accessed and modified

Accessed: None

Modified: None

#### 11.1.3.4 Parameter list (Input/Output)

Inputs: T\_UINT32 nvic\_vector\_tab - specifies if the vector table is in RAM or FLASH memory.

T\_UINT32 offset - Vector Table base offset field.

Outputs: None

#### 11.1.3.5 Return Value

None

#### 11.1.3.6 Other CSUs called by this CSU

None

#### 11.1.3.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to NvicSetVectorTable.

##### 11.1.3.7.1 daulibmisc-NvicSetVectorTable-LLR-001

Requirement ID: H698-LLD-CMU-FNC-1462

The function shall configure vector table offset register of system control block with vector table location (RAM or FLASH, bit 29) and offset (bits [29:7]) by setting vtor of M\_SCB to (nvic\_vector\_tab bitwise OR with (offset bitwise AND with M\_NVIC\_VTOR)).

## 11.2 daulibstm32f4xxcan

This module provides firmware functions to manage the following functionalities of the Controller area network (CAN) peripheral:

* Initialization and Configuration
* CAN Frames Transmission
* CAN Frames Reception
* Interrupts and flags

### 11.2.1 CanDeInit

Low Level Design Details about CSU CanDeInit will follow in the sub sections.

#### 11.2.1.1 Brief Description

The function CanDeInit deinitializes the CAN peripheral registers to their default reset values.

#### 11.2.1.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-012-CMU).

#### 11.2.1.3 List of global variables accessed and modified

Accessed: None

Modified: None

#### 11.2.1.4 Parameter list (Input/Output)

Inputs: T\_CAN\_TYPEDEF \*can\_x - Where x can be 1 or 2 to select the CAN peripheral.

Outputs: None

#### 11.2.1.5 Return Value

None

#### 11.2.1.6 Other CSUs called by this CSU

RccApb1PeriphResetCmd

#### 11.2.1.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to CanDeInit

##### 11.2.1.7.1 daulibstm32f4xxcan-CanDeInit-LLR-001

Requirement ID: H698-LLD-CMU-FNC-1472

The function shall do the following when M\_CAN1 is equal to can\_x:

a) Enable CAN1 reset state by calling the function RccApb1PeriphResetCmd with parameters M\_RCC\_APB1PERIPH\_CAN1 and ENABLE.

b) Release CAN1 from reset state by calling the function RccApb1PeriphResetCmd with parameters M\_RCC\_APB1PERIPH\_CAN1 and DISABLE .

##### 11.2.1.7.2 daulibstm32f4xxcan-CanDeInit-LLR-002

Requirement ID: H698-LLD-CMU-FNC-1473

The function shall do the following when M\_CAN1 is not equal to can\_x:

a) Enable CAN2 reset state by calling the function RccApb1PeriphResetCmd with parameters M\_RCC\_APB1PERIPH\_CAN2 and ENABLE

b) Release CAN2 from reset state by calling the function RccApb1PeriphResetCmd with parameters M\_RCC\_APB1PERIPH\_CAN2 and DISABLE

### 11.2.2 CanInit

Low Level Design Details about CSU CanInit will follow in the sub sections.

#### 11.2.2.1 Brief Description

The function CanInit initializes the CAN peripheral according to the specified parameters in the CAN initialization structure.

#### 11.2.2.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-012-CMU).

#### 11.2.2.3 List of global variables accessed and modified

Accessed: None

Modified: None

#### 11.2.2.4 Parameter list (Input/Output)

Inputs: T\_CAN\_TYPEDEF\* can\_x - where x can be 1 or 2 to select the CAN peripheral.

T\_CAN\_INIT\* can\_init\_struct - pointer to a T\_CAN\_INIT structure that contains the configuration information for the CAN peripheral

Outputs: T\_CAN\_TYPEDEF\* can\_x - where x can be 1 or 2 to select the CAN peripheral.

#### 11.2.2.5 Return Value

T\_UINT8 - Returns the constant which Indicates initialization status which will be M\_CAN\_INITSTATUS\_FAILED or M\_CAN\_INITSTATUS\_SUCCESS.

#### 11.2.2.6 Other CSUs called by this CSU

None

#### 11.2.2.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to CanInit

##### 11.2.2.7.1 daulibstm32f4xxcan-CanInit-LLR-001

Requirement ID: H698-LLD-CMU-FNC-1482

The function shall configure CAN master control register to exit from sleep mode (set bit 1 to 0) i.e, set mcr of can\_x to (mcr of can\_x bitwise AND with negated value of M\_CAN\_MCR\_SLEEP).

##### 11.2.2.7.2 daulibstm32f4xxcan-CanInit-LLR-002

Requirement ID: H698-LLD-CMU-FNC-1483

The function shall configure CAN master control register to request initialization (set bit 0 to 1) i.e, set mcr of can\_x to (mcr of can\_x bitwise OR with M\_CAN\_MCR\_INRQ).

##### 11.2.2.7.3 daulibstm32f4xxcan-CanInit-LLR-003

Requirement ID: H698-LLD-CMU-FNC-1484

The function shall loop till INAK bit in CAN master control register is 0 (M\_CAN\_MSR\_INAK is not equal to to (msr of can\_x bitwise AND with M\_CAN\_MSR\_INAK)) and wait acknowledge is not equal to M\_INAK\_TIMEOUT).

Increment the wait acknowledge by one.

##### 11.2.2.7.4 daulibstm32f4xxcan-CanInit-LLR-004

Requirement ID: H698-LLD-CMU-FNC-1485

The function shall set the initialization status to M\_CAN\_INITSTATUS\_FAILED when INAK bit in CAN master control register is 0 ((msr of can\_x bitwise AND with M\_CAN\_MSR\_INAK) is not equal to M\_CAN\_MSR\_INAK).

##### 11.2.2.7.5 daulibstm32f4xxcan-CanInit-LLR-005

Requirement ID: H698-LLD-CMU-FNC-1486

The function shall enable time triggered communication mode (set TTCM bit in master control register to 1) i.e, set mcr of can\_x to (mcr of can\_x bitwise OR with M\_CAN\_MCR\_TTCM) when (M\_CAN\_MSR\_INAK is equal to (msr of can\_x bitwise AND with M\_CAN\_MSR\_INAK ) (INAK bit in msr of can\_x is 1)) and can\_ttcm of can\_init\_struct is equal ENABLE.

##### 11.2.2.7.6 daulibstm32f4xxcan-CanInit-LLR-006

Requirement ID: H698-LLD-CMU-FNC-1487

The function shall disable time triggered communication mode (reset TTCM bit in mcr of can\_x to 0) i.e, set mcr of can\_x to (mcr of can\_x bitwise AND with negated value of M\_CAN\_MCR\_TTCM) when (M\_CAN\_MSR\_INAK is equal to (msr of can\_x bitwise AND with M\_CAN\_MSR\_INAK ) (INAK bit in msr of can\_x is 1)) and can\_ttcm of can\_init\_struct is not ENABLE .

##### 11.2.2.7.7 daulibstm32f4xxcan-CanInit-LLR-007

Requirement ID: H698-LLD-CMU-FNC-1488

The function shall enable automatic bus-off management (set ABOM bit in master control register to 1) i.e, set mcr of can\_x to (mcr of can\_x bitwise OR with M\_CAN\_MCR\_ABOM) when (M\_CAN\_MSR\_INAK is equal to (msr of can\_x bitwise AND with M\_CAN\_MSR\_INAK ) (INAK bit in msr of can\_x is 1)) and can\_abom of can\_init\_struct is ENABLE.

##### 11.2.2.7.8 daulibstm32f4xxcan-CanInit-LLR-008

Requirement ID: H698-LLD-CMU-FNC-1489

The function shall disable automatic bus-off management (reset ABOM bit in master control register to 0) i.e., set mcr of can\_x to (mcr of can\_x bitwise AND with negated value of M\_CAN\_MCR\_ABOM) when (M\_CAN\_MSR\_INAK is equal to (msr of can\_x bitwise AND with M\_CAN\_MSR\_INAK ) (INAK bit in msr of can\_x is 1)) and can\_abom of can\_init\_struct is DISABLE.

##### 11.2.2.7.9 daulibstm32f4xxcan-CanInit-LLR-009

Requirement ID: H698-LLD-CMU-FNC-1490

The function shall enable automatic wake-up mode (set AWUM bit in master control register to 1) i.e, set mcr of can\_x to (mcr of can\_x bitwise OR with M\_CAN\_MCR\_AWUM) when (M\_CAN\_MSR\_INAK is equal to (msr of can\_x bitwise AND with M\_CAN\_MSR\_INAK ) (INAK bit in msr of can\_x is 1)) and can\_awum of can\_init\_struct is ENABLE.

##### 11.2.2.7.10 daulibstm32f4xxcan-CanInit-LLR-010

Requirement ID: H698-LLD-CMU-FNC-1491

The function shall disable automatic wake-up mode (set AWUM bit in master control register to 0) i.e, set mcr of can\_x to (mcr of can\_x bitwise AND with negated value of M\_CAN\_MCR\_AWUM) when (M\_CAN\_MSR\_INAK is equal to (msr of can\_x bitwise AND with M\_CAN\_MSR\_INAK ) (INAK bit in msr of can\_x is 1)) and can\_awum of can\_init\_struct is DISABLE

##### 11.2.2.7.11 daulibstm32f4xxcan-CanInit-LLR-011

Requirement ID: H698-LLD-CMU-FNC-1492

The function shall enable no automatic retransmission (set NART bit in master control register to 1) i.e, set mcr of can\_x to (mcr of can\_x bitwise OR with M\_CAN\_MCR\_NART) when (M\_CAN\_MSR\_INAK is equal to (msr of can\_x bitwise AND with M\_CAN\_MSR\_INAK ) (INAK bit in msr of can\_x is 1)) and can\_nart of can\_init\_struct is ENABLE

##### 11.2.2.7.12 daulibstm32f4xxcan-CanInit-LLR-012

Requirement ID: H698-LLD-CMU-FNC-1493

The function shall disable no automatic retransmission (set NART bit in master control register to 0) i.e, set mcr of can\_x to (mcr of can\_x bitwise AND with negated value of M\_CAN\_MCR\_NART) when (M\_CAN\_MSR\_INAK is equal to (msr of can\_x bitwise AND with M\_CAN\_MSR\_INAK ) (INAK bit in msr of can\_x is 1)) and can\_nart of can\_init\_struct is DISABLE

##### 11.2.2.7.13 daulibstm32f4xxcan-CanInit-LLR-013

Requirement ID: H698-LLD-CMU-FNC-1494

The function shall enable receive FIFO locked mode (set RFLM bit in mter control register to 1) i.e, set mcr of can\_x to (mcr of can\_x bitwise OR with M\_CAN\_MCR\_RFLM) when (M\_CAN\_MSR\_INAK is equal to (msr of can\_x bitwise AND with M\_CAN\_MSR\_INAK ) (INAK bit in msr of can\_x is 1)) and can\_rflm of can\_init\_struct is ENABLE.

##### 11.2.2.7.14 daulibstm32f4xxcan-CanInit-LLR-014

Requirement ID: H698-LLD-CMU-FNC-1495

The function shall disable receive FIFO locked mode (set RFLM bit in mter control register to 0) i.e, set mcr of can\_x to (mcr of can\_x bitwise AND with negated value of M\_CAN\_MCR\_RFLM) when (M\_CAN\_MSR\_INAK is equal to (msr of can\_x bitwise AND with M\_CAN\_MSR\_INAK ) (INAK bit in msr of can\_x is 1)) and can\_rflm of can\_init\_struct is DISABLE.

##### 11.2.2.7.15 daulibstm32f4xxcan-CanInit-LLR-015

Requirement ID: H698-LLD-CMU-FNC-1496

The function shall set transmit FIFO priority (set TXFP bit in master control register to 1) i.e, set mcr of can\_x to (mcr of can\_x bitwise OR with M\_CAN\_MCR\_TXFP) when (M\_CAN\_MSR\_INAK is equal to (msr of can\_x bitwise AND with M\_CAN\_MSR\_INAK ) (INAK bit in msr of can\_x is 1)) and can\_txfp of can\_init\_struct is ENABLE.

##### 11.2.2.7.16 daulibstm32f4xxcan-CanInit-LLR-016

Requirement ID: H698-LLD-CMU-FNC-1497

The function shall reset transmit FIFO priority (set TXFP bit in master control register to 0) i.e, set mcr of can\_x to (mcr of can\_x bitwise AND with negated value of M\_CAN\_MCR\_TXFP) when can\_txfp of can\_init\_struct is DISABLE and (M\_CAN\_MSR\_INAK is equal to (msr of can\_x bitwise AND with M\_CAN\_MSR\_INAK ) (INAK bit in msr of can\_x is 1)).

##### 11.2.2.7.17 daulibstm32f4xxcan-CanInit-LLR-017

Requirement ID: H698-LLD-CMU-FNC-1498

The function shall set bit timing register as below when (M\_CAN\_MSR\_INAK is equal to (msr of can\_x bitwise AND with M\_CAN\_MSR\_INAK ) (INAK bit in msr of can\_x is 1)).

Set btr of can\_x to (can\_mode of can\_init\_struct left shift by M\_SHIFT\_30 bitwise OR with

can\_sjw of can\_init\_struct left shift by M\_SHIFT\_24 bitwise OR with

can\_bs1 of can\_init\_struct left shift by M\_SHIFT\_16 bitwise OR with

can\_bs2 of can\_init\_struct left shift by M\_SHIFT\_20 bitwise OR with

(can\_prescaler of can\_init\_struct minus M\_ONE))

Note:

a) set Bit 30 with Loop back mode

b) set Bit 24 and 25 with Resynchronization jump width

c) set Bit 16-19 with Time segment 1

d) set Bit 20-22 with Time segment 2

e) set Bit 0-9 with Baud rate prescaler

##### 11.2.2.7.18 daulibstm32f4xxcan-CanInit-LLR-018

Requirement ID: H698-LLD-CMU-FNC-1499

The function shall Request leave initialization (set INRQ bit of Master control register 0) i.e, set mcr of can\_x to (mcr of can\_x bitwise AND with Negation of M\_CAN\_MCR\_INRQ) when (M\_CAN\_MSR\_INAK is equal to (msr of can\_x bitwise AND with M\_CAN\_MSR\_INAK ) (INAK bit in msr of can\_x is 1)).

##### 11.2.2.7.19 daulibstm32f4xxcan-CanInit-LLR-019

Requirement ID: H698-LLD-CMU-FNC-1500

The function shall do the following when (M\_CAN\_MSR\_INAK is equal to (msr of can\_x bitwise AND with M\_CAN\_MSR\_INAK ) (INAK bit in msr of can\_x is 1)).

1. Set wait acknowledgement counter to M\_ZERO.
2. Increment the wait counter by one when it loops till (M\_CAN\_MSR\_INAK is equal to msr of can\_X bitwise AND with M\_CAN\_MSR\_INAK) AND (M\_INAK\_TIMEOUT is not equal to wait acknowledgement counter.
3. Set the initialization status to M\_CAN\_INITSTATUS\_FAILED when INAK bit in MSR of can\_x is 1 (M\_CAN\_MSR\_INAK is equal to msr of can\_X bitwise AND with M\_CAN\_MSR\_INAK).
4. Set the initialization status to M\_CAN\_INITSTATUS\_SUCCESS when INAK bit in MSR of can\_x is 0 (M\_CAN\_MSR\_INAK is not equal to msr of can\_X bitwise AND with M\_CAN\_MSR\_INAK) .

##### 11.2.2.7.20 daulibstm32f4xxcan-CanInit-LLR-020

Requirement ID: H698-LLD-CMU-FNC-1501

The function shall return the initialization status.

### 11.2.3 CanFilterInit

Low Level Design Details about CSU CanFilterInit will follow in the sub sections.

#### 11.2.3.1 Brief Description

The function CanFilterInit configures the CAN reception filter according to the specified parameters in the can\_filter\_init\_struct.

#### 11.2.3.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-012-CMU).

#### 11.2.3.3 List of global variables accessed and modified

Accessed: None

Modified: None

#### 11.2.3.4 Parameter list (Input/Output)

Inputs: T\_CAN\_FILTER\_INIT\* can\_filter\_init\_struct - pointer to a T\_CAN\_FILTER\_INIT structure that contains the configuration information.

Outputs: None

#### 11.2.3.5 Return Value

None

#### 11.2.3.6 Other CSUs called by this CSU

None

#### 11.2.3.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to CanFilterInit

##### 11.2.3.7.1 daulibstm32f4xxcan-CanFilterInit-LLR-001

Requirement ID: H698-LLD-CMU-FNC-1510

The function shall set (FINIT - bit 0) in filter master register to 1 i.e, set fmr of M\_CAN1 to (fmr of M\_CAN1 bitwise OR with M\_FMR\_FINIT).

##### 11.2.3.7.2 daulibstm32f4xxcan-CanFilterInit-LLR-002

Requirement ID: H698-LLD-CMU-FNC-1511

The function shall deactivate the filter x (where x = can\_filter\_number) in filter activation register by setting the corresponding bit to 0 i.e, falr of M\_CAN1 to (falr of M\_CAN1 bitwise AND with negated value of (M\_ONE left shift by can\_filter\_number of can\_filter\_init\_struct)).

##### 11.2.3.7.3 daulibstm32f4xxcan-CanFilterInit-LLR-003

Requirement ID: H698-LLD-CMU-FNC-1512

The function shall do the following when can\_filter\_scale of can\_filter\_init\_struct is equal to M\_CAN\_FILTERSCALE\_16BIT otherwise do nothing.

1. Configure fslr of M\_CAN1 for Dual 16-bit scale configuration by setting the corresponding FSCx bit to 0 (where x= can\_filter\_number of can\_filter\_init\_struct) i.e, set fslr of M\_CAN1 to (fslr of M\_CAN1 bitwise AND with Negation (M\_ONE left shift by can\_filter\_number of can\_filter\_init\_struct).
2. set frl of sfilterregister of index (can\_filter\_number of can\_filter\_init\_struct) of M\_CAN1 to ((M\_MASK\_16 bitwise AND with (can\_filter\_mask\_id\_low of can\_filter\_init\_struct left shift by M\_SHIFT\_16)) bitwise OR with (M\_MASK\_16 bitwise AND with (can\_filter\_id\_low of can\_filter\_init\_struct))).
3. set fr2 of sfilterregister of index (can\_filter\_number of can\_filter\_init\_struct) of M\_CAN1 to ((M\_MASK\_16 bitwise AND with (can\_filter\_mask\_id\_high of can\_filter\_init\_struct left shift by M\_SHIFT\_16)) bitwise OR with (M\_MASK\_16 bitwise AND with (can\_filter\_id\_high of can\_filter\_init\_struct))

##### 11.2.3.7.4 daulibstm32f4xxcan-CanFilterInit-LLR-004

Requirement ID: H698-LLD-CMU-FNC-1513

The function shall do the following when can\_filter\_scale of can\_filter\_init\_struct is equal to M\_CAN\_FILTERSCALE\_32BIT otherwise do nothing.

1. Configure fs1r of M\_CAN1 for Dual 32-bit scale configuration by setting the corresponding FSCx bit to 1 (where x=can\_filter\_init\_struct->can\_filter\_number) i.e, set fslr of M\_CAN1 to (fslr of M\_CAN1 bitwise OR with (M\_ONE left shift by can\_filter\_number of can\_filter\_init\_struct).
2. set frl of sfilterregister of index (can\_filter\_number of can\_filter\_init\_struct) of M\_CAN1 to ((M\_MASK\_16 bitwise AND with (can\_filter\_id\_high of can\_filter\_init\_struct left shift by M\_SHIFT\_16)) bitwise OR with (M\_MASK\_16 bitwise AND with (can\_filter\_id\_low of can\_filter\_init\_struct))).
3. set fr2 of sfilterregister of index (can\_filter\_number of can\_filter\_init\_struct) of M\_CAN1 to ((M\_MASK\_16 bitwise AND with (can\_filter\_mask\_id\_high of can\_filter\_init\_struct left shift by M\_SHIFT\_16)) bitwise OR with (M\_MASK\_16 bitwise AND with (can\_filter\_mask\_id\_low of can\_filter\_init\_struct))).

##### 11.2.3.7.5 daulibstm32f4xxcan-CanFilterInit-LLR-005

Requirement ID: H698-LLD-CMU-FNC-1514

The function shall set the FBMx bit (where x=can\_filter\_init\_struct->can\_filter\_number) in filter mode register to 0 i.e, set fmlr of M\_CAN1 to (fmlr of M\_CAN1 bitwise AND with negated value (M\_ONE left shift by can\_filter\_number of can\_filter\_init\_struct)) when can\_filter\_mode of can\_filter\_init\_struct is equal to M\_CAN\_FILTERMODE\_IDMASK.

##### 11.2.3.7.6 daulibstm32f4xxcan-CanFilterInit-LLR-006

Requirement ID: H698-LLD-CMU-FNC-1515

The function shall set the FBMx bit (where x=can\_filter\_init\_struct->can\_filter\_number) in filter mode register to 1 i.e, set fmlr of M\_CAN1 to (fmlr of M\_CAN1 bitwise OR with (M\_ONE left shift by can\_filter\_number of can\_filter\_init\_struct)) when can\_filter\_mode of can\_filter\_init\_struct is not equal to M\_CAN\_FILTERMODE\_IDMASK.

##### 11.2.3.7.7 daulibstm32f4xxcan-CanFilterInit-LLR-007

Requirement ID: H698-LLD-CMU-FNC-1516

The function shall set the FFAx bit (where x=can\_filter\_init\_struct->can\_filter\_number) in CAN filter FIFO assignment register to 0 i.e, set ffalr of M\_CAN1 to (ffalr of M\_CAN1bitwise AND with negated value of (M\_ONE left shift by can\_filter\_number of can\_filter\_init\_struct)) when can\_filter\_fifo\_assignment of can\_filter\_init\_struct is equal to M\_CAN\_FILTER\_FIFO0 otherwise do nothing.

##### 11.2.3.7.8 daulibstm32f4xxcan-CanFilterInit-LLR-008

Requirement ID: H698-LLD-CMU-FNC-1517

The function shall set the FFAx bit (where x=can\_filter\_init\_struct->can\_filter\_number) in CAN filter FIFO assignment register to 1 i.e, set ffalr of M\_CAN1 to (ffalr of M\_CAN1bitwise OR with (M\_ONE left shift by can\_filter\_number of can\_filter\_init\_struct)) when can\_filter\_fifo\_assignment of can\_filter\_init\_struct is equal to M\_CAN\_FILTER\_FIFO1 otherwise do nothing.

##### 11.2.3.7.9 daulibstm32f4xxcan-CanFilterInit-LLR-009

Requirement ID: H698-LLD-CMU-FNC-1518

The function shall set the FACTx bit (where x=can\_filter\_init\_struct->can\_filter\_number) in CAN filter activation register to 1 i.e, set falr of M\_CAN1 to (falr of M\_CAN1 bitwise OR with (M\_ONE left shift by can\_filter\_number of can\_filter\_init\_struct) when can\_filter\_activation of can\_filter\_init\_struct is ENABLE otherwise do nothing.

##### 11.2.3.7.10 daulibstm32f4xxcan-CanFilterInit-LLR-010

Requirement ID: H698-LLD-CMU-FNC-1519

The function shall set the FINIT bit to 0 (active filters mode) of filter master register i.e, set fmr of M\_CAN1 to (fmr of M\_CAN1 bitwise AND with negated value of M\_FMR\_FINIT).

### 11.2.4 CanTransmit

Low Level Design Details about CSU CanTransmit will follow in the sub sections.

#### 11.2.4.1 Brief Description

The function CanTransmit initiates and transmits a CAN frame message.

#### 11.2.4.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-012-CMU).

#### 11.2.4.3 List of global variables accessed and modified

Accessed: None

Modified: None

#### 11.2.4.4 Parameter list (Input/Output)

Inputs: T\_CAN\_FILTER\_INIT\* can\_x - Where x can be 1 or 2 to select the CAN peripheral.

T\_CAN\_TX\_MSG\* tx\_message - Pointer to a structure which contains CAN Id, CAN dlc and CAN data.

Outputs: T\_CAN\_FILTER\_INIT\* can\_x - Where x can be 1 or 2 to select the CAN peripheral.

T\_CAN\_TX\_MSG\* tx\_message - Pointer to a structure which contains CAN Id, CAN dlc and CAN

#### 11.2.4.5 Return Value

T\_UINT8 - The number of the mailbox that is used for transmission to

M\_CAN\_TXSTATUS\_NOMAILBOX if there is no empty mailbox.

#### 11.2.4.6 Other CSUs called by this CSU

None

#### 11.2.4.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to CanTransmit

##### 11.2.4.7.1 daulibstm32f4xxcan-CanTransmit-LLR-001

Requirement ID: H698-LLD-CMU-FNC-1528

The function shall set transmit mailbox to 0 when TME0 bit of transmit status register is set i.e, (M\_CAN\_TSR\_TME0 is equal to (tsr of can\_x bitwise AND with M\_CAN\_TSR\_TME0)).

##### 11.2.4.7.2 daulibstm32f4xxcan-CanTransmit-LLR-002

Requirement ID: H698-LLD-CMU-FNC-1529

The function shall set transmit mailbox to 1 when TME0 bit of transmit status register is set to 0 i.e, (M\_CAN\_TSR\_TME0 is not equal to (tsr of can\_x bitwise AND with M\_CAN\_TSR\_TME0)) AND (TME1 bit of transmit status register is set to 1 i.e, (M\_CAN\_TSR\_TME1 is equal to (tsr of can\_x bitwise AND with M\_CAN\_TSR\_TME1))).

##### 11.2.4.7.3 daulibstm32f4xxcan-CanTransmit-LLR-003

Requirement ID: H698-LLD-CMU-FNC-1530

The function shall set transmit mailbox to 2 when TME0 bit of transmit status register is set to 0 i.e, (M\_CAN\_TSR\_TME0 is not equal to (tsr of can\_x bitwise AND with M\_CAN\_TSR\_TME0)) AND (TME1 bit of transmit status register is set to 0 i.e, (M\_CAN\_TSR\_TME1 is not equal to (tsr of can\_x bitwise AND with M\_CAN\_TSR\_TME1))) AND (TME2 bit of transmit status register is set to 1 i.e, (M\_CAN\_TSR\_TME2 is equal to (tsr of can\_x bitwise AND with M\_CAN\_TSR\_TME2))).

##### 11.2.4.7.4 daulibstm32f4xxcan-CanTransmit-LLR-004

Requirement ID: H698-LLD-CMU-FNC-1531

The function shall set transmit mailbox to M\_CAN\_TXSTATUS\_NOMAILBOX when TME0 bit of transmit status register is set to 0 i.e, (M\_CAN\_TSR\_TME0 is not equal to (tsr of can\_x bitwise AND with M\_CAN\_TSR\_TME0)) AND (TME1 bit of transmit status register is set to 0 i.e, (M\_CAN\_TSR\_TME1 is not equal to (tsr of can\_x bitwise AND with M\_CAN\_TSR\_TME1))) AND (TME2 bit of transmit status register is set to 0 i.e, (M\_CAN\_TSR\_TME2 is not equal to (tsr of can\_x bitwise AND with M\_CAN\_TSR\_TME2))).

##### 11.2.4.7.5 daulibstm32f4xxcan-CanTransmit-LLR-005

Requirement ID: H698-LLD-CMU-FNC-1532

The function shall reset TXRQ bit of TX mailbox identifier register i.e, set tir of stxmailbox with index transmit mailbox of can\_x to (set tir of stxmailbox with index transmit mailbox of can\_x bitwise AND with M\_TMIDXR\_TXRQ) when any of the transmit mailbox (TME0, TME1 or TME2) is empty i.e, M\_CAN\_TXSTATUS\_NOMAILBOX is not equal to transmit mailbox otherwise no nothing

##### 11.2.4.7.6 daulibstm32f4xxcan-CanTransmit-LLR-006

Requirement ID: H698-LLD-CMU-FNC-1533

The function shall configure TX mailbox identifier register with Standard identifier and RTR data i.e, set tir of stxmailbox with index as transmit mailbox of can\_x to (tir of stxmailbox with index as transmit mailbox of can\_x bitwise OR with ((std\_id of tx\_message left shift by M\_SHIFT\_21) bitwsie OR with (rtr of tx\_message))) when (any of the transmit mailbox (TME0, TME1 or TME2) is empty i.e, M\_CAN\_TXSTATUS\_NOMAILBOX is not equal to transmit mailbox) AND (M\_CAN\_ID\_STD is equal to ide of tx\_message).

##### 11.2.4.7.7 daulibstm32f4xxcan-CanTransmit-LLR-007

Requirement ID: H698-LLD-CMU-FNC-1534

The function shall configure TX mailbox identifier register with Extended identifier, identifier extension and RTR data i.e, set tir of stxmailbox with index as transmit mailbox of can\_x to (tir of stxmailbox with index as transmit mailbox of can\_x bitwise OR with ((ext\_id of tx\_message left shift by M\_SHIFT\_3) bitwsie OR with (ide of tx\_message) bitwise OR with (rtr of tx\_message))) when (any of the transmit mailbox (TME0, TME1 or TME2) is empty i.e, M\_CAN\_TXSTATUS\_NOMAILBOX is not equal to transmit mailbox) AND (M\_CAN\_ID\_STD is not equal to ide of tx\_message).

##### 11.2.4.7.8 daulibstm32f4xxcan-CanTransmit-LLR-008

Requirement ID: H698-LLD-CMU-FNC-1535

The function shall clear the tdtr bits from 4-31 and configure with Data length code of CAN mailbox data length control and time stamp register i.e,

1. set dlc of tx\_message to (dlc of tx\_message bitwise AND with M\_MASK\_4)
2. Set tdtr of stxmailbox with index as transmit mailbox of can\_x to (tdtr of stxmailbox with index as transmit mailbox of can\_x bitwise AND with M\_MASK\_28.
3. Set tdtr of stxmailbox with index as transmit mailbox of can\_x to (tdtr of stxmailbox with index as transmit mailbox of can\_x bitwise OR with dlc of tx\_message.

when (any of the transmit mailbox (TME0, TME1 or TME2) is empty i.e, M\_CAN\_TXSTATUS\_NOMAILBOX is not equal to transmit mailbox)

##### 11.2.4.7.9 daulibstm32f4xxcan-CanTransmit-LLR-009

Requirement ID: H698-LLD-CMU-FNC-1536

The function shall configure CAN mailbox data high register with data field i.e,

1. set tdlr of stx mail box with index as transmit mailbox of can\_x to ((data with index M\_THREE of tx\_message left shift by M\_SHIFT\_24) bitwise OR with (data with index M\_TWO of tx\_message left shift by M\_SHIFT\_16) bitwise OR with (data with index M\_ONE of tx\_message left shift by M\_SHIFT\_8) bitwise OR with (data with index M\_ZERO of tx\_message)).
2. set tdhr of stx mail box with index as transmit mailbox of can\_x to ((data with index M\_SEVEN of tx\_message left shift by M\_SHIFT\_24) bitwise OR with (data with index M\_SIX of tx\_message left shift by M\_SHIFT\_16) bitwise OR with (data with index M\_FIVE of tx\_message left shift by M\_SHIFT\_8) bitwise OR with (data with index M\_FOUR of tx\_message)).

when (any of the transmit mailbox (TME0, TME1 or TME2) is empty i.e, M\_CAN\_TXSTATUS\_NOMAILBOX is not equal to transmit mailbox)

##### 11.2.4.7.10 daulibstm32f4xxcan-CanTransmit-LLR-010

Requirement ID: H698-LLD-CMU-FNC-1537

The function shall Request for transmission (set TXRQ bit in CAN TX mailbox identifier register) i.e, set tir of stxmailbox with index transmit mailbox of can\_x to (tir of stxmailbox with index transmit mailbox of can\_x bitwise OR with M\_TMIDXR\_TXRQ) when (any of the transmit mailbox (TME0, TME1 or TME2) is empty i.e, M\_CAN\_TXSTATUS\_NOMAILBOX is not equal to transmit mailbox).

##### 11.2.4.7.11 daulibstm32f4xxcan-CanTransmit-LLR-011

Requirement ID: H698-LLD-CMU-FNC-1538

The function shall return the transmit mailbox.

### 11.2.5 CanReceive

Low Level Design Details about CSU CanReceive will follow in the sub sections.

#### 11.2.5.1 Brief Description

The function CanReceive Receives a correct CAN frame.

#### 11.2.5.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-012-CMU).

#### 11.2.5.3 List of global variables accessed and modified

Accessed: None

Modified: None

#### 11.2.5.4 Parameter list (Input/Output)

Inputs: T\_CAN\_TYPEDEF\* can\_x - Where x can be 1 or 2 to select the CAN peripheral.

T\_UINT8 fifo\_number: Receive FIFO number, M\_CAN\_FIFO0 or M\_CAN\_FIFO1.

T\_CAN\_RX\_MSG\* rx\_message - Pointer to a structure receive frame which contains CAN Id, CAN DLC, CAN data and FMI number

Outputs: T\_CAN\_RX\_MSG\* rx\_message - Pointer to a structure receive frame which contains CAN Id, CAN DLC, CAN data and FMI number.

T\_CAN\_TYPEDEF\* can\_x - Where x can be 1 or 2 to select the CAN peripheral.

#### 11.2.5.5 Return Value

None

#### 11.2.5.6 Other CSUs called by this CSU

None

#### 11.2.5.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to CanReceive

##### 11.2.5.7.1 daulibstm32f4xxcan-CanReceive-LLR-001

Requirement ID: H698-LLD-CMU-FNC-1547

The function shall extract the IDE (identifier extension) bit value in receive FIFO mailbox identifier register i.e, set ide of rx\_message to (M\_FOUR bitwise AND with rir of sfifomailbox with index fifo\_number of can\_x).

##### 11.2.5.7.2 daulibstm32f4xxcan-CanReceive-LLR-002

Requirement ID: H698-LLD-CMU-FNC-1548

The function shall extract Standard identifier value in receive FIFO mailbox identifier register i.e, set std\_id of rx\_message to (M\_MASK\_11 bitwise AND with (rir of sfifomailbox with index fifo number of can\_x right shift by M\_SHIFT\_21)) when ide of rx\_message is equal to M\_CAN\_ID\_STD (standard ID).

##### 11.2.5.7.3 daulibstm32f4xxcan-CanReceive-LLR-003

Requirement ID: H698-LLD-CMU-FNC-1549

The function shall extract extended identifier value in receive FIFO mailbox identifier register i.e, set ext\_id of rx\_message to (M\_MASK\_29 bitwsie AND with rir of sfifomailbox with index fifo number of can\_x right shifted by M\_SHIFT\_3) when the ide of rx\_message is not equal to M\_CAN\_ID\_STD (Extended ID).

##### 11.2.5.7.4 daulibstm32f4xxcan-CanReceive-LLR-004

Requirement ID: H698-LLD-CMU-FNC-1550

The function shall extract remote transmission request (frame type) value in receive FIFO mailbox identifier register i.e, set rtr of rx\_message to (M\_TWO bitwise AND with rir of sfifomailbox with index fifo number of can\_x).

##### 11.2.5.7.5 daulibstm32f4xxcan-CanReceive-LLR-005

Requirement ID: H698-LLD-CMU-FNC-1551

The function shall extract dlc value in receive FIFO mailbox identifier register i.e, set dlc of rx\_message to (M\_MASK\_4 bitwise AND with rdtr of sfifomailbox with index fifo number of can\_x).

##### 11.2.5.7.6 daulibstm32f4xxcan-CanReceive-LLR-006

Requirement ID: H698-LLD-CMU-FNC-1552

The function shall extract FMI (Filter match index) in receive FIFO mailbox identifier register i.e, set fmi of rx\_message to (M\_MASK\_8 bitwise AND with (rdtr of sfifomailbox with index fifo number of can\_x right shift by M\_SHIFT\_8)).

##### 11.2.5.7.7 daulibstm32f4xxcan-CanReceive-LLR-007

Requirement ID: H698-LLD-CMU-FNC-1553

The function shall extract data field data in receive FIFO mailbox identifier register i.e,

1. Set data of index M\_ZERO of rx\_message to (M\_MASK\_8 bitwise AND with rdlr of sfifomailbox with index fifo number of can\_x).
2. Set data of index M\_ONE of rx\_message to (M\_MASK\_8 bitwise AND with rdlr of sfifomailbox with index fifo number of can\_x right shift by M\_SHIFT\_8).
3. Set data of index M\_TWO of rx\_message to (M\_MASK\_8 bitwise AND with rdlr of sfifomailbox with index fifo number of can\_x right shift by M\_SHIFT\_16).
4. Set data of index M\_THREE of rx\_message to (M\_MASK\_8 bitwise AND with rdlr of sfifomailbox with index fifo number of can\_x right shift by M\_SHIFT\_24).
5. Set data of index M\_FOUR of rx\_message to (M\_MASK\_8 bitwise AND with rdhr of sfifomailbox with index fifo number of can\_x).
6. Set data of index M\_FIVE of rx\_message to (M\_MASK\_8 bitwise AND with rdhr of sfifomailbox with index fifo number of can\_x right shift by M\_SHIFT\_8).
7. Set data of index M\_SIX of rx\_message to (M\_MASK\_8 bitwise AND with rdhr of sfifomailbox with index fifo number of can\_x right shift by M\_SHIFT\_16).
8. Set data of index M\_SEVEN of rx\_message to (M\_MASK\_8 bitwise AND with rdhr of sfifomailbox with index fifo number of can\_x right shift by M\_SHIFT\_24)

##### 11.2.5.7.8 daulibstm32f4xxcan-CanReceive-LLR-008

Requirement ID: H698-LLD-CMU-FNC-1554

The function shall release FIFO 0 (set RFOM0 bit in receive FIFO 0 register (rf0r)) i.e, set rf0r of can\_x to (rf0r of can\_x bitwise OR with M\_CAN\_RF0R\_RFOM0) when M\_CAN\_FIFO0 is equal to fifo number.

##### 11.2.5.7.9 daulibstm32f4xxcan-CanReceive-LLR-009

Requirement ID: H698-LLD-CMU-FNC-1555

The function shall release FIFO 1 (set RFOM1 bit in receive FIFO 1 register (rf1r)) i.e, set rflr of can\_x to (rflr of can\_x bitwise OR with M\_CAN\_RF1R\_RFOM1) when M\_CAN\_FIFO0 is not equal to fifo number.

### 11.2.6 CanItConfig

Low Level Design Details about CSU CanItConfig will follow in the sub sections.

#### 11.2.6.1 Brief Description

The function CanItConfig Enables or disables the specified CANx interrupts.

#### 11.2.6.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-012-CMU).

#### 11.2.6.3 List of global variables accessed and modified

Accessed: None

Modified: None

#### 11.2.6.4 Parameter list (Input/Output)

Inputs: T\_CAN\_TYPEDEF\* can\_x - Where x can be 1 or 2 to select the CAN peripheral.

T\_UINT32 can\_it - Specifies the CAN interrupt sources to be enabled or disabled.

T\_FUNCTIONAL\_STATE new\_state - New state of the CAN interrupts.

Outputs: T\_CAN\_TYPEDEF\* can\_x - Where x can be 1 or 2 to select the CAN peripheral.

#### 11.2.6.5 Return Value

None

#### 11.2.6.6 Other CSUs called by this CSU

None

#### 11.2.6.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to CanItConfig

##### 11.2.6.7.1 daulibstm32f4xxcan-CanItConfig-LLR-001

Requirement ID: H698-LLD-CMU-FNC-1564

The function shall configure interrupt enable register (ier) to enable the selected can interrupt i.e, set ier of can\_x to (ier of can\_x bitwise OR with can\_it) when the new\_state is ENABLE.

##### 11.2.6.7.2 daulibstm32f4xxcan-CanItConfig-LLR-002

Requirement ID: H698-LLD-CMU-FNC-1565

The function shall configure interrupt enable register (ier) to disable the selected can interrupt i.e, set ier of can\_x to (ier of can\_x bitwise AND with negated value of can\_it when the new\_state is DISABLE.

## 11.3 daulibstm32f4xxcrc

daulibstm32f4xxcrc CSC provides all the CRC firmware functions.

### 11.3.1 CrcResetDr

Low Level Design Details about CSU CrcResetDr will follow in the sub sections.

#### 11.3.1.1 Brief Description

The function CrcResetDr resets the CRC control register (cr).

#### 11.3.1.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-012-CMU).

#### 11.3.1.3 List of global variables accessed and modified

Accessed: None

Modified: None

#### 11.3.1.4 Parameter list (Input/Output)

Inputs: None

Outputs: None

#### 11.3.1.5 Return Value

None

#### 11.3.1.6 Other CSUs called by this CSU

None

#### 11.3.1.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to CrcResetDr

##### 11.3.1.7.1 daulibstm32f4xxcrc-CrcResetDr-LLR-001

Requirement ID: H698-LLD-CMU-FNC-1575

The function shall set the RESET bit of CRC Control register (cr) to reset the CRC calculation unit i.e, set cr of M\_CRC to M\_CRC\_CR\_RESET.

### 11.3.2 CrcCalcBlockCrc

Low Level Design Details about CSU CrcCalcBlockCrc will follow in the sub sections.

#### 11.3.2.1 Brief Description

The function CrcCalcBlockCrc computes the 32-bit M\_CRC of a given buffer of data word (32-bit).

#### 11.3.2.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-012-CMU).

#### 11.3.2.3 List of global variables accessed and modified

Accessed: None

Modified: None

#### 11.3.2.4 Parameter list (Input/Output)

Inputs: T\_UINT32 pbuffer - Buffer containing the data to be computed.

T\_UINT32 buffer\_length - Length of the buffer to be computed.

Outputs: None

#### 11.3.2.5 Return Value

T\_UINT32 - Returns 32-bit CRC.

#### 11.3.2.6 Other CSUs called by this CSU

None

#### 11.3.2.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to CrcCalcBlockCrc

##### 11.3.2.7.1 daulibstm32f4xxcrc-CrcCalcBlockCrc-LLR-001

Requirement ID: H698-LLD-CMU-FNC-1584

The function shall copy the data from pbuffer with index as loop counter index to CRC data register (dr of M\_CRC) for loop counter index from M\_ZERO to buffer\_length and returns the calculated crc (dr of M\_CRC).

## 11.4 daulibstm32f4xxflash

daulibstm32f4xxflash CSC provides firmware functions to manage the following functionalities of the FLASH peripheral:

FLASH Interface configuration

a) Set the latency

b) Enable/Disable the prefetch buffer

c) Enable/Disable the Instruction cache and the Data cache

### 11.4.1 FlashSetLatency

Low Level Design Details about CSU FlashSetLatency will follow in the sub sections.

#### 11.4.1.1 Brief Description

The function FlashSetLatency sets the code latency value.

#### 11.4.1.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-012-CMU).

#### 11.4.1.3 List of global variables accessed and modified

Accessed: None

Modified: None

#### 11.4.1.4 Parameter list (Input/Output)

Inputs: T\_UINT32 flash\_latency - specifies the FLASH Latency value.

Outputs: None

#### 11.4.1.5 Return Value

None

#### 11.4.1.6 Other CSUs called by this CSU

None

#### 11.4.1.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to FlashSetLatency.

##### 11.4.1.7.1 daulibstm32f4xxflash-FlashSetLatency-LLR-001

Requirement ID: H698-LLD-CMU-FNC-1594

The function shall set the code latency value in acr (Flash access control register) i.e,Set dereference of the address M\_ACR\_BYTE0\_ADDRESS to flash\_latency.

### 11.4.2 FlashPrefetchBufferCmd

Low Level Design Details about CSU FlashPrefetchBufferCmd will follow in the sub sections.

#### 11.4.2.1 Brief Description

The function FlashPrefetchBufferCmd enables or disables the Prefetch Buffer.

#### 11.4.2.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-012-CMU).

#### 11.4.2.3 List of global variables accessed and modified

Accessed: None

Modified: None

#### 11.4.2.4 Parameter list (Input/Output)

Inputs: T\_FUNCTIONAL\_STATE new\_state - New state of the Prefetch Buffer.

Outputs: None

#### 11.4.2.5 Return Value

None

#### 11.4.2.6 Other CSUs called by this CSU

None

#### 11.4.2.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to FlashPrefetchBufferCmd

##### 11.4.2.7.1 daulibstm32f4xxflash-FlashPrefetchBufferCmd-LLR-001

Requirement ID: H698-LLD-CMU-FNC-1603

The function shall enable the prefetch buffer (set PRFTEN – bit 8) in acr (Flash access control register) when new\_state is ENABLE i.e, set acr of M\_FLASH with (acr of M\_FLASH bitwise OR with M\_FLASH\_ACR\_PRFTEN).

##### 11.4.2.7.2 daulibstm32f4xxflash-FlashPrefetchBufferCmd-LLR-002

Requirement ID: H698-LLD-CMU-FNC-1604

The function shall disable the prefetch buffer (reset PRFTEN – bit 8) in acr (Flash access control register) when new\_state is DISABLE i.e, set acr of M\_FLASH with (acr of M\_FLASH bitwise AND with negated value of M\_FLASH\_ACR\_PRFTEN).

### 11.4.3 FlashInstructionCacheCmd

Low Level Design Details about CSU FlashInstructionCacheCmd will follow in the sub sections.

#### 11.4.3.1 Brief Description

The function FlashInstructionCacheCmd enables or disables the Instruction Cache feature.

#### 11.4.3.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-012-CMU).

#### 11.4.3.3 List of global variables accessed and modified

Accessed: None

Modified: None

#### 11.4.3.4 Parameter list (Input/Output)

Inputs: T\_FUNCTIONAL\_STATE new\_state - New state of the Instruction Cache.

Outputs: None

#### 11.4.3.5 Return Value

None

#### 11.4.3.6 Other CSUs called by this CSU

None

#### 11.4.3.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to FlashInstructionCacheCmd

##### 11.4.3.7.1 daulibstm32f4xxflash-FlashInstructionCacheCmd-LLR-001

Requirement ID: H698-LLD-CMU-FNC-1613

The function shall enable the Instruction Cache (set ICEN – bit 9) in acr (Flash access control register) when new\_state is ENABLE i.e, set acr of M\_FLASH with (acr of M\_FLASH bitwise OR with M\_FLASH\_ACR\_ICEN).

##### 11.4.3.7.2 daulibstm32f4xxflash-FlashInstructionCacheCmd-LLR-002

Requirement ID: H698-LLD-CMU-FNC-1614

The function shall disable the Instruction Cache (reset ICEN – bit 9) in acr (Flash access control register) when new\_state is DISABLE i.e, set acr of M\_FLASH with (acr of M\_FLASH bitwise AND with negated value of M\_FLASH\_ACR\_ICEN).

### 11.4.4 FlashDataCacheCmd

Low Level Design Details about CSU FlashDataCacheCmd will follow in the sub sections.

#### 11.4.4.1 Brief Description

The function FlashDataCacheCmd enables or disables the Data Cache feature.

#### 11.4.4.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-012-CMU).

#### 11.4.4.3 List of global variables accessed and modified

Accessed: None

Modified: None

#### 11.4.4.4 Parameter list (Input/Output)

Inputs: T\_FUNCTIONAL\_STATE new\_state - New state of the data Cache.

Outputs: None

#### 11.4.4.5 Return Value

None

#### 11.4.4.6 Other CSUs called by this CSU

None

#### 11.4.4.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to FlashDataCacheCmd.

##### 11.4.4.7.1 daulibstm32f4xxflash-FlashDataCacheCmd-LLR-001

Requirement ID: H698-LLD-CMU-FNC-1623

The function shall enable the Data Cache (set DCEN - bit 10) in acr (Flash access control register) when new\_state is ENABLE i.e, set acr of M\_FLASH with (acr of M\_FLASH bitwise OR with M\_FLASH\_ACR\_DCEN).

##### 11.4.4.7.2 daulibstm32f4xxflash-FlashDataCacheCmd-LLR-002

Requirement ID: H698-LLD-CMU-FNC-1624

The function shall disable the Data Cache (reset DCEN - bit 10) of acr (Flash access control register) when new\_state is DISABLE i.e, set acr of M\_FLASH with (acr of M\_FLASH bitwise AND with negated value of M\_FLASH\_ACR\_DCEN).

## 11.5 daulibstm32f4xxfsmc

### 11.5.1 FsmcNorSramInit

Low Level Design Details about CSU FsmcNorSramInit will follow in the sub sections.

#### 11.5.1.1 Brief Description

#### 11.5.1.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-012-CMU).

#### 11.5.1.3 List of global variables accessed and modified

Accessed: None

Modified: None

#### 11.5.1.4 Parameter list (Input/Output)

Inputs: T\_FSMC\_NORSRAM\_INIT\* fsmc\_norsraminit\_struct - Pointer to a T\_FSMC\_NORSRAM\_INIT structure that contains the configuration information for the FSMC NOR/SRAM specified Banks.

Outputs: None

#### 11.5.1.5 Return Value

None

#### 11.5.1.6 Other CSUs called by this CSU

None

#### 11.5.1.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to FsmcNorSramInit

##### 11.5.1.7.1 daulibstm32f4xxfsmc-FsmcNorSramInit-LLR-001

Requirement ID: H698-LLD-CMU-FNC-1634

The function shall configure the FSMC BANK1 NOR/SRAM FSMC Bank control register with the received configuration information (Address/data multiplexing enable bit, Memory Type, Memory databus width, Burst enable bit, Wait signal during asynchronous transfers, Wait signal polarity bit, Wrapped burst mode, Wait timing configuration, Write enable bit, Wait enable bit, Extended mode enable, Write burst enable) i.e,

Set btcr of M\_FSMC\_BANK1 with index (fsmc\_bank of fsmc\_norsraminit\_struct) to (fsmc\_data\_address\_mux of fsmc\_norsraminit\_struct bitwise OR with

fsmc\_memory\_type of fsmc\_norsraminit\_struct bitwise OR with

fsmc\_memory\_datawidth of fsmc\_norsraminit\_struct bitwise OR with

fsmc\_burst\_accessmode of fsmc\_norsraminit\_struct bitwise OR with

fsmc\_asynchronous\_wait of fsmc\_norsraminit\_struct bitwise OR with

fsmc\_waitsignal\_polarity of fsmc\_norsraminit\_struct bitwise OR with

fsmc\_wrap\_mode of fsmc\_norsraminit\_struct bitwise OR with

fsmc\_waitsignal\_active of fsmc\_norsraminit\_struct bitwise OR with

fsmc\_write\_operation of fsmc\_norsraminit\_struct bitwise OR with

fsmc\_waitsignal of fsmc\_norsraminit\_struct bitwise OR with

fsmc\_extended\_mode of fsmc\_norsraminit\_struct bitwise OR with

fsmc\_write\_burst of fsmc\_norsraminit\_struct).

##### 11.5.1.7.2 daulibstm32f4xxfsmc-FsmcNorSramInit-LLR-002

Requirement ID: H698-LLD-CMU-FNC-1635

The function shall configure the FSMC BANK1 NOR/SRAM Bank Timing register with the received configuration information (Address setup phase duration, Address-hold phase duration, Memory databus width, Data-phase duration, Bus turnaround phase duration, Clock divide ratio, Data latency, Access mode) i.e,

Set btcr of M\_FSMC\_BANK1 with index (fsmc\_bank of fsmc\_norsraminit\_struct added with M\_ONE) to (fsmc\_address\_setuptime of fsmc\_readwrite\_timing\_struct of fsmc\_norsraminit\_struct bitwise OR with (fsmc\_address\_holdtime of fsmc\_readwrite\_timing\_struct of fsmc\_norsraminit\_struct left shift by M\_BTR\_ADDR\_HOLD\_TIME) bitwise OR with

(fsmc\_data\_setuptime of fsmc\_readwrite\_timing\_struct of fsmc\_norsraminit\_struct left shift by M\_BTR\_DATA\_SETUP\_TIME) bitwise OR with

(fsmc\_bus\_turnaround\_duration of fsmc\_readwrite\_timing\_struct of fsmc\_norsraminit\_struct left shift by M\_BTR\_TURN\_ARND\_DURATION) bitwise OR with

(fsmc\_clk\_division of fsmc\_readwrite\_timing\_struct of fsmc\_norsraminit\_struct left shift by M\_BTR\_CLK\_DIVISION) bitwise OR with

(fsmc\_data\_latency of fsmc\_readwrite\_timing\_struct of fsmc\_norsraminit\_struct left shift by M\_BTR\_DATA\_LATENCY) bitwise OR with

(fsmc\_access\_mode of fsmc\_readwrite\_timing\_struct of fsmc\_norsraminit\_struct))

##### 11.5.1.7.3 daulibstm32f4xxfsmc-FsmcNorSramInit-LLR-003

Requirement ID: H698-LLD-CMU-FNC-1636

The function shall reset Bank1 NOR/SRAM Flash write timing register i.e, Set bwtr of M\_FSMC\_BANK1E with index (fsmc\_bank of fsmc\_norsraminit\_struct) to M\_BWTR\_RESET\_VALUE.

### 11.5.2 FsmcNorSramCmd

Low Level Design Details about CSU FsmcNorSramCmd will follow in the sub sections.

#### 11.5.2.1 Brief Description

The function FsmcNorSramCmd enables or disables the specified NOR/SRAM Memory Bank.

#### 11.5.2.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-012-CMU).

#### 11.5.2.3 List of global variables accessed and modified

Accessed: None

Modified: None

#### 11.5.2.4 Parameter list (Input/Output)

Inputs: T\_UINT32 fsmc\_bank - Specifies the FSMC Bank to be used

T\_FUNCTIONAL\_STATE new\_state - New state of the fsmc\_bank.

Outputs: None

#### 11.5.2.5 Return Value

None

#### 11.5.2.6 Other CSUs called by this CSU

None

#### 11.5.2.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to FsmcNorSramCmd

##### 11.5.2.7.1 daulibstm32f4xxfsmc-FsmcNorSramCmd-LLR-001

Requirement ID: H698-LLD-CMU-FNC-1645

The function shall enable selected NOR/SRAM Memory Bank (MBKEN - bit 0) in the Bank Control Register when new state of FSMC bank is ENABLE i.e,Set btcr of M\_FSMC\_BANK1 with index fsmc\_bank to (btcr of M\_FSMC\_BANK1 with index fsmc\_bank bitwise OR with M\_BCR\_MBKEN\_SET).

##### 11.5.2.7.2 daulibstm32f4xxfsmc-FsmcNorSramCmd-LLR-002

Requirement ID: H698-LLD-CMU-FNC-1646

The function shall disable selected NOR/SRAM Memory Bank (MBKEN - bit 0) in the Bank Control Register when new state of FSMC bank is DISABLE i.e,Set btcr of M\_FSMC\_BANK1 with index fsmc\_bank to (btcr of M\_FSMC\_BANK1 with index fsmc\_bank bitwise AND with value of M\_BCR\_MBKEN\_RESET).

## 11.6 daulibstm32f4xxgpio

This module provides the implementation of firmware functions to manage the following functionalities of the GPIO peripheral:

* Initialization and Configuration
* GPIO Read and Write
* GPIO Alternate functions configuration

### 11.6.1 GpioInit

Low Level Design Details about CSU GpioInit will follow in the sub sections.

#### 11.6.1.1 Brief Description

The function GpioInit initializes the gpio peripheral according to the received parameters.

#### 11.6.1.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-012-CMU).

#### 11.6.1.3 List of global variables accessed and modified

Accessed: None

Modified: None

#### 11.6.1.4 Parameter list (Input/Output)

Inputs: T\_GPIO\* gpio\_x - GPIO peripheral, where x can be (A...I) to select the GPIO peripheral.

T\_GPIO\_INIT\* gpio\_init\_struct - Pointer to a T\_GPIO\_INIT structure that contains the configuration information for the specified GPIO peripheral.

Outputs: T\_GPIO\* gpio\_x - GPIO peripheral, where x can be (A...I) to select the GPIO peripheral.

#### 11.6.1.5 Return Value

None

#### 11.6.1.6 Other CSUs called by this CSU

None

#### 11.6.1.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to GpioInit

##### 11.6.1.7.1 daulibstm32f4xxgpio-GpioInit-LLR-001

Requirement ID: H698-LLD-CMU-FNC-1656

The function shall loop for all gpio pin position (M\_ZERO to M\_GPIO\_PERIPHERALS) and does the following:

1. Set position variable to M\_ONE left shift by gpio pin position
2. Calculate the current pin position i.e, set current pin position to (gpio\_pin of gpio\_init\_struct bitwise AND with position variable).
3. when current pin position (pin\_position) is equal to position variable(position), perform as follows otherwise do nothing.

* Clear the GPIO\_MODER register for the current pin position i.e, set moder of gpio\_x to (moder of gpio\_x bitwise AND with negated value of (M\_GPIO\_MODER\_MODER0 left shift by (gpio pin position multiplied with M\_TWO))).
* Set the GPIO\_MODER register with the received gpio mode i.e, set moder of gpio\_x to (moder of gpio\_x bitwise OR with (gpio\_mode of gpio\_init\_struct left shift by (gpio pin position multiplied by M\_TWO))).
* When the received gpio mode is output or alternate function(GPIO\_MODE\_OUT is equal to gpio\_mode of gpio\_init\_struct OR GPIO\_MODE\_AF is equal to gpio\_mode of gpio\_init\_struct), perform as follows otherwise do nothing.
* Clear the GPIO\_OSPEEDR register for the current pin position i.e, set ospeedr of gpio\_x to (ospeedr of gpio\_x bitwise AND with negated value of (M\_GPIO\_OSPEEDER\_OSPEEDR0 left shift by (gpio pin position multiplied by M\_TWO))).
* Set the GPIO\_OSPEEDR register with the received gpio speed i.e, set ospeedr of gpio\_x to (ospeedr of gpio\_x bitwise OR with (gpio\_speed of gpio\_init\_struct left shift by (gpio pin position multiplied by M\_TWO))).
* Clear the GPIO\_OTYPER register for the current pin position i.e, set otyper of gpio\_x to (otyper of gpio\_x bitwise AND with negated value of (M\_GPIO\_OTYPER\_OT\_0 left shift by gpio pin position)).
* Set the GPIO\_OTYPER register with the received gpio output type i.e, set otyper of gpio\_x to (otyper of gpio\_x bitwise OR with (gpio\_otype of gpio\_init\_struct left shift by gpio pin position)).
* Clear the GPIO\_PUPDR register for the current pin position i.e, set pupdr of gpio\_x to (pupdr of gpio\_x bitwise AND with negated value of (M\_GPIO\_PUPDR\_PUPDR0 left shift by (gpio pin position multiplied by M\_TWO))).
* Set the GPIOx\_PUPDR register with Pull-up Pull down resistor configuration, i.e, set pupdr of gpio\_x to (pupdr of gpio\_x bitwise OR with gpio\_pupd of gpio\_init\_struct left shift by (gpio pin position multiplied by M\_TWO)).

### 11.6.2 GpioSetBits

Low Level Design Details about CSU GpioSetBits will follow in the sub sections.

#### 11.6.2.1 Brief Description

The function GpioSetBits sets the selected data port bits.

#### 11.6.2.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-012-CMU).

#### 11.6.2.3 List of global variables accessed and modified

Accessed: None

Modified: None

#### 11.6.2.4 Parameter list (Input/Output)

Inputs: T\_GPIO\* gpio\_pin - Specifies the port bits to be set.

Outputs: T\_UINT16 gpio\_x - GPIO peripheral,where x can be (A..I) to select the GPIO peripheral.

#### 11.6.2.5 Return Value

None

#### 11.6.2.6 Other CSUs called by this CSU

None

#### 11.6.2.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to GpioSetBits

##### 11.6.2.7.1 daulibstm32f4xxgpio-GpioSetBits-LLR-001

Requirement ID: H698-LLD-CMU-FNC-1665

The function shall set the GPIO port bit set/reset low register i.e, set bsrrl of gpio\_x to gpio\_pin.

### 11.6.3 GpioResetBits

Low Level Design Details about CSU GpioResetBits will follow in the sub sections.

#### 11.6.3.1 Brief Description

The function GpioResetBits clears the selected data port bits.

#### 11.6.3.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-012-CMU).

#### 11.6.3.3 List of global variables accessed and modified

Accessed: None

Modified: None

#### 11.6.3.4 Parameter list (Input/Output)

Inputs: T\_UINT16 gpio\_pin - specifies the port bits to be reset.

Outputs: T\_GPIO\* gpio\_x - GPIO peripheral,where x can be (A..I) to select the GPIO peripheral.

#### 11.6.3.5 Return Value

None

#### 11.6.3.6 Other CSUs called by this CSU

None

#### 11.6.3.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to GpioResetBits

##### 11.6.3.7.1 daulibstm32f4xxgpio-GpioResetBits-LLR-001

Requirement ID: H698-LLD-CMU-FNC-1674

The function shall set the GPIO port bit set/reset high register i.e, bsrrh of gpio\_x to gpio\_pin.

### 11.6.4 GpioReadInputDataBit

Low Level Design Details about CSU GpioReadInputDataBit will follow in the sub sections.

#### 11.6.4.1 Brief Description

The function GpioReadInputDataBit reads the specified input port pin.

#### 11.6.4.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-012-CMU).

#### 11.6.4.3 List of global variables accessed and modified

Accessed: None

Modified: None

#### 11.6.4.4 Parameter list (Input/Output)

Inputs: T\_UINT16 gpio\_pin - Specifies the port bits to read.

T\_GPIO\* gpio\_x - GPIO peripheral,where x can be (A..I) to select the GPIO peripheral.

Outputs: None

#### 11.6.4.5 Return Value

T\_UINT8 - Returns the input port pin value.

#### 11.6.4.6 Other CSUs called by this CSU

None

#### 11.6.4.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to GpioReadInputDataBit

##### 11.6.4.7.1 daulibstm32f4xxgpio-GpioReadInputDataBit-LLR-001

Requirement ID: H698-LLD-CMU-FNC-1683

The function shall return BIT\_SET if gpio\_pin in gpio\_x input data register (idr) is not equal to BIT\_RESET .

##### 11.6.4.7.2 daulibstm32f4xxgpio-GpioReadInputDataBit-LLR-002

Requirement ID: H698-LLD-CMU-FNC-1684

The function shall return BIT\_RESET if gpio\_pin in gpio\_x input data register (idr) is equal to BIT\_RESET.

### 11.6.5 GpioToggleBits

Low Level Design Details about CSU GpioToggleBits will follow in the sub sections.

#### 11.6.5.1 Brief Description

The function GpioToggleBits toggles the specified GPIO pins.

#### 11.6.5.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-012-CMU).

#### 11.6.5.3 List of global variables accessed and modified

Accessed: None

Modified: None

#### 11.6.5.4 Parameter list (Input/Output)

Inputs: T\_GPIO\* gpio\_x - GPIO peripheral,where x can be (A..I) to select the GPIO peripheral.

T\_UINT16 gpio\_pin - Specifies the pins to be toggled.

Outputs: T\_GPIO\* gpio\_x - GPIO peripheral,where x can be (A..I) to select the GPIO peripheral.

#### 11.6.5.5 Return Value

None

#### 11.6.5.6 Other CSUs called by this CSU

None

#### 11.6.5.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to GpioToggleBits.

##### 11.6.5.7.1 daulibstm32f4xxgpio-GpioToggleBits-LLR-001

Requirement ID: H698-LLD-CMU-FNC-1693

The function shall toggle the received gpio pin gpio\_pin in Gpio output data register i.e, set odr of gpio\_x to (odr of gpio\_x bitwise XOR with gpio\_pin).

### 11.6.6 GpioPinAFConfig

Low Level Design Details about CSU GpioPinAFConfig will follow in the sub sections.

#### 11.6.6.1 Brief Description

The function GpioPinAFConfig changes the mapping of the specified pin.

#### 11.6.6.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-012-CMU).

#### 11.6.6.3 List of global variables accessed and modified

Accessed: None

Modified: None

#### 11.6.6.4 Parameter list (Input/Output)

Inputs: T\_GPIO\* gpio\_x - GPIO peripheral,where x can be (A..I) to select the GPIO peripheral.

T\_UINT16 gpio\_pinsource - specifies the pin for the Alternate function.

T\_UINT8 gpio\_af - selects the pin to be used as Alternate function.

Outputs: T\_GPIO\* gpio\_x - GPIO peripheral,where x can be (A..I) to select the GPIO peripheral.

#### 11.6.6.5 Return Value

None

#### 11.6.6.6 Other CSUs called by this CSU

None

#### 11.6.6.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to GpioPinAFConfig.

##### 11.6.6.7.1 daulibstm32f4xxgpio-GpioPinAFConfig-LLR-001

Requirement ID: H698-LLD-CMU-FNC-1702

The function shall change the mapping of specified pin gpio\_pinsource to the alternate function by configuring the afr (Alternate function register) i.e,

1. set Temp afr register to (gpio\_af left shift by (gpio\_pinsource bitwise AND with (M\_SEVEN multiplied with M\_FOUR))).
2. Set afr of gpio\_x with index (gpio\_ pinsource right shift by M\_THREE) to (afr of gpio\_x with index (gpio\_ pinsource right shift by M\_THREE) bitwise AND with negated value of (M\_FIFTEEN left shift by (gpio\_ pinsource bitwise AND with (M\_SEVEN multiplied with M\_FOUR)))).
3. set Temp afr register 2 to (afr of gpio\_x with index (gpio\_ pinsource right shift by M\_THREE) bitwise OR with Temp afr register.
4. Set afr of gpio\_x with index (gpio\_ pinsource right shift by M\_THREE) to Temp afr register 2.

## 11.7 daulibstm32f4xxi2c

daulibstm32f4xxi2c module provides firmware functions to manage functionalities of the Inter-integrated circuit (I2C).

### 11.7.1 I2cInit

Low Level Design Details about CSU I2cInit will follow in the sub sections.

#### 11.7.1.1 Brief Description

The function I2cInit initializes the i2cx peripheral.

#### 11.7.1.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD (H698-003-012-CMU).

#### 11.7.1.3 List of global variables accessed and modified

Accessed : None

Modified : None

#### 11.7.1.4 Parameter list (Input/Output)

Inputs : T\_I2C\* i2cx - Pointer to select the I2C peripheral where x can be 1, 2 or 3

T\_I2C\_INIT\* i2c\_init\_struct - Pointer to a T\_I2C\_INIT structure that contains the configuration

information for the specified I2C peripheral

Outputs :T\_I2C\* i2cx - Pointer to select the I2C peripheral where x can be 1, 2 or 3

#### 11.7.1.5 Return Value

None

#### 11.7.1.6 Other CSUs called by this CSU

RccGetClocksFreq

#### 11.7.1.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to I2cInit.

##### 11.7.1.7.1 daulibstm32f4xxi2c-I2cInit-LLR-001

Requirement ID: H698-LLD-CMU-FNC-2185

The function shall configure i2cx CR2 by performing the following operations:

- Store the i2cx CR2 value in temporary register

- Clear frequency FREQ[5:0] bits from temporary register by performing Bitwise AND operation of temporary register with negation of M\_I2C\_CR2\_FREQ

- Call RccGetClocksFreq with parameter as reference to rcc clocks and then store pclk\_1\_frequency of

rcc clocks to pclk1.

- Set frequency with division of pclk1 by M\_TEN\_LAKH and set temporary register to Bitwise OR of temporary register with frequency

- Write to i2cx CR2 from temporary register.

##### 11.7.1.7.2 daulibstm32f4xxi2c-I2cInit-LLR-002

Requirement ID: H698-LLD-CMU-FNC-2186

The function shall disable the selected I2C peripheral to configure TRISE by performing cr1 of i2cx Bitwise AND with negation of M\_I2C\_CR1\_PE and reset the temporary register in order to clear F/S, DUTY and CCR[11:0] bits.

##### 11.7.1.7.3 daulibstm32f4xxi2c-I2cInit-LLR-003

Requirement ID: H698-LLD-CMU-FNC-2187

The function shall configure speed in standard mode by performing the following operations when I2C\_ClockSpeed of i2c\_init\_struct is less than or equal to M\_ONE\_LAKH:

- Calculate standard mode speed as result is equal to division of pclk1 by (Bitshift of I2C\_ClockSpeed of i2c\_init\_struct to left by M\_ONE) and store it in result variable.

- Set minimum allowed speed M\_HEX\_FOUR to result when result is less than minimum allowed speed M\_HEX\_FOUR.

- Set speed value for standard mode by performing bitwise OR of result with temporary register and then store it in temporary register.

- Set Maximum Rise Time(trise of i2cx) to sum of frequency range and M\_ONE.

##### 11.7.1.7.4 daulibstm32f4xxi2c-I2cInit-LLR-004

Requirement ID: H698-LLD-CMU-FNC-2188

The function shall do the following,

When I2C\_ClockSpeed of i2c\_init\_struct is greater than M\_ONE\_LAKH and I2C\_DutyCycle of i2c\_init\_struct is equal to M\_I2C\_DUTYCYCLE\_2

Set result variable to division of pclk1 by (product of I2C\_ClockSpeed of i2c\_init\_struct and M\_THREE) .

##### 11.7.1.7.5 daulibstm32f4xxi2c-I2cInit-LLR-005

Requirement ID: H698-LLD-CMU-FNC-2189

The function shall do the following,

When I2C\_ClockSpeed of i2c\_init\_struct is greater than M\_ONE\_LAKH and I2C\_DutyCycle of i2c\_init\_struct is other than M\_I2C\_DUTYCYCLE\_2.

Set result variable to division of pclk1 by (product of I2C\_ClockSpeed of i2c\_init\_struct and M\_RT\_ERROR\_NUM25) and set result variable to result variable Bitwise OR with M\_I2C\_DUTYCYCLE\_16\_9.

##### 11.7.1.7.6 daulibstm32f4xxi2c-I2cInit-LLR-006

Requirement ID: H698-LLD-CMU-FNC-2190

The function shall Set result variable to (result variable bitwise OR with M\_HEX\_ONE), When I2C\_ClockSpeed of i2c\_init\_struct is greater than M\_ONE\_LAKH and result bitwise AND with M\_I2C\_CCR\_CCR is equal to M\_ZERO .

##### 11.7.1.7.7 daulibstm32f4xxi2c-I2cInit-LLR-007

Requirement ID: H698-LLD-CMU-FNC-2191

The function shall set the following when I2C\_ClockSpeed of i2c\_init\_struct is greater than M\_ONE\_LAKH.

- Set temporary register to Bitwise OR of temporary register with (Bitwise OR of result with M\_I2C\_CCR\_FS)

- Set trise of i2cx with product of (freq\_range with ((M\_THREE\_HUNDRED divided by M\_ONE\_THOUSAND) plus M\_ONE)).

##### 11.7.1.7.8 daulibstm32f4xxi2c-I2cInit-LLR-008

Requirement ID: H698-LLD-CMU-FNC-2192

The function shall set CCR of i2cx with temporary register and set cr1 of i2cx with Bitwise OR of cr1 of i2cx and M\_I2C\_CR1\_PE.

##### 11.7.1.7.9 daulibstm32f4xxi2c-I2cInit-LLR-009

Requirement ID: H698-LLD-CMU-FNC-2193

The function shall configure i2cx CR1 by performing the following operations:

- Store the CR1 of i2cx value in temporary register.

- Clear ACK, SMBTYPE and SMBUS bits by performing Bitwise AND of M\_CR1\_CLEAR\_MASK with temporary register and then store it in temporary register .

- Set temporary register with Bitwise OR of temporary register with (I2C\_Mode of i2c\_init\_struct Bitwise OR I2C\_Ack of i2c\_init\_struct).

- Set CR1 of i2cx with temporary register.

##### 11.7.1.7.10 daulibstm32f4xxi2c-I2cInit-LLR-010

Requirement ID: H698-LLD-CMU-FNC-2194

The function shall set oar1 of i2cx with Bitwise OR of (I2C\_AcknowledgedAddress of i2c\_init\_struct and I2C\_OwnAddress1 of i2c\_init\_struct).

### 11.7.2 I2cCmd

Low Level Design Details about CSU I2cCmd will follow in the sub sections.

#### 11.7.2.1 Brief Description

The function I2cCmd enables or disables the specified I2C peripheral.

#### 11.7.2.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD (H698-003-012-CMU).

#### 11.7.2.3 List of global variables accessed and modified

Accessed : None

Modified : None

#### 11.7.2.4 Parameter list (Input/Output)

Inputs : T\_FUNCTIONAL\_STATE NewState - New state of the i2cx peripheral

Outputs : T\_I2C\* i2cx - Pointer to select the I2C peripheral where x can be

1, 2 or 3

#### 11.7.2.5 Return Value

None

#### 11.7.2.6 Other CSUs called by this CSU

None

#### 11.7.2.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to I2cCmd

##### 11.7.2.7.1 daulibstm32f4xxi2c-I2cCmd-LLR-001

Requirement ID: H698-LLD-CMU-FNC-2203

The function shall enable the I2C peripheral M\_I2C\_CR1\_PE by performing Bitwise OR operation of cr1 of i2cx with M\_I2C\_CR1\_PE and then store it in cr1 of i2cx when NewState is other than DISABLE.

##### 11.7.2.7.2 daulibstm32f4xxi2c-I2cCmd-LLR-002

Requirement ID: H698-LLD-CMU-FNC-2204

The function shall disable the I2C peripheral M\_I2C\_CR1\_PE by performing Bitwise AND operation of cr1 of i2cx with negation of M\_I2C\_CR1\_PE and then store it in cr1 of i2cx when NewState is equal to DISABLE.

### 11.7.3 I2cGenerateStart

Low Level Design Details about CSU I2cGenerateStart will follow in the sub sections.

#### 11.7.3.1 Brief Description

The function I2cGenerateStart generates i2cx communication START condition.

#### 11.7.3.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD (H698-003-012-CMU).

#### 11.7.3.3 List of global variables accessed and modified

Accessed : None

Modified : None

#### 11.7.3.4 Parameter list (Input/Output)

Inputs : T\_FUNCTIONAL\_STATE NewState - New state of the I2C START condition generation

(ENABLE or DISABLE)

Outputs : T\_I2C\* i2cx - Pointer to select the I2C peripheral where x can be

1, 2 or 3

#### 11.7.3.5 Return Value

None

#### 11.7.3.6 Other CSUs called by this CSU

None

#### 11.7.3.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to I2cGenerateStart.

##### 11.7.3.7.1 daulibstm32f4xxi2c-I2cGenerateStart-LLR-001

Requirement ID: H698-LLD-CMU-FNC-2213

The function shall generate a START condition (M\_I2C\_CR1\_START) by performing Bitwise OR operation of cr1 of i2cx with M\_I2C\_CR1\_START and then store it in cr1 of i2cx when NewState is other than DISABLE.

##### 11.7.3.7.2 daulibstm32f4xxi2c-I2cGenerateStart-LLR-002

Requirement ID: H698-LLD-CMU-FNC-2214

The function shall disable the START condition (M\_I2C\_CR1\_START) generation by performing Bitwise AND operation of cr1 of i2cx with negation of M\_I2C\_CR1\_START and then store it in cr1 of i2cx when NewState is equal to DISABLE.

### 11.7.4 I2cGenerateStop

Low Level Design Details about CSU I2cGenerateStop will follow in the sub sections.

#### 11.7.4.1 Brief Description

The function I2cGenerateStop generates i2cx communication STOP condition.

#### 11.7.4.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD (H698-003-012-CMU).

#### 11.7.4.3 List of global variables accessed and modified

Accessed : None

Modified : None

#### 11.7.4.4 Parameter list (Input/Output)

Inputs : T\_FUNCTIONAL\_STATE NewState - New state of the I2C STOP condition generation

(ENABLE or DISABLE)

Outputs : T\_I2C\* i2cx - Pointer to select the I2C peripheral where x can be

1, 2 or 3

#### 11.7.4.5 Return Value

None

#### 11.7.4.6 Other CSUs called by this CSU

None

#### 11.7.4.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to I2cGenerateStop.

##### 11.7.4.7.1 daulibstm32f4xxi2c-I2cGenerateStop-LLR-001

Requirement ID: H698-LLD-CMU-FNC-2223

The function shall generate a STOP condition(M\_I2C\_CR1\_STOP) by performing Bitwise OR operation of cr1 of i2cx with M\_I2C\_CR1\_STOP and then store it in cr1 of i2cx when NewState is other than DISABLE.

##### 11.7.4.7.2 daulibstm32f4xxi2c-I2cGenerateStop-LLR-002

Requirement ID: H698-LLD-CMU-FNC-2224

The function shall disable the STOP condition(M\_I2C\_CR1\_STOP) generation by performing Bitwise AND operation of cr1 of i2cx with negation of M\_I2C\_CR1\_STOP and then store it in cr1 of i2cx when NewState is equal to DISABLE.

### 11.7.5 I2cSend7bitAddress

Low Level Design Details about CSU I2cSend7bitAddress will follow in the sub sections.

#### 11.7.5.1 Brief Description

The function I2cSend7bitAddress transmits the address byte to select the slave device.

#### 11.7.5.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD (H698-003-012-CMU).

#### 11.7.5.3 List of global variables accessed and modified

Accessed : None

Modified : None

#### 11.7.5.4 Parameter list (Input/Output)

Inputs : T\_UINT8 I2C\_Direction - Specifies whether the I2C device will be a Transmitter or a

Receiver

T\_UINT8 Address - Specifies the slave address which will be transmitted

Outputs : T\_I2C\* i2cx - Pointer to select the I2C peripheral where x can be

1, 2 or 3

T\_UINT8 Address - Specifies the slave address which will be transmitted

#### 11.7.5.5 Return Value

None

#### 11.7.5.6 Other CSUs called by this CSU

None

#### 11.7.5.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to I2cSend7bitAddress.

##### 11.7.5.7.1 daulibstm32f4xxi2c-I2cSend7bitAddress-LLR-001

Requirement ID: H698-LLD-CMU-FNC-2233

The function shall set the Address for read with Bitwise OR operation of Address and M\_I2C\_OAR1\_ADD0 when I2C\_Direction is other than M\_I2C\_DIRECTION\_TRANSMITTER (i.e Receiver mode).

##### 11.7.5.7.2 daulibstm32f4xxi2c-I2cSend7bitAddress-LLR-002

Requirement ID: H698-LLD-CMU-FNC-2234

The function shall reset the Address for write with Bitwise AND operation of Address and negation of M\_I2C\_OAR1\_ADD0 when I2C\_Direction is equal to M\_I2C\_DIRECTION\_TRANSMITTER.

##### 11.7.5.7.3 daulibstm32f4xxi2c-I2cSend7bitAddress-LLR-003

Requirement ID: H698-LLD-CMU-FNC-2235

The function shall send the Address to I2C Data register (dr of i2cx).

### 11.7.6 I2cAcknowledgeConfig

Low Level Design Details about CSU I2cAcknowledgeConfig will follow in the sub sections.

#### 11.7.6.1 Brief Description

The function I2cAcknowledgeConfig enables or disables the specified I2C acknowledge feature.

#### 11.7.6.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD (H698-003-012-CMU).

#### 11.7.6.3 List of global variables accessed and modified

Accessed : None

Modified : None

#### 11.7.6.4 Parameter list (Input/Output)

Inputs : T\_FUNCTIONAL\_STATE NewState - New state of the I2C Acknowledgement

(ENABLE or DISABLE)

Outputs : T\_I2C\* i2cx - Pointer to select the I2C peripheral where x can be

1, 2 or 3

#### 11.7.6.5 Return Value

None

#### 11.7.6.6 Other CSUs called by this CSU

None

#### 11.7.6.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to I2cAcknowledgeConfig.

##### 11.7.6.7.1 daulibstm32f4xxi2c-I2cAcknowledgeConfig-LLR-001

Requirement ID: H698-LLD-CMU-FNC-2244

The function shall enable the acknowledgeacknowledgement(M\_I2C\_CR1\_ACK) by setting cr1 of i2cx with Bitwise OR operation of cr1 of i2cx and M\_I2C\_CR1\_ACK when NewState is other than DISABLE.

##### 11.7.6.7.2 daulibstm32f4xxi2c-I2cAcknowledgeConfig-LLR-002

Requirement ID: H698-LLD-CMU-FNC-2245

The function shall disable the acknowledgement by setting cr1 of i2cx with Bitwise AND operation of cr1 of i2cx and negation of M\_I2C\_CR1\_ACK when NewState is equal to DISABLE.

### 11.7.7 I2cSendData

Low Level Design Details about CSU I2cSendData will follow in the sub sections.

#### 11.7.7.1 Brief Description

The function I2cSendData sends a data byte through the i2cx peripheral.

#### 11.7.7.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD (H698-003-012-CMU).

#### 11.7.7.3 List of global variables accessed and modified

Accessed : None

Modified : None

#### 11.7.7.4 Parameter list (Input/Output)

Inputs : T\_UINT8 Data - Byte to be transmitted

Outputs : T\_I2C\* i2cx - Pointer to select the I2C peripheral where x can be

1, 2 or 3

#### 11.7.7.5 Return Value

None

#### 11.7.7.6 Other CSUs called by this CSU

None

#### 11.7.7.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to I2cSendData.

##### 11.7.7.7.1 daulibstm32f4xxi2c-I2cSendData-LLR-001

Requirement ID: H698-LLD-CMU-FNC-2254

The function shall write in the DR register(dr of i2cx) the Data to be sent.

### 11.7.8 I2cReceiveData

Low Level Design Details about CSU I2cReceiveData will follow in the sub sections.

#### 11.7.8.1 Brief Description

The function I2cReceiveData returns the most recent received data by the i2cx peripheral.

#### 11.7.8.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD (H698-003-012-CMU).

#### 11.7.8.3 List of global variables accessed and modified

Accessed : None

Modified : None

#### 11.7.8.4 Parameter list (Input/Output)

Inputs : T\_I2C\* i2cx - Pointer to select the I2C peripheral where x can be

1, 2 or 3

Outputs : None

#### 11.7.8.5 Return Value

T\_UINT8 - The value of the received data.

#### 11.7.8.6 Other CSUs called by this CSU

None

#### 11.7.8.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to I2cReceiveData

##### 11.7.8.7.1 daulibstm32f4xxi2c-I2cReceiveData-LLR-001

Requirement ID: H698-LLD-CMU-FNC-2263

The function shall return the Data in the DR register (dr of i2cx).

### 11.7.9 I2cITConfig

Low Level Design Details about CSU I2cITConfig will follow in the sub sections.

#### 11.7.9.1 Brief Description

The function I2cITConfig enables or disables the specified I2C interrupts.

#### 11.7.9.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD (H698-003-012-CMU).

#### 11.7.9.3 List of global variables accessed and modified

Accessed : None

Modified : None

#### 11.7.9.4 Parameter list (Input/Output)

Inputs : T\_FUNCTIONAL\_STATE NewState - New state of the specified I2C interrupts

T\_UINT16 I2C\_IT - specifies the I2C interrupts sources to be enabled

or disabled

Outputs : T\_I2C\* i2cx - Pointer to select the I2C peripheral where x can be

1, 2 or 3

#### 11.7.9.5 Return Value

None

#### 11.7.9.6 Other CSUs called by this CSU

None

#### 11.7.9.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to I2cITConfig.

##### 11.7.9.7.1 daulibstm32f4xxi2c-I2cITConfig-LLR-001

Requirement ID: H698-LLD-CMU-FNC-2272

The function shall set cr2 of i2cx to Bitwise OR of cr2 of i2cx and I2C\_IT when NewState is other than DISABLE.

##### 11.7.9.7.2 daulibstm32f4xxi2c-I2cITConfig-LLR-002

Requirement ID: H698-LLD-CMU-FNC-2273

The function shall set cr2 of i2cx to Bitwise AND of cr2 of i2cx and negation of I2C\_IT when NewState is equal to DISABLE.

### 11.7.10 I2cGetLastEvent

Low Level Design Details about CSU I2cGetLastEvent will follow in the sub sections.

#### 11.7.10.1 Brief Description

The function I2cGetLastEvent returns the last i2cx Event.

#### 11.7.10.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD (H698-003-012-CMU).

#### 11.7.10.3 List of global variables accessed and modified

Accessed : None

Modified : None

#### 11.7.10.4 Parameter list (Input/Output)

Inputs : T\_I2C\* i2cx - Pointer to select the I2C peripheral where x can be

1, 2 or 3

Outputs : None

#### 11.7.10.5 Return Value

T\_UINT32 - The last event.

#### 11.7.10.6 Other CSUs called by this CSU

None

#### 11.7.10.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to I2cGetLastEvent.

##### 11.7.10.7.1 daulibstm32f4xxi2c-I2cGetLastEvent-LLR-001

Requirement ID: H698-LLD-CMU-FNC-2282

The function shall read the i2cx status register by performing the following:

-Set flag1 to sr1 of i2cx

-Set flag2 to sr2 of i2cx

-Set flag2 as Bitshift of flag2 to left by M\_SHIFT\_16

##### 11.7.10.7.2 daulibstm32f4xxi2c-I2cGetLastEvent-LLR-002

Requirement ID: H698-LLD-CMU-FNC-2283

The function shall set the last event as Bitwise AND of (Bitwise OR of flag1 and flag2) and M\_FLAG\_MASK.

##### 11.7.10.7.3 daulibstm32f4xxi2c-I2cGetLastEvent-LLR-003

Requirement ID: H698-LLD-CMU-FNC-2284

The function shall return the last event.

### 11.7.11 I2cGetITStatus

Low Level Design Details about CSU I2cGetITStatus will follow in the sub sections.

#### 11.7.11.1 Brief Description

The function I2cGetITStatus checks whether the specified I2C interrupt has occurred or not.

#### 11.7.11.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD (H698-003-012-CMU).

#### 11.7.11.3 List of global variables accessed and modified

Accessed : None

Modified : None

#### 11.7.11.4 Parameter list (Input/Output)

Inputs : T\_I2C\* i2cx - Pointer to select the I2C peripheral where x can be

1, 2 or 3

T\_UINT32 I2C\_IT - Specifies the interrupt source to check

Outputs :T\_UINT32 I2C\_IT - Specifies the interrupt source to check

#### 11.7.11.5 Return Value

T\_ITSTATUS - The new state of I2C\_IT (SET or RESET)

#### 11.7.11.6 Other CSUs called by this CSU

None

#### 11.7.11.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to I2cGetITStatus

##### 11.7.11.7.1 daulibstm32f4xxi2c-I2cGetITStatus-LLR-001

Requirement ID: H698-LLD-CMU-FNC-2293

The function shall set enable status as Bitwise AND of ((I2C\_IT Bitwise AND M\_ITEN\_MASK) right shift by sixteen) and cr2 of i2cx.

##### 11.7.11.7.2 daulibstm32f4xxi2c-I2cGetITStatus-LLR-002

Requirement ID: H698-LLD-CMU-FNC-2294

The function shall set bit[23:0] of the flag by performing Bitwise AND operation of I2C\_IT with M\_FLAG\_MASK and then store it in I2C\_IT.

##### 11.7.11.7.3 daulibstm32f4xxi2c-I2cGetITStatus-LLR-003

Requirement ID: H698-LLD-CMU-FNC-2295

The function shall set bit status to SET when ((sr1 of i2cx Bitwise AND with I2C\_IT) is not equal to (RESET AND with enable status)) returns true.

##### 11.7.11.7.4 daulibstm32f4xxi2c-I2cGetITStatus-LLR-004

Requirement ID: H698-LLD-CMU-FNC-2296

The function shall set bit status to RESET when ((sr1 of i2cx Bitwise AND I2C\_IT) is not equal to (RESET AND enable status)) returns false.

##### 11.7.11.7.5 daulibstm32f4xxi2c-I2cGetITStatus-LLR-005

Requirement ID: H698-LLD-CMU-FNC-2297

The function shall return the bit status.

### 11.7.12 I2cClearITPendingBit

Low Level Design Details about CSU I2cClearITPendingBit will follow in the sub sections.

#### 11.7.12.1 Brief Description

The function I2cClearITPendingBit clears the i2cx's interrupt pending bits.

#### 11.7.12.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD (H698-003-012-CMU).

#### 11.7.12.3 List of global variables accessed and modified

Accessed : None

Modified : None

#### 11.7.12.4 Parameter list (Input/Output)

Inputs : T\_UINT32 I2C\_IT - Specifies the interrupt pending bit to clear

Outputs : T\_I2C\* i2cx - Pointer to select the I2C peripheral where x can be 1, 2 or 3

#### 11.7.12.5 Return Value

None

#### 11.7.12.6 Other CSUs called by this CSU

None

#### 11.7.12.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to I2cClearITPendingBit.

##### 11.7.12.7.1 daulibstm32f4xxi2c-I2cClearITPendingBit-LLR-001

Requirement ID: H698-LLD-CMU-FNC-2306

The function shall set the flag position by performing Bitwise AND operation of I2C\_IT with M\_FLAG\_MASK.

##### 11.7.12.7.2 daulibstm32f4xxi2c-I2cClearITPendingBit-LLR-002

Requirement ID: H698-LLD-CMU-FNC-2307

The function shall clear the I2C flag by setting sr1 of i2cx with negation of flag position.

## 11.8 daulibstm32f4xxiwdg

This module provides firmware functions to manage the following functionality of the Independent watchdog (IWDG) peripheral:

* Prescaler and Counter configuration
* IWDG activation

### 11.8.1 IwdgWriteAccessCmd

Low Level Design Details about CSU IwdgWriteAccessCmd will follow in the sub sections.

#### 11.8.1.1 Brief Description

The function IwdgWriteAccessCmd enables or disables write access to IWDG\_PR and IWDG\_RLR registers.

#### 11.8.1.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-012-CMU).

#### 11.8.1.3 List of global variables accessed and modified

Accessed: None

Modified: None

#### 11.8.1.4 Parameter list (Input/Output)

Inputs: T\_UINT16 iwdg\_write\_access – New state of write access to IWDG\_PR and IWDG\_RLR registers.

Outputs: None

#### 11.8.1.5 Return Value

None

#### 11.8.1.6 Other CSUs called by this CSU

None

#### 11.8.1.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to IwdgWriteAccessCmd.

##### 11.8.1.7.1 daulibstm32f4xxiwdg-IwdgWriteAccessCmd-LLR-001

Requirement ID: H698-LLD-CMU-FNC-1712

The function shall configure IWDG Key register with iwdg\_write\_access to enable or disable access to the IWDG\_PR and IWDG\_RLR registers.

### 11.8.2 IwdgSetPrescaler

Low Level Design Details about CSU IwdgSetPrescaler will follow in the sub sections.

#### 11.8.2.1 Brief Description

The function IwdgSetPrescaler sets IWDG prescaler value.

#### 11.8.2.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-012-CMU).

#### 11.8.2.3 List of global variables accessed and modified

Accessed: None

Modified: None

#### 11.8.2.4 Parameter list (Input/Output)

Inputs: T\_UINT8 iwdg\_prescaler - Specifies the IWDG Prescaler value.

Outputs: None

#### 11.8.2.5 Return Value

None

#### 11.8.2.6 Other CSUs called by this CSU

None

#### 11.8.2.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to IwdgSetPrescaler.

##### 11.8.2.7.1 daulibstm32f4xxiwdg-IwdgSetPrescaler-LLR-001

Requirement ID: H698-LLD-CMU-FNC-1721

The function shall configure IWDG Prescaler register (pr) with prescaler value iwdg\_prescaler i.e, set pr of M\_IWDG to iwdg\_prescaler.

### 11.8.3 IwdgSetReload

Low Level Design Details about CSU IwdgSetReloadwill follow in the sub sections.

#### 11.8.3.1 Brief Description

The function IwdgSetReload sets IWDG reload value.

#### 11.8.3.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-012-CMU).

#### 11.8.3.3 List of global variables accessed and modified

Accessed: None

Modified: None

#### 11.8.3.4 Parameter list (Input/Output)

Inputs: T\_UINT16 reload - specifies the IWDG Reload value. This parameter must be a number between 0 and 0x0FFF

Outputs: None

#### 11.8.3.5 Return Value

None

#### 11.8.3.6 Other CSUs called by this CSU

None

#### 11.8.3.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to IwdgSetReload.

##### 11.8.3.7.1 daulibstm32f4xxiwdg-IwdgSetReload-LLR-001

Requirement ID: H698-LLD-CMU-FNC-1730

The function shall configure IWDG Reload register (rlr) with reload value i.e, set rlr of M\_IWDG to reload.

### 11.8.4 IwdgReloadCounter

Low Level Design Details about CSU IwdgReloadCounter will follow in the sub sections.

#### 11.8.4.1 Brief Description

The function IwdgReloadCounter reloads IWDG counter with value defined in the reload register.

#### 11.8.4.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-012-CMU).

#### 11.8.4.3 List of global variables accessed and modified

Accessed: None

Modified: None

#### 11.8.4.4 Parameter list (Input/Output)

Inputs: None

Outputs: None

#### 11.8.4.5 Return Value

None

#### 11.8.4.6 Other CSUs called by this CSU

None

#### 11.8.4.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to IwdgReloadCounter.

##### 11.8.4.7.1 daulibstm32f4xxiwdg-IwdgReloadCounter-LLR-001

Requirement ID: H698-LLD-CMU-FNC-1739

The function shall reload the watchdog counter with the reload value in reload register (IWDG\_RLR) i.e, set kr of M\_IWDG to M\_KR\_KEY\_RELOAD.

### 11.8.5 IwdgEnable

Low Level Design Details about CSU IwdgEnable will follow in the sub sections.

#### 11.8.5.1 Brief Description

The function IwdgEnable enables IWDG (write access to IWDG\_PR and IWDG\_RLR registers disabled).

#### 11.8.5.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-012-CMU).

#### 11.8.5.3 List of global variables accessed and modified

Accessed: None

Modified: None

#### 11.8.5.4 Parameter list (Input/Output)

Inputs: None

Outputs: None

#### 11.8.5.5 Return Value

None

#### 11.8.5.6 Other CSUs called by this CSU

None

#### 11.8.5.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to IwdgEnable

##### 11.8.5.7.1 daulibstm32f4xxiwdg-IwdgEnable-LLR-001

Requirement ID: H698-LLD-CMU-FNC-1748

The function shall configure IWDG Key register with the value M\_KR\_KEY\_ENABLE to start the watchdog i.e, set kr of M\_IWDG to M\_KR\_KEY\_ENABLE.

## 11.9 daulibstm32f4xxpwr

This module provides firmware functions to manage the following functionality of the Power Controller (PWR) peripheral:

Main Regulator configuration

### 11.9.1 PwrMainRegulatorModeConfig

Low Level Design Details about CSU PwrMainRegulatorModeConfig will follow in the sub sections.

#### 11.9.1.1 Brief Description

The function PwrMainRegulatorModeConfig configures the main internal regulator output voltage.

#### 11.9.1.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-012-CMU).

#### 11.9.1.3 List of global variables accessed and modified

Accessed: None

Modified: None

#### 11.9.1.4 Parameter list (Input/Output)

Inputs: T\_UINT32 pwr\_regulator\_voltage - Specifies the regulator output voltage to achieve a tradeoff between performance and power consumption when the device does not operate at the maximum frequency.

Outputs: None

#### 11.9.1.5 Return Value

None

#### 11.9.1.6 Other CSUs called by this CSU

None

#### 11.9.1.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to PwrMainRegulatorModeConfig

##### 11.9.1.7.1 daulibstm32f4xxpwr-PwrMainRegulatorModeConfig-LLR-001

Requirement ID: H698-LLD-CMU-FNC-1758

The function shall configure power control register (PWR\_CR) to Scale 2 mode (bit 14) when scale 2 is seleted i.e, set cr of M\_PWR to (cr of M\_PWR bitwise AND with negated value of M\_PWR\_REGULATOR\_VOLTAGE\_SCALE1) when M\_PWR\_REGULATOR\_VOLTAGE\_SCALE2 is equal to pwr\_regulator\_voltage.

##### 11.9.1.7.2 daulibstm32f4xxpwr-PwrMainRegulatorModeConfig-LLR-002

Requirement ID: H698-LLD-CMU-FNC-1759

The function shall configure power control register (PWR\_CR) to Scale 1 mode (bit 14) when scale 1 is seleted i.e, set cr of M\_PWR to (cr of M\_PWR bitwise OR with M\_PWR\_REGULATOR\_VOLTAGE\_SCALE1) when M\_PWR\_REGULATOR\_VOLTAGE\_SCALE2 is not equal to pwr\_regulator\_voltage.

## 11.10 daulibstm32f4xxrcc

The daulibstm32f4xxrcc CSC provides firmware functions to manage the following functionalities of the Reset and clock control (RCC) peripheral:

* Internal/external clocks, PLL, CSS and MCO configuration
* System, AHB and APB busses clocks configuration
* Peripheral clocks configuration
* Interrupts and flags management

### 11.10.1 RccDeInit

Low Level Design Details about CSU RccDeInit will follow in the sub sections.

#### 11.10.1.1 Brief Description

The function RccDeInit Resets the RCC clock configuration to the default reset state.

#### 11.10.1.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-012-CMU).

#### 11.10.1.3 List of global variables accessed and modified

Accessed: None

Modified: None

#### 11.10.1.4 Parameter list (Input/Output)

Inputs: None

Outputs: None

#### 11.10.1.5 Return Value

None

#### 11.10.1.6 Other CSUs called by this CSU

None

#### 11.10.1.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to RccDeInit

##### 11.10.1.7.1 daulibstm32f4xxrcc-RccDeInit-LLR-001

Requirement ID: H698-LLD-CMU-FNC-1769

The function shall set the HSION bit in RCC clock control register (cr).(i.e., cr of M\_RCC Bitwise OR with M\_HEX\_ONE)

##### 11.10.1.7.2 daulibstm32f4xxrcc-RccDeInit-LLR-002

Requirement ID: H698-LLD-CMU-FNC-1770

The function shall reset the cfgr register of RCC (i.e., set cfgr of M\_RCC to M\_HEX\_ZERO)

##### 11.10.1.7.3 daulibstm32f4xxrcc-RccDeInit-LLR-003

Requirement ID: H698-LLD-CMU-FNC-1771

The function shall reset HSEON, CSSON and PLLON bits of RCC clock control register (cr).(i.e., cr of M\_RCC Bitwise AND with M\_RESET\_HSEON\_CSSON\_PLLON)

##### 11.10.1.7.4 daulibstm32f4xxrcc-RccDeInit-LLR-004

Requirement ID: H698-LLD-CMU-FNC-1772

The function shall reset pllcfgr register of RCC.(i.e., pllcfgr of M\_RCC to M\_RESET\_PLLCFGR)

##### 11.10.1.7.5 daulibstm32f4xxrcc-RccDeInit-LLR-005

Requirement ID: H698-LLD-CMU-FNC-1773

The function shall reset HSEBYP bit of RCC clock control register (cr).(i.e., cr of M\_RCC Bitwise AND with M\_RESET\_HSEBYP)

##### 11.10.1.7.6 daulibstm32f4xxrcc-RccDeInit-LLR-006

Requirement ID: H698-LLD-CMU-FNC-1774

The function shall disable all interrupts by resetting RCC clock interrupt register (cir).(i.e., cir of M\_RCC to M\_HEX\_ZERO)

### 11.10.2 RccHseConfig

Low Level Design Details about CSU RccHseConfig will follow in the sub sections.

#### 11.10.2.1 Brief Description

The function RccHseConfig configures the External High Speed oscillator (HSE).

#### 11.10.2.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-012-CMU).

#### 11.10.2.3 List of global variables accessed and modified

Accessed: None

Modified: None

#### 11.10.2.4 Parameter list (Input/Output)

Inputs: T\_UINT8 rcc\_hse - Specifies the new state of the HSE

Outputs: None

#### 11.10.2.5 Return Value

None

#### 11.10.2.6 Other CSUs called by this CSU

None

#### 11.10.2.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to RccHseConfig.

##### 11.10.2.7.1 daulibstm32f4xxrcc-RccHseConfig-LLR-001

Requirement ID: H698-LLD-CMU-FNC-1783

The function shall reset HSEON and HSEBYP bits in 3rd byte of RCC clock control register (cr) and configure with new configuration rcc\_hse.(i.e., pointer to M\_CR\_BYTE3\_ADDRESS to M\_RCC\_HSE\_OFF and pointer to M\_CR\_BYTE3\_ADDRESS to rcc\_hse).

### 11.10.3 RccWaitForHseStartUp

Low Level Design Details about CSU RccWaitForHseStartUp will follow in the sub sections.

#### 11.10.3.1 Brief Description

The function RccWaitForHseStartUp waits for HSE start-up.

#### 11.10.3.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-012-CMU).

#### 11.10.3.3 List of global variables accessed and modified

Accessed: None

Modified: None

#### 11.10.3.4 Parameter list (Input/Output)

Inputs: None

Outputs: None

#### 11.10.3.5 Return Value

T\_ERROR\_STATUS - Returns error status.

#### 11.10.3.6 Other CSUs called by this CSU

RccGetFlagStatus

#### 11.10.3.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to RccWaitForHseStartUp.

##### 11.10.3.7.1 daulibstm32f4xxrcc-RccWaitForHseStartUp-LLR-001

Requirement ID: H698-LLD-CMU-FNC-1792

The function shall wait till HSE is ready and increments the counter. When Time out is reached as below:

a) Loop till the loop count is not equal to M\_HSE\_STARTUP\_TIMEOUT AND

b) Return value of the function RccGetFlagStatus called with the parameter M\_RCC\_FLAG\_HSERDY is equal to RESET.

##### 11.10.3.7.2 daulibstm32f4xxrcc-RccWaitForHseStartUp-LLR-002

Requirement ID: H698-LLD-CMU-FNC-1793

The function shall return SUCCESS when RCC flag is SET i.e return value of the function RccGetFlagStatus with parameter M\_RCC\_FLAG\_HSERDY is not equal to RESET.

##### 11.10.3.7.3 daulibstm32f4xxrcc-RccWaitForHseStartUp-LLR-003

Requirement ID: H698-LLD-CMU-FNC-1794

The function shall do nothing if RCC flag is reset i.e. return value of the function RccGetFlagStatus with parameter M\_RCC\_FLAG\_HSERDY is RESET.

### 11.10.4 RccPllConfig

Low Level Design Details about CSU RccPllConfig will follow in the sub sections.

#### 11.10.4.1 Brief Description

The function RccPllConfig configures the main PLL clock source, multiplication and division factors.

#### 11.10.4.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-012-CMU).

#### 11.10.4.3 List of global variables accessed and modified

Accessed: None

Modified: None

#### 11.10.4.4 Parameter list (Input/Output)

Inputs: T\_UINT32 rcc\_pll\_source - Specifies the PLL entry clock source.

T\_UINT32 pll\_m - Specifies the division factor for PLL VCO input clock

T\_UINT32 pll\_n - Specifies the multiplication factor for PLL VCO output clock

T\_UINT32 pll\_p - Specifies the division factor for main system clock (SYSCLK)

T\_UINT32 pll\_q - Specifies the division factor for OTG FS, SDIO and RNG clocks

Outputs: None

#### 11.10.4.5 Return Value

None

#### 11.10.4.6 Other CSUs called by this CSU

None

#### 11.10.4.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to RccPllConfig.

##### 11.10.4.7.1 daulibstm32f4xxrcc-RccPllConfig-LLR-001

Requirement ID: H698-LLD-CMU-FNC-1803

The function shall configure RCC PLL configuration register with pll\_m, pll\_n, pll\_p, pll\_q and rcc\_pll\_source.(i.e., pllcfgr of M\_RCC to logical Bitwise OR with pll\_m, (pll\_n left shift of M\_SHIFT\_6), (pll\_p right shift of M\_SHIFT\_1) minus M\_ONE left shift of M\_SHIFT\_16, rcc\_pll\_source, (pll\_q left shift of M\_SHIFT\_24))

### 11.10.5 RccPllCmd

Low Level Design Details about CSU RccPllCmd will follow in the sub sections.

#### 11.10.5.1 Brief Description

The function RccPllCmd enables or disables the main PLL.

#### 11.10.5.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-012-CMU).

#### 11.10.5.3 List of global variables accessed and modified

Accessed: None

Modified: None

#### 11.10.5.4 Parameter list (Input/Output)

Inputs: T\_FUNCTIONAL\_STATE new\_state - New state of the main PLL.

Outputs: None

#### 11.10.5.5 Return Value

None

#### 11.10.5.6 Other CSUs called by this CSU

None

#### 11.10.5.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to RccPllCmd.

##### 11.10.5.7.1 daulibstm32f4xxrcc-RccPllCmd-LLR-001

Requirement ID: H698-LLD-CMU-FNC-1812

The function shall configure PLLON bit of RCC clock control register with new\_state(i.e., pointer to M\_CR\_PLLON\_BB is set to new state).

### 11.10.6 RccSysClkConfig

Low Level Design Details about CSU RccSysClkConfig will follow in the sub sections.

#### 11.10.6.1 Brief Description

The function RccSysClkConfig configures the system clock (SYSCLK).

#### 11.10.6.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-012-CMU).

#### 11.10.6.3 List of global variables accessed and modified

Accessed: None

Modified: None

#### 11.10.6.4 Parameter list (Input/Output)

Inputs: T\_UINT32 rcc\_sysclk\_source - specifies the clock source used as system clock.

Outputs: None

#### 11.10.6.5 Return Value

None

#### 11.10.6.6 Other CSUs called by this CSU

None

#### 11.10.6.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to RccSysClkConfig.

##### 11.10.6.7.1 daulibstm32f4xxrcc-RccSysClkConfig-LLR-001

Requirement ID: H698-LLD-CMU-FNC-1821

The function shall clear the System clock switch bits (bit 0 and 1) in RCC clock configuration register (cfgr) and configure with clock source used as system clock (rcc\_sysclk\_source)(i.e., sets the temporary register to cfgr of M\_RCC, Bitwise AND with not of M\_RCC\_CFGR\_SW, Bitwise OR with rcc\_sysclk\_source).

And store the new value(i.e., sets the cfgr of M\_RCC to temporary register).

### 11.10.7 RccGetSysClkSource

Low Level Design Details about CSU RccGetSysClkSource will follow in the sub sections.

#### 11.10.7.1 Brief Description

The function RccGetSysClkSource returns the clock source used as system clock.

#### 11.10.7.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-012-CMU).

#### 11.10.7.3 List of global variables accessed and modified

Accessed: None

Modified: None

#### 11.10.7.4 Parameter list (Input/Output)

Inputs: None

Outputs: None

#### 11.10.7.5 Return Value

T\_UINT8 - Returns the clock source used as system clock.

#### 11.10.7.6 Other CSUs called by this CSU

None

#### 11.10.7.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to RccGetSysClkSource.

##### 11.10.7.7.1 daulibstm32f4xxrcc-RccGetSysClkSource-LLR-001

Requirement ID: H698-LLD-CMU-FNC-1830

The function shall return the clock source used as system clock by extracting the SWS0 and SWS1 bits in RCC clock configuration register (cfgr).(i.e., returns the cfgr of M\_RCC Bitwise AND with M\_RCC\_CFGR\_SWS)

### 11.10.8 RccHclkConfig

Low Level Design Details about CSU RccHclkConfig will follow in the sub sections.

#### 11.10.8.1 Brief Description

The function RccHclkConfig configures the AHB clock (HCLK).

#### 11.10.8.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-012-CMU).

#### 11.10.8.3 List of global variables accessed and modified

Accessed: None

Modified: None

#### 11.10.8.4 Parameter list (Input/Output)

Inputs: T\_UINT32 rcc\_sysclk - Defines the AHB clock divider. This clock is derived from the system clock (SYSCLK).

Outputs: None

#### 11.10.8.5 Return Value

None

#### 11.10.8.6 Other CSUs called by this CSU

None

#### 11.10.8.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to RccHclkConfig.

##### 11.10.8.7.1 daulibstm32f4xxrcc-RccHclkConfig-LLR-001

Requirement ID: H698-LLD-CMU-FNC-1839

The function shall clear the AHB prescaler bits (bit 4 to 7) in RCC clock configuration register (cfgr) and configure with AHB clock divider rcc\_sysclk.(i.e., sets the temporary register to cfgr of M\_RCC, Bitwise AND with not of M\_RCC\_CFGR\_HPRE, Bitwise OR with rcc\_sysclk).

And stores the new value(i.e., cfgr of M\_RCC to temporary register)

### 11.10.9 RccPclk1Config

Low Level Design Details about CSU RccPclk1Config will follow in the sub sections.

#### 11.10.9.1 Brief Description

The function RccPclk1Config configures the Low Speed APB clock (PCLK1).

#### 11.10.9.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-012-CMU).

#### 11.10.9.3 List of global variables accessed and modified

Accessed: None

Modified: None

#### 11.10.9.4 Parameter list (Input/Output)

Inputs: T\_UINT32 rcc\_hclk - Defines the APB1 clock divider. This clock is derived from the AHB clock (HCLK).

Outputs: None

#### 11.10.9.5 Return Value

None

#### 11.10.9.6 Other CSUs called by this CSU

None

#### 11.10.9.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to RccPclk1Config.

##### 11.10.9.7.1 daulibstm32f4xxrcc-RccPclk1Config-LLR-001

Requirement ID: H698-LLD-CMU-FNC-1848

The function shall clear the APB Low speed prescaler bits (bit 10 to 12) in RCC clock configuration register (cfgr) and configure with APB1 clock divider rcc\_hclk.(i.e., sets the temporary register to cfgr of M\_RCC, Bitwise AND with not of M\_RCC\_CFGR\_PPRE1, Bitwise OR with rcc\_hclk ).

And stores the new value(i.e., cfgr of M\_RCC to temporary register).

### 11.10.10 RccPclk2Config

Low Level Design Details about CSU RccPclk2Config will follow in the sub sections.

#### 11.10.10.1 Brief Description

The function RccPclk2Config configures the High Speed APB clock (PCLK2).

#### 11.10.10.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-012-CMU).

#### 11.10.10.3 List of global variables accessed and modified

Accessed: None

Modified: None

#### 11.10.10.4 Parameter list (Input/Output)

Inputs: T\_UINT32 rcc\_hclk - Defines the APB2 clock divider. This clock is derived from the AHB clock (HCLK).

Outputs: None

#### 11.10.10.5 Return Value

None

#### 11.10.10.6 Other CSUs called by this CSU

None

#### 11.10.10.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to RccPclk2Config.

##### 11.10.10.7.1 daulibstm32f4xxrcc-RccPclk2Config-LLR-001

Requirement ID: H698-LLD-CMU-FNC-1857

The function shall clear the APB high-speed prescaler bits (bit 13 to 15) in RCC clock configuration register(cfgr) and write APB2 clock divider 'rcc\_hclk' value into APB high-speed prescaler bits of cfgr.(i.e., sets the temporary register to cfgr of M\_RCC, Bitwise AND with not of M\_RCC\_CFGR\_PPRE2, Bitwise OR with rcc\_hclk left shift of M\_SHIFT\_3)

And stores the new value(cfgr of M\_RCC to temporary register).

### 11.10.11 RccGetClocksFreq

Low Level Design Details about CSU RccGetClocksFreq will follow in the sub sections.

#### 11.10.11.1 Brief Description

The function RccGetClocksFreq gets the clock frequency.

#### 11.10.11.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-012-CMU).

#### 11.10.11.3 List of global variables accessed and modified

Accessed: None

Modified: None

#### 11.10.11.4 Parameter list (Input/Output)

Inputs: T\_RCC\_CLOCKS\* rcc\_clocks - System clock source.

Outputs: T\_RCC\_CLOCKS\* rcc\_clocks - System clock source.

#### 11.10.11.5 Return Value

None

#### 11.10.11.6 Other CSUs called by this CSU

None

#### 11.10.11.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to RccGetClocksFreq.

##### 11.10.11.7.1 daulibstm32f4xxrcc-RccGetClocksFreq-LLR-001

Requirement ID: H698-LLD-CMU-FNC-1866

The function shall find the sysclk source by extracting 4 LSB bits of RCC clock configuration register (cfgr)

(i.e., cfgr of M\_RCC Bitwise AND with M\_RCC\_CFGR\_SWS)

##### 11.10.11.7.2 daulibstm32f4xxrcc-RccGetClocksFreq-LLR-002

Requirement ID: H698-LLD-CMU-FNC-1867

The function shall set SYSCLK frequency of Rcc clocks to M\_HSI\_VALUE when sysclk source value is M\_HEX\_ZERO.

##### 11.10.11.7.3 daulibstm32f4xxrcc-RccGetClocksFreq-LLR-003

Requirement ID: H698-LLD-CMU-FNC-1868

The function shall set SYSCLK frequency of Rcc clocks to M\_HSE\_VALUE when sysclk source value is M\_HEX\_FOUR.

##### 11.10.11.7.4 daulibstm32f4xxrcc-RccGetClocksFreq-LLR-004

Requirement ID: H698-LLD-CMU-FNC-1869

The function shall do the following when sysclk source value is M\_HEX\_EIGHT:

a) Extract pll source(PLLSRC bit) from RCC PLL configuration register(pllcfgr)(i.e., pllcfgr of M\_RCC Bitwise AND with M\_RCC\_PLLCFGR\_PLLSRC right shift of M\_SHIFT\_22).

b) Extract pll\_m (pllm bits 0 to 5) from RCC PLL configuration register(pllcfgr)(i.e., pllcfgr of M\_RCC Bitwise AND with M\_RCC\_PLLCFGR\_PLLM).

c) when pll source is not M\_ZERO, compute the VCO output frequency using HSE as PLL clock source(i.e., sets the pllvco to (M\_HSE\_VALUE divide by pllm) multiply with (pllcfgr of M\_RCC Bitwise AND with M\_RCC\_PLLCFGR\_PLLN) right shift of M\_SHIFT\_6)

d) when pll source is M\_ZERO, compute the VCO output frequency using HSI as PLL clock source(i.e., set the pllvco to (M\_HSI\_VALUE divide by pllm) multiply with (pllcfgr of M\_RCC Bitwise AND with M\_RCC\_PLLCFGR\_PLLN) right shift of M\_SHIFT\_6)

e) Compute PLL output clock frequency using VCO output frequency and set it to Sysclk\_frequency of rcc\_clocks(i.e., sets the pllp to (((pllcfgr of M\_RCC Bitwise AND with M\_RCC\_PLLCFGR\_PLLP) right shift of M\_SHIFT\_16) plus M\_ONE) multiply with M\_TWO and sets the Sysclk\_frequency of rcc\_clocks to pllvco divide by pllp).

##### 11.10.11.7.5 daulibstm32f4xxrcc-RccGetClocksFreq-LLR-005

Requirement ID: H698-LLD-CMU-FNC-1870

The function shall set SYSCLK frequency of Rcc clocks with M\_HSI\_VALUE if sysclk source value is other than M\_HEX\_ZERO, M\_HEX\_FOUR and M\_HEX\_EIGHT.

##### 11.10.11.7.6 daulibstm32f4xxrcc-RccGetClocksFreq-LLR-006

Requirement ID: H698-LLD-CMU-FNC-1871

The function shall do the following:

a) Extract the HPRE bit from RCC clock configuration register (cfgr) i.e, sets the sysclk source to cfgr of M\_RCC Bitwise AND with M\_RCC\_CFGR\_HPRE

b) Get the corresponding prescaler value from Apbahb prescaler table with index of sysclk source (sysclk source is right shift of M\_SHIFT\_4).

c) Set the hclk\_frequency of rcc\_clocks to sysclk\_frequency of rcc\_clocks right shifted by prescaler (extracted using prescaler value obtained Apbahb\_presc\_table with index as sysclk source).

##### 11.10.11.7.7 daulibstm32f4xxrcc-RccGetClocksFreq-LLR-007

Requirement ID: H698-LLD-CMU-FNC-1872

The function shall do the following:

a) Extract the PPRE1 bit from RCC clock configuration register (cfgr) i.e, sets the sysclk source to cfgr of M\_RCC Bitwise AND with M\_RCC\_CFGR\_PPRE1 and right shifted by M\_SHIFT\_10.

b) Get the corresponding prescaler value from Apbahb prescaler table with index as sysclk source.

c) Set the pclk\_1\_frequency of rcc\_clocks to hclk\_frequency of rcc\_clocks right shifted by prescaler.

d) Extract the PPRE2 bit from RCC clock configuration register(cfgr)(i.e., sets the sysclk src to cfgr of M\_RCC Bitwise AND M\_RCC\_CFGR\_PPRE2 and right shift of M\_SHIFT\_13)

e) Get the corresponding prescaler value from Apbahb prescaler table with index as sysclk source.

f) Set the pclk\_2\_frequency of rcc\_clocks with hclk\_frequency of rcc\_clocks right shifted by prescaler value.

### 11.10.12 RccAhb1PeriphClockCmd

Low Level Design Details about CSU RccAhb1PeriphClockCmd will follow in the sub sections.

#### 11.10.12.1 Brief Description

The function RccAhb1PeriphClockCmd enables or disables the AHB1 peripheral clock.

#### 11.10.12.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-012-CMU).

#### 11.10.12.3 List of global variables accessed and modified

Accessed: None

Modified: None

#### 11.10.12.4 Parameter list (Input/Output)

Inputs: T\_UINT32 rcc\_ahb1\_periph - Specifies the AHB1 peripheral to gate its clock.

T\_FUNCTIONAL\_STATE new\_state - New state of the specified peripheral clock.

Outputs: None

#### 11.10.12.5 Return Value

None

#### 11.10.12.6 Other CSUs called by this CSU

None

#### 11.10.12.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to RccAhb1PeriphClockCmd.

##### 11.10.12.7.1 daulibstm32f4xxrcc-RccAhb1PeriphClockCmd-LLR-001

Requirement ID: H698-LLD-CMU-FNC-1881

The function shall configure RCC AHB1 peripheral clock enable register to set the bit for received peripheral rcc\_ahb1\_periph i.e, ahb1enr of M\_RCC Bitwise OR with rcc\_ahb1\_periph when new\_state is not equal to DISABLE.

##### 11.10.12.7.2 daulibstm32f4xxrcc-RccAhb1PeriphClockCmd-LLR-002

Requirement ID: H698-LLD-CMU-FNC-1882

The function shall configure RCC AHB1 peripheral clock enable register to reset the bit for received peripheral rcc\_ahb1\_periph(i.e., ahb1enr of M\_RCC Bitwise AND with not of rcc\_ahb1\_periph) when new\_state is equal to DISABLE .

### 11.10.13 RccAhb3PeriphClockCmd

Low Level Design Details about CSU RccAhb3PeriphClockCmd will follow in the sub sections.

#### 11.10.13.1 Brief Description

The function RccAhb3PeriphClockCmd enables or disables the AHB3 peripheral clock.

#### 11.10.13.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-012-CMU).

#### 11.10.13.3 List of global variables accessed and modified

Accessed: None

Modified: None

#### 11.10.13.4 Parameter list (Input/Output)

Inputs: T\_UINT32 rcc\_ahb3\_periph - Specifies the AHB3 peripheral to gates its clock.

T\_FUNCTIONAL\_STATE new\_state - New state of the specified peripheral clock.

Outputs: None

#### 11.10.13.5 Return Value

None

#### 11.10.13.6 Other CSUs called by this CSU

None

#### 11.10.13.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to RccAhb3PeriphClockCmd.

##### 11.10.13.7.1 daulibstm32f4xxrcc-RccAhb3PeriphClockCmd-LLR-001

Requirement ID: H698-LLD-CMU-FNC-1891

The function shall configure RCC AHB3 peripheral clock enable register to set the bit for received peripheral rcc\_ahb3\_periph (i.e., set ahb3enr of M\_RCC to ahb3enr of M\_RCC with Bitwise OR with rcc\_ahb3\_periph) when new state is not equal to DISABLE.

##### 11.10.13.7.2 daulibstm32f4xxrcc-RccAhb3PeriphClockCmd-LLR-002

Requirement ID: H698-LLD-CMU-FNC-1892

The function shall configure RCC AHB3 peripheral clock enable register to reset the bit for received peripheral rcc\_ahb3\_periph (i.e set ahb3enr of M\_RCC to ahb3enr of M\_RCC with Bitwise AND with negation of rcc\_ahb3\_periph) when new state is equal to DISABLE.

### 11.10.14 RccApb1PeriphClockCmd

Low Level Design Details about CSU RccApb1PeriphClockCmd will follow in the sub sections.

#### 11.10.14.1 Brief Description

The function RccApb1PeriphClockCmd enables or disables the Low Speed APB (APB1) peripheral clock.

#### 11.10.14.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-012-CMU).

#### 11.10.14.3 List of global variables accessed and modified

Accessed: None

Modified: None

#### 11.10.14.4 Parameter list (Input/Output)

Inputs: T\_UINT32 rcc\_apb1\_periph - Specifies the APB1 peripheral to gate its clock.

T\_FUNCTIONAL\_STATE new\_state - New state of the specified peripheral clock.

Outputs: None

#### 11.10.14.5 Return Value

None

#### 11.10.14.6 Other CSUs called by this CSU

None

#### 11.10.14.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to RccApb1PeriphClockCmd.

##### 11.10.14.7.1 daulibstm32f4xxrcc-RccApb1PeriphClockCmd-LLR-001

Requirement ID: H698-LLD-CMU-FNC-1901

The function shall configure RCC APB1 peripheral clock enable register to set the bit for received peripheral rcc\_apb1\_periph (i.e., set ahb1enr of M\_RCC to ahb1enr of M\_RCC with Bitwise OR with rcc\_ahb1\_periph) when new state is not equal to DISABLE.

##### 11.10.14.7.2 daulibstm32f4xxrcc-RccApb1PeriphClockCmd-LLR-002

Requirement ID: H698-LLD-CMU-FNC-1902

The function shall configure RCC APB1 peripheral clock enable register to reset the bit for received peripheral rcc\_apb1\_periph (i.e., set ahb1enr of M\_RCC to ahb1enr of M\_RCC with Bitwise AND with negation of rcc\_ahb1\_periph) when new state is DISABLE.

### 11.10.15 RccGetFlagStatus

Low Level Design Details about CSU RccGetFlagStatus will follow in the sub sections.

#### 11.10.15.1 Brief Description

The function RccGetFlagStatus checks whether the specified RCC flag is set or not.

#### 11.10.15.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-012-CMU).

#### 11.10.15.3 List of global variables accessed and modified

Accessed: None

Modified: None

#### 11.10.15.4 Parameter list (Input/Output)

Inputs: T\_UINT8 rcc\_flag - Specifies the flag to check.

Outputs: None

#### 11.10.15.5 Return Value

T\_FLAG\_STATUS - The new state of rcc\_flag (SET or RESET)

#### 11.10.15.6 Other CSUs called by this CSU

None

#### 11.10.15.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to RccGetFlagStatus

##### 11.10.15.7.1 daulibstm32f4xxrcc-RccGetFlagStatus-LLR-001

Requirement ID: H698-LLD-CMU-FNC-1911

The function shall identify the RCC register (cr, BDCR or CSR) in which the rcc\_flag resides and check whether the specified rcc\_flag is set or not in the respective register. If the flag is set return status as SET else RESET.(i.e., sets the status register to cr, bdcr, csr of M\_RCC when the register index(rcc\_flag right shift of M\_SHIFT\_5) is equal to M\_ONE, M\_TWO, and other than M\_ONE, M\_TWO and sets the bit status to SET or RESET(i.e., returns bit\_status) when RESET is not of (status register AND M\_ONE left shift of register index(rcc\_flag AND M\_FLAG\_MASK))

### 11.10.16 RccApb1PeriphResetCmd

Low Level Design Details about CSU RccApb1PeriphResetCmd will follow in the sub sections.

#### 11.10.16.1 Brief Description

The function RccApb1PeriphResetCmd forces or releases Low Speed APB (APB1) peripheral reset.

#### 11.10.16.2 List of HLRs allocated

The list of HLRs allocated to this CSU is available in the Bi-Directional Traceability Matrix from SLL to SRS/SAD(H698-003-012-CMU).

#### 11.10.16.3 List of global variables accessed and modified

Accessed: None

Modified: None

#### 11.10.16.4 Parameter list (Input/Output)

Inputs: T\_UINT32 rcc\_apb1\_periph - Specifies the APB1 peripheral to reset.

T\_FUNCTIONAL\_STATE new\_state - new state of the specified peripheral clock.

Outputs: None

#### 11.10.16.5 Return Value

None

#### 11.10.16.6 Other CSUs called by this CSU

None

#### 11.10.16.7 Description of list of LLRs allocated

The following section will list the LLRs allocated to RccApb1PeriphResetCmd.

##### 11.10.16.7.1 daulibstm32f4xxrcc-RccApb1PeriphResetCmd-LLR-001

Requirement ID: H698-LLD-CMU-FNC-1920

The function shall configure RCC APB2 peripheral clock enable register to set the bit for received peripheral rcc\_apb2\_periph(i.e., set apb2enr of M\_RCC to apb2enr of M\_RCC Bitwise OR with rcc\_apb2\_periph) when new state is not equal to DISABLE.

##### 11.10.16.7.2 daulibstm32f4xxrcc-RccApb1PeriphResetCmd-LLR-002

Requirement ID: H698-LLD-CMU-FNC-1921

The function shall configure RCC APB2 peripheral clock enable register to reset the bit for received peripheral rcc\_apb2\_periph(i.e., set apb2enr of M\_RCC to apb2enr of M\_RCC Bitwise AND with negation of rcc\_apb2\_periph) when new state is equal to DISABLE.

# 12 Appendix A : Data Dictionary

The Data Dictionary for the CMU Flight Module :



# 13 Appendix B : Data Constants

The Data Constants for the CMU Flight Module :

