/IPB/20_Software/SCDD Library/LVDC DSP_C

SCDD_Scheduler

Software Component Detailed Design

Version: 0.2
Printed by: I-Ritesh.K
Printed on: Friday, August 2, 2024

Generated from DOORS 9.5.1.2

ID	Software Component Detailed Design
SCDD_Scheduler_1	1 **Software Component Design Description**
SCDD_Scheduler_2	1.1 **Introduction**
SCDD_Scheduler_3	This document describes the needed requirements for a SWC or BSWM.
SCDD_Scheduler_4	This is module is the Software Component Detail Description. It contains each SW component of each SW architecture. It is always structured in: External Interface Internal Design Requirements

ID	Software Component Detailed Design
SCDD_Scheduler_5	2 **Attributes**
SCDD_Scheduler_6	Agreed attributes for SWE.3 (ENG.6)

ID	Software Component Detailed Design
SCDD_Scheduler_10	3 **Views**
SCDD_Scheduler_11	**Review View**: This view is used for SCDD edit and review.

ID | Software Component Detailed Design

SCDD_Scheduler_13

4 Scheduler

SCDD_Scheduler_14

4.1 External Interfaces

SCDD_Scheduler_15

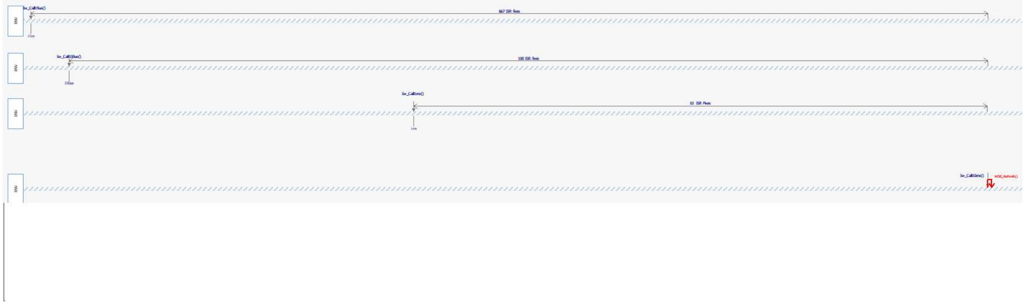
The function interface of this component are as following:

Function	Signal Name	Data Type	Direction
Isr_Call15us	N/A	N/A	N/A
Isr_Call100us	N/A	N/A	N/A
Isr_Call1ms	N/A	N/A	N/A
Isr_Call10ms	N/A	N/A	N/A

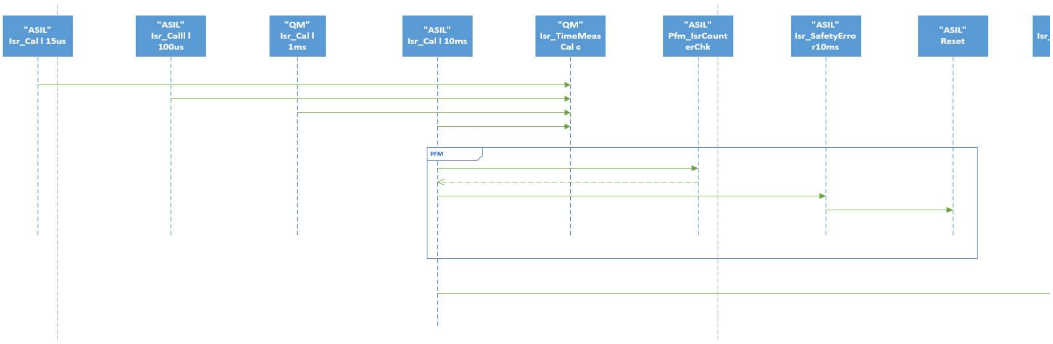
SCDD_Scheduler_16

4.2 Internal design

SCDD_Scheduler_17



SCDD_Scheduler_97



SCDD_Scheduler_18

4.3 Requirements

SCDD_Scheduler_19

4.3.1 ASIL

SCDD_Scheduler_23

The Scheduler component has ASIL-B level qualification.

ID	Software Component Detailed Design
SCDD_Scheduler_20	4.3.2 **Data Define**
SCDD_Scheduler_24	The C code of this component can be found in following link: https://desoeap16.delta.corp/svn/IPB_PPE_auto_porsche/branches/20_RelBranch/C01R0P0/LvDc/30_Bsw/Scheduler
SCDD_Scheduler_21	4.3.3 **Scheduler Functions**
SCDD_Scheduler_42	4.3.3.1 **ASIL Functions**
SCDD_Scheduler_26	4.3.3.1.1 **Isr_Call15us**
SCDD_Scheduler_28	This is the 15us timer ISR handler. It is called periodically, every 15us with a highest priority (Priority 0x60) and is a nested interrupt
SCDD_Scheduler_116	A counter shall count up from 0 upon every 15us ISR handler entry. This counter is used by PFM to determine the number of times 15us ISR handler was called in every span of 10ms.
SCDD_Scheduler_95	This function is calling : mpucfg_vSet_ActiveProtSet - Protection set 0 (ASIL) HsfbApp_Call15us HsfbApp_Call15us1 Gtm_PwmEnDis_Calc Isr_TimeMeasCalc
SCDD_Scheduler_27	4.3.3.1.2 **Isr_Call100us**
SCDD_Scheduler_29	This is the 100us timer ISR handler. It is called periodically, every 100us with Priority 0x50 and is a nested interrupt
SCDD_Scheduler_117	A counter shall count up from 0 upon every 100us ISR handler entry. This counter is used by PFM to determine the number of times 100us ISR handler was called in every span of 10ms.
SCDD_Scheduler_96	This function is calling : mpucfg_vSet_ActiveProtSet - Protection set 0 (ASIL) Isr_TimeMeasCalc HsfbApp_Call100us Gtm_CheckPwmStatus SBC_Triger100us Qspi_SBC_WWDTest Isr_TimeMeasCalc

ID	Software Component Detailed Design
SCDD_Scheduler_136	Call to SBC_Trigger100us function shall depend on PFM alive supervision and shall be the last ASIL function call in the ISR.
SCDD_Scheduler_103	If no PFM alive supervision violation is detected , the function shall keep refreshing SBC window WDG by toggling a DIO pin in the right window time frame.
SCDD_Scheduler_138	If PFM alive supervision violation is detected, the ISR shall skip triggering SBC window WDG until next PFM alive supervision result.
SCDD_Scheduler_139	The ISR shall ensure that a PFM alive supervision result is valid only for 10ms and shall set it to Invalid until next call to PFM alive supervision.
SCDD_Scheduler_99	the ISR shall disable interrupts, stop triggering SBC window WDG, wait in a "no operation" mode loop for SBC timeout and wait for SBC reset in maximum 50ms if 1. Maximum 10 PFM alive supervision violations are detected. 2. Maximum 4 QSPI errors are detected. 3. Maximum 1 MPU error is detected.
SCDD_Scheduler_104	If the function has stayed in the no operation mode loop for time more than SBC timeout (50ms), the function shall initiate internal MCU system reset by calling IOHWSF_vDefaultErrorHandler function with IOHWSF_E_TLF_ERR_DETECTION_TIMEOUT error code.
SCDD_Scheduler_44	4.3.3.1.3 **Isr_Call1ms**
SCDD_Scheduler_45	This is the 1ms timer ISR handler. It is called periodically, every 1ms, with Priority 0x40 and is a nested interrupt
SCDD_Scheduler_118	A counter shall count up from 0 upon every 1ms ISR handler entry. This counter is used by PFM to determine the number of times 1ms ISR handler was called in every span of 10ms.
SCDD_Scheduler_46	This function is calling : mpucfg_vSet_ActiveProtSet - Protection set 0 (ASIL) Isr_TimeMeasCalc mg_vASIL_1msTask mpucfg_vSet_ActiveProtSet - Protection set 1 (QM) mg_vQmTaskCall mpucfg_vSet_ActiveProtSet - Protection set 0 (ASIL) Uds_ASILTask
SCDD_Scheduler_113	4.3.3.1.4 **Isr_Call10ms**
SCDD_Scheduler_114	This is the 10ms timer ISR handler , it's called periodically each 10ms with Priority 0x30 and it's a nested interrupt.

ID	Software Component Detailed Design
SCDD_Scheduler_115	This function is calling : mpucfg_vSet_ActiveProtSet - Protection set 0 (ASIL) Isr_TimeMeasCalc ComServ_10msTask Isr_AppStatus10ms Isr_ErrorDis10ms Isr_SafetyError10ms HsfbApp_Call10ms Qspi_SBC_Check10ms Qspi_Cmd_Test10ms Adc_ShortDetect_10ms Port_OTP_AutoDiag10ms mg_vPfm_IsrCounterChk
SCDD_Scheduler_30	4.3.3.1.5 **static void mg_vPfm_IsrCounterChk(void)**
SCDD_Scheduler_34	This function is used for PFM alive supervision detection
SCDD_Scheduler_100	It performs alive supervision by checking the Alive counteres of ISR_15us , ISR_100us and ISR_1ms with the following parameters : -Isr_15usCnt be in range of 400-935 (Nominal = 667 ISRs fire +/- 40% Tol) -Isr_100usCntbe in range of 60 -140 (Nominal =100 ISRs fire +/- 40% Tol) -Isr_1msCnt be in range of 6 - 14 (Nominal =10 ISRs fire +/- 40% Tol)
SCDD_Scheduler_101	It performs alive supervision by capturing STM timer_1 frequency on ISR_10ms and check that the ISR is firing every 10ms using the following parameters : -TimerDiff should be in range of 37500 - 87500 (Nominal =62500 STM timer ticks +/- 40% Tol)
SCDD_Scheduler_102	This function has to protect ISR_15us , ISR_100us and ISR_1ms against overflow and it also has to keep updating the alive counters so as not to lose any higher priority ISR firing. Hint: using Atomic reading of the alive supervision is not possible due to High frequency and criticality of ISR_15us.
SCDD_Scheduler_38	The Function has to mind STM timer overflow probability and to calculate ISR_10ms firing time difference using the following Formula: Time_Diff = (0xFFFFFFFF - Timer_history) + Timer_value;
SCDD_Scheduler_106	PFM violation should be detected in case one of the Previous parameteres are not in Range for PFM_CHK_FAILED_COUNT_MAX (10)
SCDD_Scheduler_137	This function calls Pma_vGetPfmSts to report errors to Pms module.
SCDD_Scheduler_121	4.3.3.1.6 **void __attribute__((noinline)) mg_vASIL_1msTask(void)**

ID	Software Component Detailed Design
SCDD_Scheduler_122	This function is a wrapper for 1ms ASIL tasks
SCDD_Scheduler_123	This function calls: CANDriver_Diag ComServ_1msTask HsfbApp_Call1ms ORU_LatchTest1ms Qspi_StartUp
SCDD_Scheduler_124	4.3.3.1.7 **void Uds_ASILTask(void)**
SCDD_Scheduler_125	This is the ASIL task of the uds module. This routine handles the UDS ECU Reset request.
SCDD_Scheduler_126	The call to this function is based on QM trusted variable Trusted_UdsRstReq_Flag set by QM task Uds_Task()
SCDD_Scheduler_129	This function will verify the UDS ECU reset request and call the Uds_EcuReset function in case of correct mapping or set NRC in case of any errors detected.
SCDD_Scheduler_127	4.3.3.1.8 **static inline uint8 Uds_EcuReset(uint8 *RxTxBuf, uint16 LenOfMsg)**
SCDD_Scheduler_130	This message calls Uds_McuSwReset funcion if the subfunction of ECU reset request is successfully verified or set the required NRC.
SCDD_Scheduler_128	4.3.3.1.9 **void Uds_McuSwReset(void)**
SCDD_Scheduler_131	This function disables required GTM modules and then performs the requested ECU reset.
SCDD_Scheduler_132	This function calls: IfxScuWdt_clearSafetyEndinit SET_TOM_OUTEN (disable required GTM modules: PRIA, PRIB, PRIC, PRID, SRA, SRB, CLAMPA, CLAMPB) IOHWSF_vDefaultErrorHandler(IOHWSF_E_UDS_ECU_RESET_REQ) IfxScuWdt_setSafetyEndinit
SCDD_Scheduler_108	4.3.3.1.10 **static void Isr_SafetyError10ms(void)**
SCDD_Scheduler_109	This function is used to monitor errors related to SBC startup tests and runtime register read via QSPI and CAN E2E Rx messages errors.

ID	Software Component Detailed Design
SCDD_Scheduler_112	This function is periodically called every 10ms.
SCDD_Scheduler_43	4.3.3.2 **QM Functions**
SCDD_Scheduler_119	4.3.3.2.1 **void mg_vQmTaskCall(void)**
SCDD_Scheduler_120	This function is used to set the safety task identifier for QM task.
SCDD_Scheduler_135	The safety task identifier is defined in CPU_PSW.S register bit. For QM task: CPU_PSW.S = 0
SCDD_Scheduler_134	This function calls: IOHWSF_SCCESSPROT_vSwitchToQmISR mg_vQM_Task
SCDD_Scheduler_47	4.3.3.2.2 **void mg_vQM_Task(void)**
SCDD_Scheduler_48	This function is wrapper for all QM runnables
SCDD_Scheduler_111	This function will verify that the safety task identifier bit is cleared and access is User0 mode. in case of failure to clear the safety task identifier bit or user-0 access mode, SAFETY_TASK_ID_FAIL error would be set and 100us task would perform a SBC safety reaction. If safety task identifier bit is cleared and user0 mode is set correctly, then following functions are called: CANDriver_Diag TpTask Uds_Task Pma_vTask10ms