INDUCTION TRAINING EXERCISE - HLD DESIGN

Template ID	VIAS_T01_00_006
Reference	
Version	1.3
Status	Draft
Author	Mahmoud SHALABY
Date	30/5/2012

Name	Department	Title	Date
Approved by:			
Amr ABDELNABY	Training	Training Leader	30/5/2012
Reviewed by:			
Amr ABDELNABY	Training	Training Leader	30/5/2012

Document History

Revision	Status	Date	Author	Changes
1.0	Draft	19/04/2012	M.Shalaby	- Initial version
1.1	Draft	15/05/2012	M.Shalaby	- Removed Instruction Page - Updated assignation attribute for requirements in 3.3 - Updated section 4 - Design Constraints - Filled in section 5.1 and 5.2
1.2	Draft	16/05/2012	M.Shalaby	- Filled in section 5.3 *(note: section 5.4 remaining)* - Filled in section 7 *(note: 7.1.5 remaining)* - Defined all subsystem (section 8) - Filled in section 8.2 (HAL subsystem)
1.3	Draft	17/05/2012	M.Shalaby	- Modified the location of the HIF/TIF components - Filled in section remaining of section 8 - Filled in sections 5.4 and 7.1.5 - *Sections 5 and 7.1 are to be revisited* - *Section 7.2.3 Interface Design needs to be revisited*

Table of Contents

1 Introduction

This section provides an overview of the entire design document. It should introduce the reader to the system/software and set the context for the rest of the document.

1.1 Document Scope

This document is the High Level Design Document for project Induction Training Exercise identified by VALEO project identifier INDUC.

The software high-level/global design activity defines for a software project:

- The real time constraints on the project (issued from upstream requirements or defined during the global design),
- The software hypothesis and choices taken to respect these constraints,
- The shared data identification and the associated protection mechanisms,
- The software static and dynamic architecture.

This document contains separate parts:

- The software real-time constraints collection: Identification and classification of all real-time constraints issued from upstream requirements (project, product, hardware, COTS, standard components).
- The real-time analysis:
    - Association between the previous real-time constraints and the SA-RT functions,
    - Design hypothesis and choices (according to the constraints association) in order to identify the necessary tasks and Interrupts,
    - The dynamic architecture with detailed tasks, interrupts and treatments description,
    - The shared data identification and the associated protection mechanisms.
- The static design:
    - The software static architecture,
    - The treatments allocation in components,
    - The dataflow diagrams,
    - The components description.

This part is completed by the detailed design of components/modules.

1.2 Goals and Objectives

The software described in this document is a simulation to the TCM ECU present in most of the modern vehicles and is part of the induction training exercise given at VALEO Egypt. It simulates the basic functionalities of the TCM (right blinker, left blinker and hazard blinker –which is also called double flasher in common language) through blinking 2 SSDs attached to the system.

1.3 Design Methodology and Standards

#	Document Name	Reference	Version
1	Coding Rules	GEI-RD-H01-0000-095	1.10
2	Source Code Naming Rules Guidelines	GEI-RD-H01-0000-094	1.5
3	SW Design Guide	GEI-RD-H01-0000-081	G
4	Source Code Appearance Rules	GEI-RD-H01-0000-096	1.0

1.4 Reference Documents

#	Document Name	Reference	Version
1	Software Development Plan*	N/A*	N/A*
2	Software Requirements Specification	SRS Induction.doc	1.0
3	Hardware Software Interface	Controller Data Sheet.pdf	Rev 3 1/2009

* Software Development Plan (SDP) is not applicable for induction training as this HLD is only for demonstration purposes.

2 Terms and Abbreviations

Abbreviation	Description
VIAS	Valeo Inter-branch Automotive Software (VALEO Egypt)
CEE	Center for Electronics Excellence
HW	Hardware
SW	Software
HIS	Hardware/Software Interface
ECU	Electronic Control Unit
TCM	Top Column Module
SSD	Seven Segment Display
PB	Push Button

3 System Description

3.1 Overview

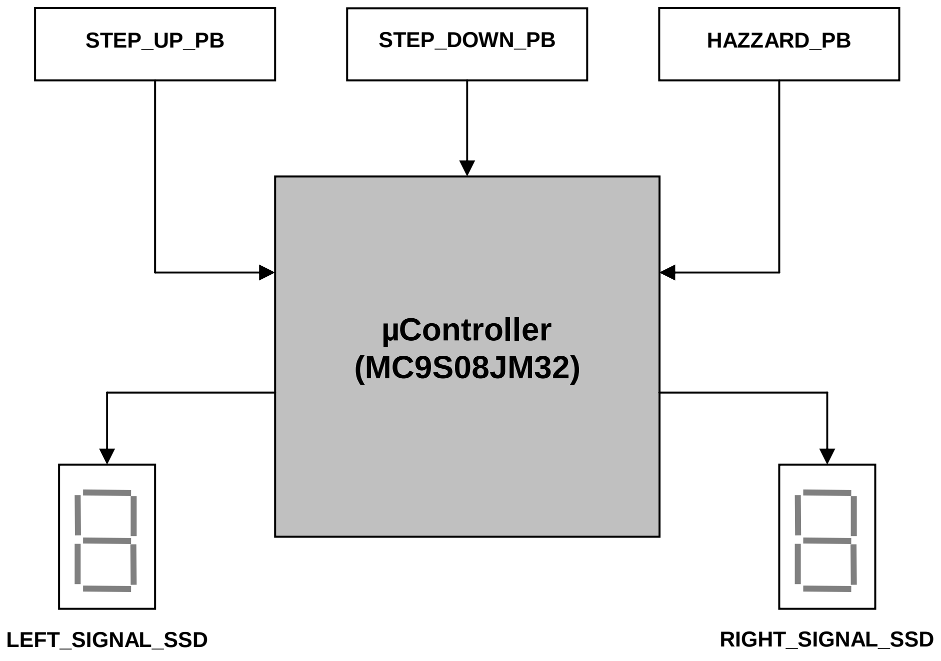

Figure 1 - System Overview

The purpose of this system is to simulate the following features of the TCM ECU:

- Left Blinker
- Right Blinker
- Hazards Blinkers (double flasher)

The simulation of the TCM arms is done using three push buttons, two of them are simulating that the arm is moved one step up or down to identify the blinking state and the third simulates pressing on the hazards button present in the vehicle.

For more details about the system's overview, refer to REF2; section 2.1.

3.2 Software Context Diagram

Refer to REF2; Figure 2 – Software External Interfaces for more details.

3.3 Software External Interfaces

Signal Name		Source/Destination	Physical Pin
Description		**Characteristics**	
{HLD-0001(0)}		{Covers: SRS-0001(0)}	
{Assignation: PORT}	{Test Method: Component Test}	{Integration Phase: }	{Justification: }
PB_PRESS2		STEP_DOWN_PB Push Button	PTG1 [Pin N°22]
The signal issued by STEP_DOWN_PB (Press2) Push Button		Type: Digital Input Properties: (1 → ON, 0 → OFF)	
{HLD-0002(0)}		{Covers: SRS-0002(0)}	
{Assignation: PORT}	{Test Method: Component Test}	{Integration Phase: }	{Justification: }
PB_PRESS3		STEP_UP_PB Push Button	PTG2 [Pin N°34]
The signal issued by STEP_UP_PB (Press3) Push Button		Type: Digital Input Properties: (1 → ON, 0 → OFF)	
{HLD-0003(0)}		{Covers: SRS-0003(0)}	
{Assignation: PORT}	{Test Method: Component Test}	{Integration Phase: }	{Justification: }
PB_PRESS4		HAZARD_UP_PB Push Button	PTG3 [Pin N°35]
The signal issued by HAZZARD_PB (Press4) Push Button		Type: Digital Input Properties: (1 → ON, 0 → OFF)	
{HLD-0004(0)}		{Covers: SRS-0004(0)}	
{Assignation: PORT}	{Test Method: Component Test}	{Integration Phase: }	{Justification: }
SS_A		AFF1/AFF2	PTB0 [Pin N°23]
The signal issued by the micro-controller to control segment “a” of the 2 SSDs, RIGHT_SIGNAL_SSD (AFF1) and LEFT_SIGNAL_SSD (AFF2).		Type: Digital Output Properties: (1 → ON, 0 → OFF)	
{HLD-0005(0)}		{Covers: SRS-0005(0)}	
{Assignation: PORT}	{Test Method: Component Test}	{Integration Phase: }	{Justification: }
SS_B		AFF1/AFF2	PTB1 [Pin N°24]
The signal issued by the micro-controller to control segment “b” of the 2 SSDs, RIGHT_SIGNAL_SSD (AFF1) and LEFT_SIGNAL_SSD (AFF2).		Type: Digital Output Properties: (1 → ON, 0 → OFF)	
{HLD-0006(0)}		{Covers: SRS-0006(0)}	
{Assignation: PORT}	{Test Method: Component Test}	{Integration Phase: }	{Justification: }
SS_C		AFF1/AFF2	PTB2 [Pin N°25]
The signal issued by the micro-controller to control segment “c” of the 2 SSDs, RIGHT_SIGNAL_SSD (AFF1) and LEFT_SIGNAL_SSD (AFF2).		Type: Digital Output Properties: (1 → ON, 0 → OFF)	
{HLD-0007(0)}		{Covers: SRS-0007(0)}	
{Assignation: PORT}	{Test Method: Component Test}	{Integration Phase: }	{Justification: }
SS_D		AFF1/AFF2	PTB3 [Pin N°26]
The signal issued by the micro-controller to control segment “d” of the 2 SSDs, RIGHT_SIGNAL_SSD (AFF1) and LEFT_SIGNAL_SSD (AFF2).		Type: Digital Output Properties: (1 → ON, 0 → OFF)	
{HLD-0008(0)}		{Covers: SRS-0008(0)}	
{Assignation: PORT}	{Test Method: Component Test}	{Integration Phase: }	{Justification: }
SS_E		AFF1/AFF2	PTE4 [Pin N°12]
The signal issued by the micro-controller to control segment “e” of the 2 SSDs, RIGHT_SIGNAL_SSD (AFF1) and LEFT_SIGNAL_SSD (AFF2).		Type: Digital Output Properties: (1 → ON, 0 → OFF)	
{HLD-0009(0)}		{Covers: SRS-0009(0)}	
{Assignation: PORT}	{Test Method: Component Test}	{Integration Phase: }	{Justification: }
SS_F		AFF1/AFF2	PTE5 [Pin N°13]
The signal issued by the micro-controller to control segment “f” of the 2 SSDs, RIGHT_SIGNAL_SSD (AFF1) and LEFT_SIGNAL_SSD (AFF2).		Type: Digital Output Properties: (1 → ON, 0 → OFF)	

{HLD-0010(0)}		{Covers: SRS-0010(0)}	
{Assignation: PORT}	{Test Method: Component Test}	{Integration Phase: }	{Justification: }
SS_G		**AFF1/AFF2**	**PTE6 [Pin Nº14]**
The signal issued by the micro-controller to control segment "g" of the 2 SSDs, RIGHT_SIGNAL_SSD (AFF1) and LEFT_SIGNAL_SSD (AFF2).		**Type: Digital Output** **Properties: (1 → ON, 0 → OFF)**	
{HLD-0011(0)}		{Covers: SRS-0011(0)}	
{Assignation: PORT}	{Test Method: Component Test}	{Integration Phase: }	{Justification: }
SS_DOT		**AFF1/AFF2**	**PTE7 [Pin Nº15]**
The signal issued by the micro-controller to control segment "dp" of the 2 SSDs, RIGHT_SIGNAL_SSD (AFF1) and LEFT_SIGNAL_SSD (AFF2).		**Type: Digital Output** **Properties: (1 → ON, 0 → OFF)**	
{HLD-0012(0)}		{Covers: SRS-0012(0)}	
{Assignation: PORT}	{Test Method: Component Test}	{Integration Phase: }	{Justification: }
SS_SEG1		**AFF1**	**PTB4 [Pin Nº27]**
The signal issued by the micro-controller to enable/disable the Seven Segment Display RIGHT_SIGNAL_SSD (AFF1).		**Type: Digital Output** **Properties: (1 → ON, 0 → OFF)**	
{HLD-0013(0)}		{Covers: SRS-0013(0)}	
{Assignation: PORT}	{Test Method: Component Test}	{Integration Phase: }	{Justification: }
SS_SEG2		**AFF2**	**PTB5 [Pin Nº28]**
The signal issued by the micro-controller to enable/disable the Seven Segment Display LEFT_SIGNAL_SSD (AFF2).		**Type: Digital Output** **Properties: (1 → ON, 0 → OFF)**	

Table 1 - Software External Interfaces

4 Design Constraints

4.1 Constraints on Initialization

None

4.2 Constraints on Inputs

Constraint ID	Description	Constraint Type	Value
[INP_1]	For the HAZARD_PB, the input signal is considered valid once it is released after being pressed for at least 200ms (signal stabilization time).	Interval	200ms
[INP_2]	For STEP_DOWN_PB and STEP_UP_PB, the input signal is considered valid once it is released after being pressed for at least 10ms (signal stabilization time).	Interval	10ms

Table 2 - Input Constraints

4.3 Constraints on Outputs

Constraint ID	Description	Constraint Type	Value
[OUT_1]	The SSDs controlled must be switched ON for 200ms ± 5 ms if the system is in either the LEFT or RIGHT blink states.	Period Jitter	200ms ±5ms
[OUT_2]	The SSDs controlled must be switched OFF for 300ms ± 5 ms if the system is in either the LEFT or RIGHT blink states.	Period Jitter	300ms ±5ms
[OUT_3]	The SSDs controlled must be switched ON for 200ms ± 5 ms if the system is in the HAZARD blink state.	Period Jitter	200ms ±5ms
[OUT_4]	The SSDs controlled must be switched OFF for 400ms ± 5 ms if the system is in the HAZARD blink state.	Period Jitter	400ms ±5ms

Table 3 - Output Constraints

4.4 Hardware Constraints & Dependencies

Constraint ID	Description	Type
[HW_1]	The software shall initialize and configure the PLL to obtain 24 MHz bus frequency taking into consideration that the oscillator frequency is 4 MHz.	PLL
[HW_2]	The microcontroller ports should be configured as digital input/output based on the information present in section 3.3	Digital Ports

Table 4 - Hardware Constraints

4.5 Communication/Network Constraints

None

4.6 Diagnostic Constraints

None

4.7 Dimensioning Constraints

Constraint ID	Description		Type
{HLD-0014(0)}		{Covers: SRS-0043(0)}	
{Assignation: }	{Test Method: Static Test}	{Integration Phase: }	{Justification: }
[PERF_1]	The CPU load shall not exceed 80%.		CPU Load
{HLD-0015(0)}		{Covers: SRS-0044(0)}	
{Assignation: }	{Test Method: Static Test}	{Integration Phase: }	{Justification: }
[PERF_2]	The code size in FLASH shall not exceed 1 KB.		Code Size
{HLD-0016(0)}		{Covers: SRS-0045(0)}	
{Assignation: }	{Test Method: Static Test}	{Integration Phase: }	{Justification: }
[PERF_3]	The RAM consumption shall not exceed 1 KB.		Code Size
{HLD-0017(0)}		{Covers: SRS-0046(0)}	
{Assignation: }	{Test Method: Static Test}	{Integration Phase: }	{Justification: }
[PERF_4]	The stack consumption shall not exceed 1 KB.		Code Size
{HLD-0018(0)}		{Covers: SRS-0041(0)}	
{Assignation: }	{Test Method: Static Test}	{Integration Phase: }	{Justification: }
[PERF_5]	The source code shall conform to MISRA rules.		Code Quality
{HLD-0019(0)}		{Covers: SRS-0042(0)}	
{Assignation: }	{Test Method: Static Test}	{Integration Phase: }	{Justification: }
[PERF_6]	The source code shall follow the Valeo naming, coding and presentation rules.		Code Quality
{HLD-0020(0)}		{Covers: SRS-0039(0)}	
{Assignation: TIF}	{Test Method: Component Test}	{Integration Phase: }	{Justification: }
[PERF_7]	The response time for a transition between a state to another shall not exceed 20 ms.		Performance

Table 5 - Dimensioning Constraints

5 Real Time Analysis

5.1 Processes Identification

Constraint ID	Basic Operation	Process	Period / Frequency	Response Time	Jitter
[INP_1]	Managing the correct capturing of the HAZZARD_PB press	[HIF]	200ms	N/A	
[INP_2]	Managing the correct capturing of both the STEP_DOWN_PB and STEP_UP_PB presses	[TIF]	100ms	N/A	
[OUT_1]	Control the turning on of the SSD of RIGHT and LEFT SSDs	[TIF]	200ms	N/A	±5ms
[OUT_2]	Control the turning off of the SSD of RIGHT and LEFT SSDs	[TIF]	300ms	N/A	±5ms
[OUT_3]	Control the turning on of the SSD of the HAZARD signal	[HIF]	200ms	N/A	±5ms
[OUT_4]	Control the turning off of the SSD of the HAZARD signal	[HIF]	400ms	N/A	±5ms
[PREF_7]	Control the response time to guarantee smooth transitions between different system states	[TIF]		20ms	

Table 6 - Processes Identification

5.2 Timing Bases Identification

Process	Description	Constraints	Period / Frequency	Response Time	Jitter
[HIF]	HAZARD signal capturing	[INP_1]	200ms		
	Turn on of both Left/Right SSDs	[OUT_3]	200ms		±5ms
	Turn off of both Left/Right SSDs	[OUT_4]	400ms		±5ms
[TIF]	STEP_DOWN/UP capturing	[INP_2]	100ms		
	Turn on of Left/Right SSDs	[OUT_1]	200ms		±5ms
	Turn off of Left/Right SSDs	[OUT_2]	300ms		±5ms
	Guarantee smooth transitions	[PREF_7]		20ms	

Table 7 - Time Bases Identification

5.3 Software Operating Modes

The following table lists the software operating modes.

Operating Mode	Description	Constraints
[IDLE]	Both left and right SSDs must be switched off	
[LEFT_BLINK]	Only the left SSD is to be blinking as specified above	[OUT1, OUT2]
[RIGHT_BLINK]	Only the right SSD is to be blinking as specified above	[OUT1, OUT2]
[HAZARD_BLINK]	Both left and right SSDs must be blinking as specified above	[OUT3, OUT4]

Table 8 - Software Operating Modes

The following diagram shows the software operating modes including transitions.

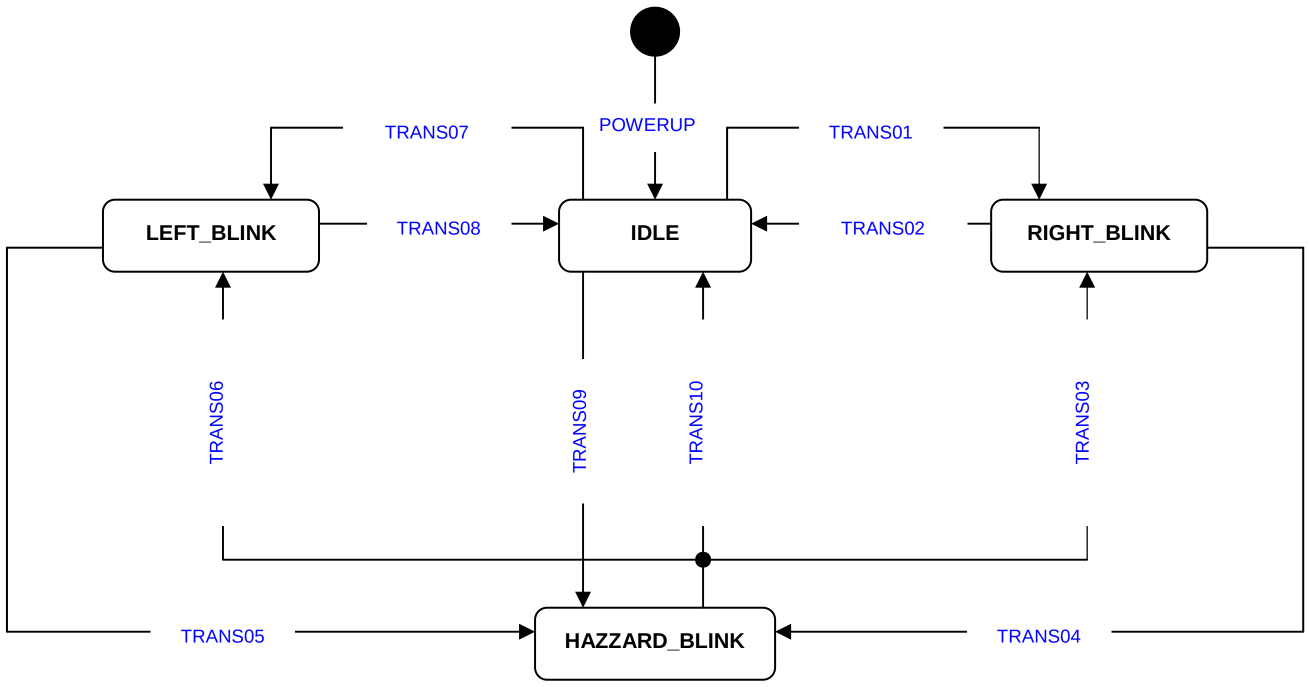

Figure 2 - Software Operating Modes and Transitions

The following table describes the transitions between the operating modes.

Transition ID	Event Description
[TRANS01]	The system goes to RIGHT_BLINK state upon a correct press of STEP_UP_PB
[TRANS02]	The system goes to IDLE state upon a correct press of STEP_DOWN_PB
[TRANS03]	The system goes to RIGHT_BLINK state upon a correct press of HAZZARD_PB given that the previous state was RIGHT_BLINK
[TRANS04]	The system goes to the HAZARD_BLINK state upon a correct press of HAZARD_PB
[TRANS05]	The system goes to the HAZARD_BLINK state upon a correct press of HAZARD_PB
[TRANS06]	The system goes to LEFT_BLINK state upon a correct press of HAZZARD_PB given that the previous state was LEFT_BLINK

Transition ID	Event Description
[TRANS07]	The system goes to LEFT_BLINK state upon a correct press of STEP_DOWN_PB
[TRANS08]	The system goes to IDLE state upon a correct press of STEP_UP_PB
[TRANS09]	The system goes to the HAZARD_BLINK state upon a correct press of HAZARD_PB
[TRANS10]	The system goes to IDLE state upon a correct press of HAZZARD_PB given that the previous state was IDLE
[POWERUP]	Upon power up, the system directly goes to the IDLE state

Table 9 - Transitions Description

The following table lists the processes to be performed in each operating mode.

Operating Mode	Processes
[IDLE]	MODE
[LEFT_BLINK]	TIF
[RIGHT_BLINK]	TIF
[HAZARD_BLINK]	HIF

5.4 Shared Resources

5.4.1 Hardware Resources

None

5.4.2 Software Resources

None

6 Design Alternatives and Justification

None

7 Architectural Design

7.1 Real Time Architecture

7.1.1 Interrupts

IT	Period	Process	Maximum Delay	Response Time	Estimated Duration	Estimated CPU Load	IT Level
[RESET]*	-	-			2us	0.00%	1
[UNUSED]*	-	-			2us	0.00%	1
[RTC]	1ms	[SCHED_P]			8us	0.80%	1
					Estimated CPU Load	**0.80%**	

* Those interrupts are not used by the system and are not expected to occur; accordingly no estimated CPU load is taken into account

7.1.2 Cyclic Tasks

Task	Period	Process	Maximum Delay	Response Time	Estimated Duration	Estimated CPU Load	Priorit y
[TASK1]	1ms	[SCHED_P]			20us	2.00%	1
[TASK2]	5ms	[SSD_P]			182us	3.64%	1
[TASK3]	10ms	[PBD_P]			10us	0.10%	1
[TASK4]	10ms	[MODE_P]			160us	1.60%	1
[TASK5]	10ms	[HIF_P]			64us	0.64%	1
[TASK6]	10ms	[TIF_P]			82us	0.82%	1
					Estimated CPU Load	**8.80%**	

7.1.3 Event-triggered Tasks

None

7.1.4 Estimated Workload

Total Workload Estimate = **9.60%.**

7.1.5 Shared Resources Protection

7.1.5.1 Hardware Resources

None

7.1.5.2 Software Resources

None

7.1.6 Dynamic Architecture Requirements

7.1.6.1 Operating System Configuration

The following table describes the required operating system configuration for the dynamic decisions taken of the interrupts and tasks in the system and allocates them to a certain component.

For more information about the interrupts and tasks design; refer to sections 7.1.1, 7.1.2 and 7.1.3.

Description		**Interrupt / Task**	**Type**
			Value
{HLD-0021(0)}	{Covers: }		
{Assignation: SCHED}	{Test Method: Integration Test}	{Integration Phase: }	{Justification: Real Time Design }
RTC interrupt periodicity		[SCHED_P]	Tick
			1ms
{HLD-0022(0)}	{Covers: }		
{Assignation: SCHED}	{Test Method: Integration Test}	{Integration Phase: }	{Justification: Real Time Design }
RTC interrupt priority		[SCHED_P]	Priority
			1
{HLD-0023(0)}	{Covers: }		
{Assignation: SCHED}	{Test Method: Integration Test}	{Integration Phase: }	{Justification: Real Time Design }
Management of the different system tasks		[SCHED_P]	Period
			1ms
{HLD-0024(0)}	{Covers: }		
{Assignation: SCHED}	{Test Method: Integration Test}	{Integration Phase: }	{Justification: Real Time Design }
Management of the different system tasks priority		[SCHED_P]	Priority
			1
{HLD-0025(0)}	{Covers: }		
{Assignation: SSD}	{Test Method: Integration Test}	{Integration Phase: }	{Justification: Real Time Design }
Management of the 3 SSDs		[SSD_P]	Period
			5ms
{HLD-0026(0)}	{Covers: }		
{Assignation: SCHED}	{Test Method: Integration Test}	{Integration Phase: }	{Justification: Real Time Design }
Management of the 3 SSDs priority		[SSD_P]	Priority
			1
{HLD-0027(0)}	{Covers: }		
{Assignation: PBD}	{Test Method: Integration Test}	{Integration Phase: }	{Justification: Real Time Design }
Management of the push buttons		[PBD_P]	Period
			10ms
{HLD-0028(0)}	{Covers: }		
{Assignation: SCHED}	{Test Method: Integration Test}	{Integration Phase: }	{Justification: Real Time Design }
Management of the push buttons priority		[PBD_P]	Priority
			1
{HLD-0029(0)}	{Covers: }		
{Assignation: MODE}	{Test Method: Integration Test}	{Integration Phase: }	{Justification: Real Time Design }
Management of different system operating modes		[MODE_P]	Period
			10ms

Description		Interrupt / Task	Type
			Value
{HLD-0030(0)}		{Covers: }	
{Assignation: SCHED}	{Test Method: Integration Test}	{Integration Phase: }	{Justification: Real Time Design }
Management of different system operating modes priority		[MODE_P]	Priority
			1
{HLD-0031(0)}		{Covers: }	
{Assignation: HIF}	{Test Method: Integration Test}	{Integration Phase: }	{Justification: Real Time Design }
Management of the hazards interface		[HIF_P]	Period
			10ms
{HLD-0032(0)}		{Covers: }	
{Assignation: SCHED}	{Test Method: Integration Test}	{Integration Phase: }	{Justification: Real Time Design }
Management of the hazards interface priority		[HIF_P]	Priority
			1
{HLD-0033(0)}		{Covers: }	
{Assignation: TIF}	{Test Method: Integration Test}	{Integration Phase: }	{Justification: Real Time Design }
Management of the turning interface		[TIF_P]	Period
			10ms
{HLD-0034(0)}		{Covers: }	
{Assignation: SCHED}	{Test Method: Integration Test}	{Integration Phase: }	{Justification: Real Time Design }
Management of the turning interface priority		[TIF_P]	Priority
			1

7.1.6.2 Estimated Execution Times

The following table lists all estimated execution times for the interrupts/tasks in addition to distributing the operations done of each task on different components.

For more information about the interrupts and tasks design; refer to sections 7.1.1, 7.1.2 and 7.1.3.

Description		Interrupt / Task	Duration
{HLD-0035(0)}		{Covers: }	
{Assignation: }	{Test Method: Integration Test}	{Integration Phase: }	{Justification: Real Time Design }
Execution time for the RTC interrupt		[RTC]	8us
{HLD-0036(0)}		{Covers: }	
{Assignation: }	{Test Method: Integration Test}	{Integration Phase: }	{Justification: Real Time Design }
Execution time for TASK1		[TASK1]	20us
{HLD-0037(0)}		{Covers: }	
{Assignation: }	{Test Method: Integration Test}	{Integration Phase: }	{Justification: Real Time Design }
Execution time for TASK2		[TASK2]	182us
{HLD-0038(0)}		{Covers: }	
{Assignation: }	{Test Method: Integration Test}	{Integration Phase: }	{Justification: Real Time Design }
Execution time for TASK3		[TASK3]	10us
{HLD-0039(0)}		{Covers: }	
{Assignation: }	{Test Method: Integration Test}	{Integration Phase: }	{Justification: Real Time Design }

Description	Interrupt / Task	Duration
Execution time for TASK4	[TASK4]	160us
{HLD-0040(0)}	{Covers: }	
{Assignation: } {Test Method: Integration Test}	{Integration Phase: }	{Justification: Real Time Design }
Execution time for TASK5	[TASK5]	64us
{HLD-0041(0)}	{Covers: }	
{Assignation: } {Test Method: Integration Test}	{Integration Phase: }	{Justification: Real Time Design }
Execution time for TASK6	[TASK6]	82us

7.2 Static Architecture (Internal Interfaces)

7.2.1 Layered Architecture

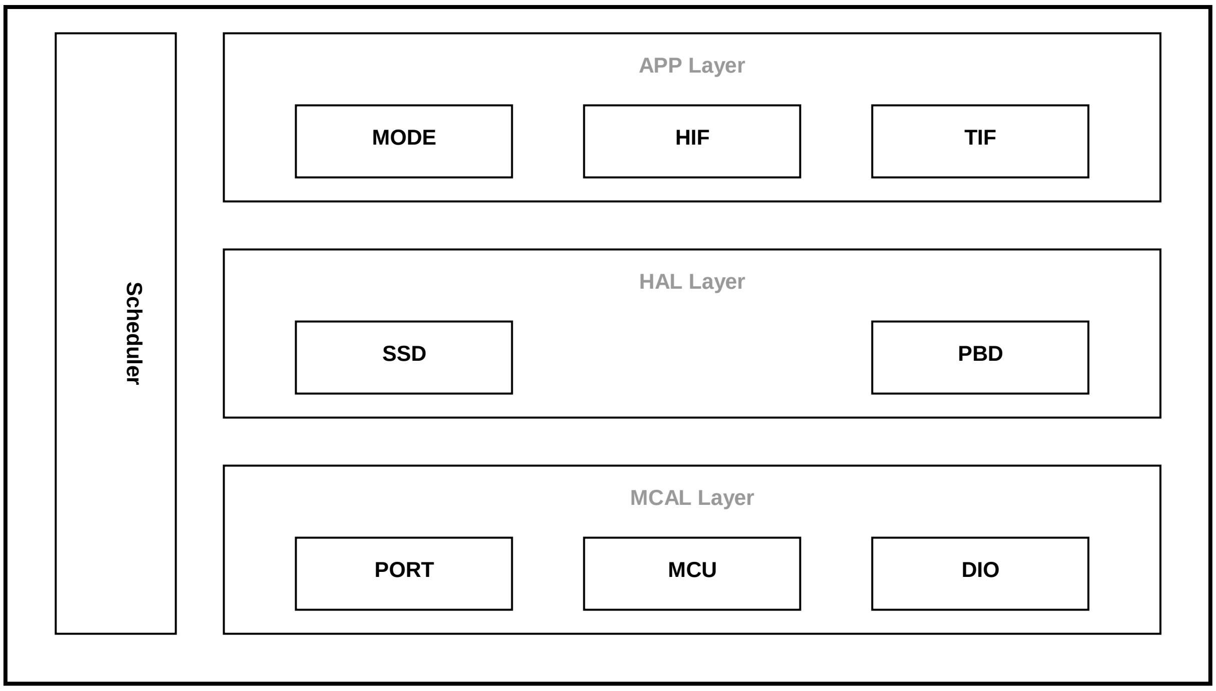

Figure 3 - Layered System Architecture

7.2.2 Components Design

Port Driver (PORT): Configures the different microcontroller ports.
Mcu Driver (MCU): Configures the microcontroller clock.
Digital Inputs/Outputs Driver (DIO): Commands the digital outputs and reads the digital inputs.
Seven Segment Driver (SSD): Hardware abstraction driver to command the seven segments.
Push Button Driver (PBD): Hardware abstraction driver to read the push buttons.
Mode Manager (MODE): Manages the different modes of the system.
Hazards Interface (HIF): Manages everything related to the hazard TCM operation.
Turning Interface (TIF): Manages everything related to the right and left TCM operations.
Scheduler: Responsible for scheduling different system tasks and manages the RTC interrupt.

7.2.3 Interfaces Design

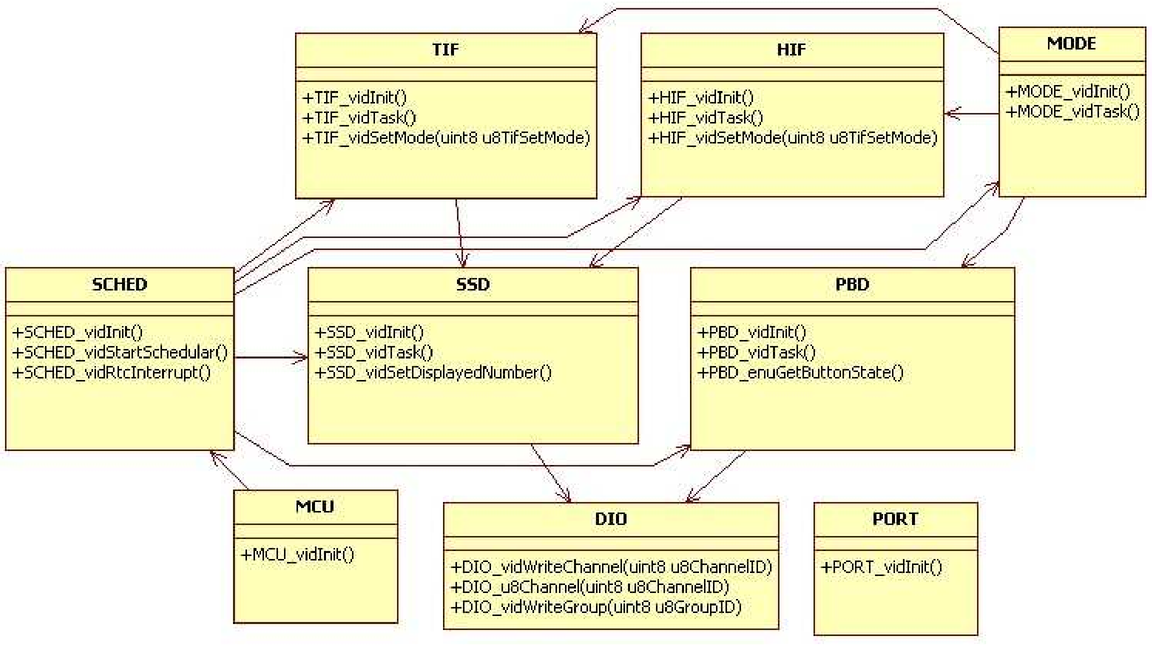

Figure 4 - Interfaces Design

7.2.4 Subsystems

For the sake of the exercise, the MCAL layer will be in one subsystem, HAL layer will be in one subsystem and the APP layer will be one subsystem. For the “Scheduler” component, it will be considered part of the APP layer.

7.3 Memory Consumption

7.3.1 Memory Size

Refer to constraints in section 4.7 for details about memory size requirements.

7.3.2 Memory Map

The default memory mapping of the microcontroller shall be used. No specific memory mapping is required for this system.

8 Software Components

8.1 MCAL Subsystem

8.1.1 DIO Component

The DIO (Digital Input Output) component is responsible for reading and writing digital signals to the microcontroller channels.

8.1.1.1 Resources Constraints

None

8.1.1.2 External Interfaces (Dependencies)

{HLD-0059(0)} {Covers: }
{Assignation: DIO} {Test Method: Static Test} {Integration Phase: } {Justification: }

The DIO component shall include the microcontroller header file through “mc9s08jm32.h”.

{HLD-0060(0)} {Covers: }
{Assignation: DIO} {Test Method: Static Test} {Integration Phase: } {Justification: }

The DIO component shall include the basic types through the inclusion of “BTY_int.h”.

8.1.1.3 Design Constraints

None

8.1.1.4 Component APIs

8.1.1.4.1 Symbols

None

8.1.1.4.2 Types

Type			Status
{HLD-0065(0)}		{Covers: }	
{Assignation: DIO}	{Test Method: Integration Test}	{Integration Phase: }	{Justification: }
Channel Groups Structure			New

8.1.1.4.3 Constants

None

8.1.1.4.4 Data

None

8.1.1.4.5 Services

8.1.1.4.5.1 DIO_vidWriteChannelGroup

{HLD-0061(0)}	{Covers: }		
{Assignation: DIO}	{Test Method: Integration Test}	{Integration Phase: }	{Justification: }

Syntax:	`void DIO_vidWriteChannelGroup( DIO_tenuChannelGroupId enuChannelGroupId, uint8 u8Value)`	
Description:	Writes the value to a group of microcontroller channels	
Sync/Async:	Synchronous	
Reentrancy:	non reentrant	
Parameters (in):	enuChannelGroupId	Group ID to be written to (Group 1 / Group 2)
	u8Value	Value to be written to the whole group
Parameters (out):	None	None
Return value:	None	None
Caveats:	None	
Configuration:	None	
MemSegment:	Default compiler memories will be used	
MemClass:	Default compiler memories will be used	

8.1.1.4.5.2 DIO_vidWriteChannel

{HLD-0062(0)}	{Covers: }		
{Assignation: DIO}	{Test Method: Integration Test}	{Integration Phase: }	{Justification: }

Syntax:	`void DIO_vidWriteChannel( DIO_tenuChannelId enuChannelId, DIO_tenuChannelLevel enuLevel )`	
Description:	Writes the value to a single microcontroller channel	
Sync/Async:	Synchronous	
Reentrancy:	non reentrant	
Parameters (in):	enuChannelId	ID of the microcontroller channel to write to
	enuLevel	Channel level (HIGH/LOW)
Parameters (out):	None	None
Return value:	None	None
Caveats:	None	
Configuration:	None	
MemSegment:	Default compiler memories will be used	
MemClass:	Default compiler memories will be used	

8.1.1.4.5.3 DIO_u8ReadChannel

{HLD-0063(0)}	{Covers: }		
{Assignation: DIO}	{Test Method: Integration Test}	{Integration Phase: }	{Justification: }

Syntax:	`uint8 DIO_u8ReadChannel(DIO_tenuChannelId enuChannelId)`	
Description:	Reads the value from a microcontroller channel	
Sync/Async:	Synchronous	
Reentrancy:	non reentrant	
Parameters (in):	enuChannelId	ID of the microcontroller channel to read from
Parameters (out):	None	None
Return value:	uint8	Acquired value read from the channel
Caveats:	None	
Configuration:	None	
MemSegment:	Default compiler memories will be used	
MemClass:	Default compiler memories will be used	

8.1.1.4.6 Tasks

None

8.1.1.4.7 Call-backs

None

8.1.1.4.8 Interrupts

None

8.1.1.5 Component Configuration

Configuration Parameter	**Range**	**Value**
Description		
{HLD-0064(0)}	{Covers: }	
{Assignation: DIO} {Test Method: Integration Test}	{Integration Phase: }	{Justification: }
Channel Groups	Left to the design	N/A
Defines different channel groups from different microcontroller ports into a single structure.		

8.1.2 MCU Component

The MCU component is responsible for the initialization process of the micro-controller that is related to adjusting the clock frequency, defining the interrupt vector table … etc. It only communicates with the scheduler interface to call it when an interrupt occur through “SCHED_vidRtcInterrupt”.

8.1.2.1 Resources Constraints

None

8.1.2.2 External Interfaces (Dependencies)

{HLD-0066(0)} {Covers: }
{Assignation: MCU} {Test Method: Static Test} {Integration Phase: } {Justification: }

The MCU component shall use the service "SCHED_vidRtcInterrupt" from the SCHED component through the inclusion of "SCHED_int.h".

{HLD-0067(0)} {Covers: }
{Assignation: MCU} {Test Method: Static Test} {Integration Phase: } {Justification: }

The MCU component shall include the basic types through the inclusion of "BTY_int.h".

{HLD-0068(0)} {Covers: }
{Assignation: MCU} {Test Method: Static Test} {Integration Phase: } {Justification: }

The MCU component shall include the compiler types through the inclusion of "CTY_int.h".

{HLD-0069(0)} {Covers: }
{Assignation: MCU} {Test Method: Static Test} {Integration Phase: } {Justification: }

The MCU component shall include the microcontroller header file through "mc9s08jm32.h".

8.1.2.3 Design Constraints

None

8.1.2.4 Component APIs

8.1.2.4.1 Symbols

None

8.1.2.4.2 Types

None

8.1.2.4.3 Constants

Constant Name	Type	Value
Description		
{HLD-0070(0)}	{Covers: }	
{Assignation: MCU} {Test Method: Integration Test}	{Integration Phase: }	{Justification: }
MCU_u16INT_VTAB_ADDRESS	array of void * function	N/A
Defines the list of different interrupt functions to be called in case of an interrupt occurred		

8.1.2.4.4 Data

None

8.1.2.4.5 Services

8.1.2.4.5.1 MCU_vidInit

{HLD-0071(0)}	{Covers: }		
{Assignation: MCU}	{Test Method: Integration Test}	{Integration Phase: }	{Justification: }
Syntax:	`void MCU_vidInit(void)`		

Description:	Initializes the microcontroller clock and prepares the RTC	
Sync/Async:	Synchronous	
Reentrancy:	non reentrant	
Parameters (in):	None	None
Parameters (out):	None	None
Return value:	None	None
Caveats:	None	
Configuration:	None	
MemSegment:	Default compiler memories will be used	
MemClass:	Default compiler memories will be used	

{HLD-0072(0)} {Covers: SRS-0040(0)}
{Assignation: MCU} {Test Method: Component Test} {Integration Phase: } {Justification: }

This service shall initialize the PLL to give a frequency of 24MHz.

{HLD-0073(0)} {Covers: }
{Assignation: MCU} {Test Method: Component Test} {Integration Phase: } {Justification: Refer to sec 7.1}

This service shall set the Real Time Counter to fire every 1ms.

8.1.2.4.6 Tasks

None

8.1.2.4.7 Call-backs

None

8.1.2.4.8 Interrupts

None

8.1.2.5 Component Configuration

None

8.1.3 PORT Component

The PORT component is responsible for configuring the microcontroller ports to their different functionalities as per specified in the microcontroller's datasheet and required in the system. It shall be responsible for configuring all the external interfaces defined in section 3.3.

8.1.3.1 Resources Constraints

None

8.1.3.2 External Interfaces (Dependencies)

{HLD-0074(0)} {Covers: }
{Assignation: PORT} {Test Method: Static Test} {Integration Phase: } {Justification: }

The PORT component shall include the microcontroller header file through “mc9s08jm32.h”.

{HLD-0075(0)} {Covers: }
{Assignation: PORT} {Test Method: Static Test} {Integration Phase: } {Justification: }

The PORT component shall include the basic types through the inclusion of “BTY_int.h”.

8.1.3.3 Design Constraints

None

8.1.3.4 Component APIs

8.1.3.4.1 Symbols

None

8.1.3.4.2 Types

Type Name	Type	Members	
Description			
{HLD-0076(0)}		{Covers: }	
{Assignation: PORT}	{Test Method: Integration Test}	{Integration Phase: }	{Justification: }
Channel IDs	enumeration	0..56	
Enumeration to define the different channels available in the microcontroller.			
{HLD-0077(0)}		{Covers: }	
{Assignation: PORT}	{Test Method: Integration Test}	{Integration Phase: }	{Justification: }
Port Direction	enumeration	IN/OUT/UNDEFINED	
Enumeration to define the different port directions available.			

8.1.3.4.3 Constants

None

8.1.3.4.4 Data

None

8.1.3.4.5 Services

8.1.3.4.5.1 PORT_vidInit

{HLD-0078(0)}		{Covers: }	
{Assignation: PORT}	{Test Method: Integration Test}	{Integration Phase: }	{Justification: }
Syntax:	`void PORT_vidInit(void)`		
Description:	Initializes the different microcontroller ports to their correct values		
Sync/Async:	Synchronous		

Reentrancy:	non reentrant	
Parameters (in):	None	None
Parameters (out):	None	None
Return value:	None	None
Caveats:	None	
Configuration:	None	
MemSegment:	Default compiler memories will be used	
MemClass:	Default compiler memories will be used	

{HLD-0079(0)} {Covers: }
{Assignation: PORT} {Test Method: Component Test} {Integration Phase: } {Justification: }

This service shall be responsible for initializing all microcontroller ports as Digital Input Output ports and setting their values as specified in section 3.3.

8.1.3.4.6 Tasks

None

8.1.3.4.7 Call-backs

None

8.1.3.4.8 Interrupts

None

8.1.3.5 Component Configuration

Configuration Parameter	**Range**	**Value**
Description		
{HLD-0080(0)} {Assignation: PORT} {Test Method: Integration Test}	{Covers: } {Integration Phase: }	{Justification: }
Ports Directions	0..7 (maximum # of ports)	N/A
Configuration parameter that defines the directions of each port of the microcontroller. The type “BTY_tuniPort” shall be used to facilitate the definition of ports directions.		

8.2 HAL Subsystem

8.2.1 SSD Component

The SSD component (Seven Segment Driver) is responsible for driving the seven segment displays available in this system with the required value of “8” when ON and assure that when OFF all connected pins to the seven segments are derived OFF. It communicates with the DIO component in the MCAL layer as it uses the service “DIO_vidWriteChannel” from it.

8.2.1.1 Resources Constraints

None

8.2.1.2 External Interfaces (Dependencies)

{HLD-0081(0)} {Covers: }
{Assignation: SSD} {Test Method: Static Test} {Integration Phase: } {Justification: }

The SSD component shall use the service “DIO_vidWriteChannel” from the DIO component through the inclusion of “DIO_int.h”.

{HLD-0082(0)} {Covers: }
{Assignation: SSD} {Test Method: Static Test} {Integration Phase: } {Justification: }

The SSD component shall include the basic types through the inclusion of “BTY_int.h”.

8.2.1.3 Design Constraints

None

8.2.1.4 Component APIs

8.2.1.4.1 Symbols

None

8.2.1.4.2 Types

None

8.2.1.4.3 Constants

None

8.2.1.4.4 Data

None

8.2.1.4.5 Services

8.2.1.4.5.1 SSD_vidInit

{HLD-0083(0)}		{Covers: }	
{Assignation: SSD}	{Test Method: Integration Test}	{Integration Phase: }	{Justification: }

Syntax:	`void SSD_vidInit(void)`	
Description:	Initializes the SSD component	
Sync/Async:	Synchronous	
Reentrancy:	non reentrant	
Parameters (in):	None	None
Parameters (out):	None	None
Return value:	None	None

Caveats:	None
Configuration:	None
MemSegment:	Default compiler memories will be used
MemClass:	Default compiler memories will be used

8.2.1.4.5.2 SSD_SetDisplayedNumber

{HLD-0084(0)}		{Covers: }		
{Assignation: SSD}	{Test Method: Integration Test}	{Integration Phase: }	{Justification: }	
Syntax:	`void SSD_SetDisplayedNumber( SSD_tenuSegmentId enuSegmentId, uint8 u8Number)`			
Description:	Sets the displayed number value on the seven segment displays			
Sync/Async:	Synchronous			
Reentrancy:	non reentrant			
Parameters (in):	enuSegmentId	Segment to display the value on (right or left)		
	u8Number	Number to be displayed on the segment		
Parameters (out):	None	None		
Return value:	None	None		
Caveats:	None			
Configuration:	None			
MemSegment:	Default compiler memories will be used			
MemClass:	Default compiler memories will be used			

{HLD-0085(0)} {Covers: }
{Assignation: SSD} {Test Method: Component Test} {Integration Phase: } {Justification: }

This service sets a shared local variable with the value to be displayed on the seven segment present on board that is identified by the segment ID parameter.

8.2.1.4.6 Tasks

8.2.1.4.6.1 SSD_vidTask

{HLD-0086(0)}		{Covers: }		
{Assignation: SSD}	{Test Method: Integration Test}	{Integration Phase: }	{Justification: }	
Syntax:	`void SSD_vidTask(void)`			
Description:	Task of the SSD component			
Sync/Async:	Synchronous			
Reentrancy:	non reentrant			
Parameters (in):	None	None		
Parameters (out):	None	None		
Return value:	None	None		
Caveats:	None			

Configuration:	None
MemSegment:	Default compiler memories will be used
MemClass:	Default compiler memories will be used

{HLD-0087(0)} {Covers: SRS-0016(0)}
{Assignation: SSD} {Test Method: Component Test} {Integration Phase: } {Justification: }

In case of the SSD is controlling the RIGHT SSD, SS_SEG1 signal shall be issued with value HIGH.

{HLD-0088(0)} {Covers: SRS-0016(0)}
{Assignation: SSD} {Test Method: Component Test} {Integration Phase: } {Justification: }

In case of the SSD is controlling the LEFT SSD, SS_SEG2 signal shall be issued with value HIGH.

{HLD-0089(0)} {Covers: SRS-0017(0)}
{Assignation: SSD} {Test Method: Component Test} {Integration Phase: } {Justification: }

In case of the SSD is controlling the RIGHT SSD, SS_SEG2 signal shall be issued with value LOW.

{HLD-0090(0)} {Covers: SRS-0017(0)}
{Assignation: SSD} {Test Method: Component Test} {Integration Phase: } {Justification: }

In case of the SSD is controlling the LEFT SSD, SS_SEG1 signal shall be issued with value LOW.

{HLD-0091(0)} {Covers: SRS-0014(0)}
{Assignation: SSD} {Test Method: Component Test} {Integration Phase: } {Justification: }

This task will issue a HIGH value on the corresponding SSD desired pins (as defined in the SRS) through the call to the service DIO_vidWriteChannelGroup to display the required value that is stored in the shared local variable previously set in SSD_SetDisplayedNumber.

{HLD-0092(0)} {Covers: SRS-0015(0)}
{Assignation: SSD} {Test Method: Component Test} {Integration Phase: } {Justification: }

This task will issue a LOW value on the corresponding SSD desired pins (as defined in the SRS) through the call to the service DIO_vidWriteChannelGroup to display the required value that is stored in the shared local variable previously set in SSD_SetDisplayedNumber.

{HLD-0106(0)} {Covers: SRS-0014(0)}
{Assignation: SSD} {Test Method: Component Test} {Integration Phase: } {Justification: }

To switch-on a certain segment of the Seven Segment Displays (a, b, c, d, e, f, g, or dp), the corresponding digital signal (SS_A, SS_B, SS_C, SS_D, SS_E, SS_F, SS_G, or SS_DOT respectively) -connected to the segment’s pin of the SDD chips- must be issued with the value HIGH.

{HLD-0107(0)} {Covers: SRS-0015(0)}
{Assignation: SSD} {Test Method: Component Test} {Integration Phase: } {Justification: }

To switch-off a certain segment of the Seven Segment Displays (a, b, c, d, e, f, g, or dp), the corresponding digital signal (SS_A, SS_B, SS_C, SS_D, SS_E, SS_F, SS_G, or SS_DOT respectively) -connected to the segment’s pin of the SDD chips- must be issued with the value LOW.

{HLD-0108(0)} {Covers: SRS-0016(0)}
{Assignation: SSD} {Test Method: Component Test} {Integration Phase: } {Justification: }

To enable the operation of LEFT_SIGNAL_SSD or RIGHT_SIGNAL_SSD, the corresponding digital signals (SS_SEG2 or SS_SEG1 respectively) shall be issued with the value HIGH.

{HLD-0109(0)} {Covers: SRS-0017(0)}
{Assignation: SSD} {Test Method: Component Test} {Integration Phase: } {Justification: }

To disable the operation of LEFT_SIGNAL_SSD or RIGHT_SIGNAL_SSD, the corresponding digital signals (SS_SEG2 or SS_SEG1 respectively) shall be issued with the value LOW.

8.2.1.4.7 Call-backs

None

8.2.1.4.8 Interrupts

None

8.2.1.5 Component Configuration

None

8.2.2 PBD Component

The PBD component (Push Buttons Driver) is responsible for detecting a correct press on the push buttons connected on the microcontroller pins. It communicates with the DIO component in the MCAL layer as it uses the service "DIO_u8ReadChannel" from it.

8.2.2.1 Resources Constraints

None

8.2.2.2 External Interfaces (Dependencies)

{HLD-0093(0)} {Covers: }
{Assignation: PBD} {Test Method: Static Test} {Integration Phase: } {Justification: }

The PBD component shall use the service "DIO_u8ReadChannel" from the DIO component through the inclusion of "DIO_int.h".

{HLD-0094(0)} {Covers: }
{Assignation: PBD} {Test Method: Static Test} {Integration Phase: } {Justification: }

The PBD component shall include the basic types through the inclusion of "BTY_int.h".

8.2.2.3 Design Constraints

None

8.2.2.4 Component APIs

8.2.2.4.1 Symbols

None

8.2.2.4.2 Types

Type Name	Type	Members
Description		
{HLD-0098(0)}	{Covers: }	
{Assignation: PBD} {Test Method: Integration Test}	{Integration Phase: }	{Justification: }
Push Button State	enumeration	PRE_PUSH, PUSHED, PRE_HOLD, HOLD, PRE_RELEASE, RELEASED, UNDEFINED
To manage the push buttons, this enumeration is defined to ease the management.		

8.2.2.4.3 Constants

None

8.2.2.4.4 Data

None

8.2.2.4.5 Services

8.2.2.4.5.1 PBD_vidInit

{HLD-0095(0)}		{Covers: }	
{Assignation: PBD}	{Test Method: Integration Test}	{Integration Phase: }	{Justification: }

Syntax:	`void PBD_vidInit(void)`	
Description:	Initializes the PBD component	
Sync/Async:	Synchronous	
Reentrancy:	non reentrant	
Parameters (in):	None	None
Parameters (out):	None	None
Return value:	None	None
Caveats:	None	
Configuration:	None	
MemSegment:	Default compiler memories will be used	
MemClass:	Default compiler memories will be used	

8.2.2.4.5.2 PBD_enuGetButtonState

{HLD-0099(0)}		{Covers: }	
{Assignation: PBD}	{Test Method: Integration Test}	{Integration Phase: }	{Justification: }

Syntax:	`PBD_tenuButtonState PBD_enuGetButtonState(`
	`PBD_tenuButtonId enuButtonId)`	

Description:	Sets the displayed number value on the seven segment displays	
Sync/Async:	Synchronous	
Reentrancy:	non reentrant	
Parameters (in):	enuButtonId	Button to check its status (down, up, hazard)
Parameters (out):	PBD_tenuButtonState	State of button as described in the types above
Return value:	None	None
Caveats:	None	
Configuration:	None	
MemSegment:	Default compiler memories will be used	
MemClass:	Default compiler memories will be used	

{HLD-0100(0)} {Covers: }
{Assignation: PBD} {Test Method: Component Test} {Integration Phase: } {Justification: }

This service reads the state of the push button and returns it back to the caller. If the button ID passed is undefined, this service shall return UNDEFINED state.

8.2.2.4.6 Tasks

8.2.2.4.6.1 PBD_vidTask

{HLD-0096(0)}		{Covers: }	
{Assignation: PBD}	{Test Method: Integration Test}	{Integration Phase: }	{Justification: }

Syntax:	`void PBD_vidTask(void)`	
Description:	Task of the SSD component	
Sync/Async:	Synchronous	
Reentrancy:	non reentrant	
Parameters (in):	None	None
Parameters (out):	None	None
Return value:	None	None
Caveats:	None	
Configuration:	None	
MemSegment:	Default compiler memories will be used	
MemClass:	Default compiler memories will be used	

{HLD-0101(0)} {Covers: }
{Assignation: PBD} {Test Method: Component Test} {Integration Phase: } {Justification: }

The task shall loop on all buttons of the system and check its state.

{HLD-0102(0)} {Covers: SRS-0018(0)}
{Assignation: PBD} {Test Method: Component Test} {Integration Phase: } {Justification: }

For HAZZARD_PB, the press is considered as valid once it’s released after being pressed for at least 200 ms. Therefore a transition from HIGH state to LOW state (negative edge) must be detected on the signal PB_PRESS3 followed by a minimum of 200 ms at the LOW state (signal stabilization time) followed by a transition from LOW state to HIGH state (positive edge).

{HLD-0103(0)} {Covers: SRS-0019(0)}
{Assignation: PBD} {Test Method: Component Test} {Integration Phase: } {Justification: }

For STEP_DOWN_PB, the press is considered as valid once it’s pressed for at least 10 ms. Therefore a transition from HIGH state to LOW state (negative edge) must be detected on the signal PB_PRESS1 followed by a minimum of 10 ms at the LOW state (signal stabilization time).

{HLD-0104(0)} {Covers: SRS-0020(0)}
{Assignation: PBD} {Test Method: Component Test} {Integration Phase: } {Justification: }

For STEP_UP_PB, the press is considered as valid once pressed for at least 10 ms. Therefore a transition from HIGH state to LOW state (negative edge) must be detected on the signal PB_PRESS2 followed by a minimum of 10 ms at the LOW state (signal stabilization time).

{HLD-0105(0)} {Covers: SRS-0021(0)}
{Assignation: PBD} {Test Method: Component Test} {Integration Phase: } {Justification: }

If a push button is being pressed, any other push button press shall be ignored.

8.2.2.4.7 Call-backs

None

8.2.2.4.8 Interrupts

None

8.2.2.5 Component Configuration

None

8.3 APP Subsystem

8.3.1 HIF Component

The HIF component is responsible for handling hazard signal application requirements. It only communicates externally with the lower SSD component for the service "SSD_SetDisplayedNumber".

8.3.1.1 Resources Constraints

None, HIF is on the hardware abstraction layer.

8.3.1.2 External Interfaces (Dependencies)

{HLD-0042(0)} {Covers: }
{Assignation: HIF} {Test Method: Static Test} {Integration Phase: } {Justification: }

The HIF component shall use the service "SSD_SetDisplayedNumber" from the SSD component through the inclusion of "SSD_int.h".

{HLD-0043(0)} {Covers: }
{Assignation: HIF} {Test Method: Static Test} {Integration Phase: } {Justification: }

The HIF component shall include the basic types through the inclusion of "BTY_int.h".

8.3.1.3 Design Constraints

None

8.3.1.4 Component APIs

8.3.1.4.1 Symbols

None

8.3.1.4.2 Types

None

8.3.1.4.3 Constants

None

8.3.1.4.4 Data

None

8.3.1.4.5 Services

8.3.1.4.5.1 HIF_vidInit

{HLD-0044(0)}	{Covers: }		
{Assignation: HIF}	{Test Method: Integration Test}	{Integration Phase: }	{Justification: }

Syntax:	`void HIF_vidInit(void)`	
Description:	Initializes the HIF component	
Sync/Async:	Synchronous	
Reentrancy:	non reentrant	
Parameters (in):	None	None
Parameters (out):	None	None
Return value:	None	None
Caveats:	None	
Configuration:	None	
MemSegment:	Default compiler memories will be used	
MemClass:	Default compiler memories will be used	

8.3.1.4.5.2 HIF_vidSetMode

{HLD-0045(0)}		{Covers: }	
{Assignation: HIF}	{Test Method: Integration Test}	{Integration Phase: }	{Justification: }
Syntax:	`void HIF_vidSetMode(HIF_tenuMode enuHifSetMode)`		
Description:	Sets the mode of the HIF component		
Sync/Async:	Synchronous		
Reentrancy:	non reentrant		
Parameters (in):	enuHifSetMode	HIF_OFF_MODE HIF_ON_MODE HIF_UNDEFINED_MODE	
Parameters (out):	None	None	
Return value:	None	None	
Caveats:	None		
Configuration:	None		
MemSegment:	Default compiler memories will be used		
MemClass:	Default compiler memories will be used		

8.3.1.4.6 Tasks

8.3.1.4.6.1 HIF_vidTask

{HLD-0046(0)}		{Covers: }	
{Assignation: HIF}	{Test Method: Integration Test}	{Integration Phase: }	{Justification: }
Syntax:	`void HIF_vidTask(void)`		
Description:	Task of the HIF component		
Sync/Async:	Synchronous		
Reentrancy:	non reentrant		
Parameters (in):	None	None	

Parameters (out):	None	None
Return value:	None	None
Caveats:	None	
Configuration:	None	
MemSegment:	Default compiler memories will be used	
MemClass:	Default compiler memories will be used	

{HLD-0047(0)} {Covers: SRS-0036(0)}
{Assignation: HIF} {Test Method: Component Test} {Integration Phase: } {Justification: }

If the system is not in the HAZARD_BLINK state, the task is responsible to call the SSD interface to command both SSDs with output values LOW to disable both SSDs on board.

{HLD-0048(0)} {Covers: SRS-0036(0)}
{Assignation: HIF} {Test Method: Component Test} {Integration Phase: } {Justification: }

If the system is in the HAZARD_BLINK state, the task is responsible to call the SSD interface to command both SSDs with output values HIGH to display value 8 on both SSDs for a duration of 200ms ± 5 ms and command the output values LOW to disable both SSDs for a duration of 400ms ± 5 ms.

8.3.1.4.7 Callbacks

None

8.3.1.4.8 Interrupts

None

8.3.1.5 Component Configuration

None

8.3.2 TIF Component

The TIF component is responsible for handling the turning signals (turn right/left) application requirements. It only communicates externally with the lower SSD component for the service “SSD_SetDisplayedNumber”.

8.3.2.1 Resources Constraints

None, TIF is on the hardware abstraction layer.

8.3.2.2 External Interfaces (Dependencies)

{HLD-0050(0)} {Covers: }
{Assignation: TIF} {Test Method: Static Test} {Integration Phase: } {Justification: }

The TIF component shall use the service “SSD_SetDisplayedNumber” from the SSD component through the inclusion of “SSD_int.h”.

{HLD-0051(0)} {Covers: }
{Assignation: TIF} {Test Method: Static Test} {Integration Phase: } {Justification: }

The HIF component shall include the basic types through the inclusion of “BTY_int.h”.

8.3.2.3 Design Constraints

None

8.3.2.4 Component APIs

8.3.2.4.1 Symbols

None

8.3.2.4.2 Types

None

8.3.2.4.3 Constants

None

8.3.2.4.4 Data

None

8.3.2.4.5 Services

8.3.2.4.5.1 TIF_vidInit

{HLD-0052(0)}		{Covers: }	
{Assignation: TIF}	{Test Method: Integration Test}	{Integration Phase: }	{Justification: }
Syntax:	`void TIF_vidInit(void)`		
Description:	Initializes the TIF component		
Sync/Async:	Synchronous		
Reentrancy:	non reentrant		
Parameters (in):	None	None	
Parameters (out):	None	None	
Return value:	None	None	
Caveats:	None		
Configuration:	None		
MemSegment:	Default compiler memories will be used		
MemClass:	Default compiler memories will be used		

8.3.2.4.5.2 TIF_vidSetMode

{HLD-0053(0)}		{Covers: }	
{Assignation: TIF}	{Test Method: Integration Test}	{Integration Phase: }	{Justification: }
Syntax:	`void TIF_vidSetMode(TIF_enuMode enuTifSetMode)`		
Description:	Sets the mode of the TIF component		
Sync/Async:	Synchronous		
Reentrancy:	non reentrant		

Parameters (in):	enuTifSetMode	TIF_OFF_MODE TIF_RIGHT_BLINK_MODE TIF_LEFT_BLINK_MODE TIF_UNDEFINED_MODE
Parameters (out):	None	None
Return value:	None	None
Caveats:	None	
Configuration:	None	
MemSegment:	Default compiler memories will be used	
MemClass:	Default compiler memories will be used	

8.3.2.4.6 Tasks

8.3.2.4.6.1 TIF_vidTask

{HLD-0054(0)}		{Covers: }	
{Assignation: TIF}	{Test Method: Integration Test}	{Integration Phase: }	{Justification: }

Syntax:	`void TIF_vidTask(void)`	
Description:	Task of the TIF component	
Sync/Async:	Synchronous	
Reentrancy:	non reentrant	
Parameters (in):	None	None
Parameters (out):	None	None
Return value:	None	None
Caveats:	None	
Configuration:	None	
MemSegment:	Default compiler memories will be used	
MemClass:	Default compiler memories will be used	

{HLD-0055(0)} {Covers: SRS-0032(0)}
{Assignation: TIF} {Test Method: Component Test} {Integration Phase: } {Justification: }

If the system is in the RIGHT_BLINK state, the task is responsible to call the SSD interface to command the right SSDs with output values HIGH to display value 8 for a duration of 200ms ± 5 ms and command the output values LOW to disable it for a duration of 300ms ± 5 ms.

{HLD-0056(0)} {Covers: SRS-0026(0)}
{Assignation: TIF} {Test Method: Component Test} {Integration Phase: } {Justification: }

If the system is in the RIGHT_BLINK state, the task is responsible to call the SSD interface to command the left SSD with all values LOW to ensure that the left SSD is not working.

{HLD-0057(0)} {Covers: SRS-0026(0)}
{Assignation: TIF} {Test Method: Component Test} {Integration Phase: } {Justification: }

If the system is in the LEFT_BLINK state, the task is responsible to call the SSD interface to command the left SSDs with output values HIGH to display value 8 for a duration of 200ms ± 5 ms and command the output values LOW to disable it for a duration of 300ms ± 5 ms.

{HLD-0058(0)} {Covers: SRS-0026(0)}
{Assignation: TIF} {Test Method: Component Test} {Integration Phase: } {Justification: }

If the system is in the LEFT_BLINK state, the task is responsible to call the SSD interface to command the right SSD with all values LOW to ensure that the right SSD is not working.

8.3.2.4.7 Callbacks

None

8.3.2.4.8 Interrupts

None

8.3.2.5 Component Configuration

None

8.3.3 MODE Component

The MODE manager is a central component present at the application layer and communicates both vertically and horizontally in the architecture with the TIF, HIF and PBD components to: get the state of the push buttons “PBD_enuGetButtonState”, set the mode of the TIF “TIF_vidSetMode” and set the mode of the HIF “HIF_vidSetMode”.

8.3.3.1 Resources Constraints

None

8.3.3.2 External Interfaces (Dependencies)

{HLD-0111(0)} {Covers: }
{Assignation: MODE} {Test Method: Static Test} {Assignation: MODE} {Test Method: Static Test}

The MODE component shall use the service “PBD_enuGetButtonState” from the PBD component through the inclusion of “PBD_int.h”.

{HLD-0112(0)} {Covers: }
{Assignation: MODE} {Test Method: Static Test} {Assignation: MODE} {Test Method: Static Test}

The MODE component shall use the service “HIF_vidSetMode” from the HIF component through the inclusion of “HIF_int.h”.

{HLD-0113(0)} {Covers: }
{Assignation: MODE} {Test Method: Static Test} {Assignation: MODE} {Test Method: Static Test}

The MODE component shall use the service “TIF_vidSetMode” from the TIF component through the inclusion of “TIF_int.h”.

8.3.3.3 Design Constraints

None

8.3.3.4 Component APIs

8.3.3.4.1 Symbols

None

8.3.3.4.2 Types

None

8.3.3.4.3 Constants

None

8.3.3.4.4 Data

None

8.3.3.4.5 Services

8.3.3.4.5.1 MODE_vidInit

{HLD-0114(0)}		{Covers: }	
{Assignation: MODE}	{Test Method: Integration Test}	{Integration Phase: }	{Justification: }

Syntax:	`void MODE_vidInit(void)`	
Description:	Initializes the MODE component	
Sync/Async:	Synchronous	
Reentrancy:	non reentrant	
Parameters (in):	None	None
Parameters (out):	None	None
Return value:	None	None
Caveats:	None	
Configuration:	None	
MemSegment:	Default compiler memories will be used	
MemClass:	Default compiler memories will be used	

{HLD-0117(0)} {Covers: SRS-0022(0)}
{Assignation: MODE} {Test Method: Component Test} {Integration Phase: } {Justification: }

Upon startup, this service should put the system in IDLE state.

{HLD-0118(0)} {Covers: SRS-0023(0)}
{Assignation: MODE} {Test Method: Component Test} {Integration Phase: } {Justification: }

Upon startup, both LEFT and RIGHT SSDs should be switched off.

8.3.3.4.6 Tasks

8.3.3.4.6.1 MODE_vidTask

{HLD-0115(0)}		{Covers: }	
{Assignation: MODE}	{Test Method: Integration Test}	{Integration Phase: }	{Justification: }

Syntax:	`void MODE_vidTask(void)`	
Description:	Task of the MODE component	
Sync/Async:	Synchronous	
Reentrancy:	non reentrant	
Parameters (in):	None	None
Parameters (out):	None	None
Return value:	None	None

Caveats:	None
Configuration:	None
MemSegment:	Default compiler memories will be used
MemClass:	Default compiler memories will be used

{HLD-0116(0)} {Covers: }
{Assignation: MODE} {Test Method: Component Test} {Integration Phase: } {Justification: }

This task shall grab the state of different push buttons and switch between states accordingly.

{HLD-0119(0)} {Covers: SRS-0025(0)}
{Assignation: MODE} {Test Method: Component Test} {Integration Phase: } {Justification: }

If the system is at the IDLE state and a valid press was detected on STEP_DOWN_PB, the system shall go to the LEFT_BLINK state.

{HLD-0120(0)} {Covers: SRS-0030(0)}
{Assignation: MODE} {Test Method: Component Test} {Integration Phase: } {Justification: }

If the system is at the IDLE state and a valid press was detected on STEP_UP_PB, the system shall go to the RIGHT_BLINK state.

{HLD-0121(0)} {Covers: SRS-0031(0)}
{Assignation: MODE} {Test Method: Component Test} {Integration Phase: } {Justification: }

If the system is at the IDLE state and a valid press was detected on HAZZARD_PB, the system shall go to the HAZZARD_BLINK state.

{HLD-0122(0)} {Covers: SRS-0027(0)}
{Assignation: MODE} {Test Method: Component Test} {Integration Phase: } {Justification: }

If the system is at the LEFT_BLINK state and a valid press was detected on STEP_UP_PB, the system shall go to the IDLE state.

{HLD-0123(0)} {Covers: SRS-0028(0)}
{Assignation: MODE} {Test Method: Component Test} {Integration Phase: } {Justification: }

If the system is at the LEFT_BLINK state and a valid press was detected on STEP_DOWN_PB, the system shall remain in the LEFT_BLINK state.

{HLD-0124(0)} {Covers: SRS-0029(0)}
{Assignation: MODE} {Test Method: Component Test} {Integration Phase: } {Justification: }

If the system is at the LEFT_BLINK state and a valid press was detected on HAZZARD_PB, the system shall go to the HAZARD_BLINK state.

{HLD-0125(0)} {Covers: SRS-0033(0)}
{Assignation: MODE} {Test Method: Component Test} {Integration Phase: } {Justification: }

If the system is at the RIGHT_BLINK state and a valid press was detected on STEP_DOWN_PB, the system shall go to the IDLE state.

{HLD-0126(0)} {Covers: SRS-0034(0)}
{Assignation: MODE} {Test Method: Component Test} {Integration Phase: } {Justification: }

If the system is at the RIGHT_BLINK state and a valid press was detected on STEP_UP_PB, the system shall remain in the RIGHT_BLINK state.

{HLD-0127(0)} {Covers: SRS-0035(0)}
{Assignation: MODE} {Test Method: Component Test} {Integration Phase: } {Justification: }

If the system is at the RIGHT_BLINK state and a valid press was detected on HAZZARD_PB, the system shall go to the HAZARD_BLINK state.

{HLD-0128(0)} {Covers: SRS-0037(0)}
{Assignation: MODE} {Test Method: Component Test} {Integration Phase: } {Justification: }

If the system is at the HAZZARD_BLINK state and a valid press was detected on STEP_UP_PB or STEP_DOWN_PB, the system shall remain in the HAZZARD_BLINK state.

{HLD-0129(0)} {Covers: SRS-0038(0)}
{Assignation: MODE} {Test Method: Component Test} {Integration Phase: } {Justification: }

If the system is at the HAZZARD_BLINK state and a valid press was detected on HAZZARD_PB, the system shall go back to its previous state before going to HAZARD_BLINK state.*

* For example if the system was in IDLE state then a valid press was detected on HAZZARD_PB moving the system to HAZARD_BLINK state, if another valid press was detected on HAZZARD_PB the system shall go back to the IDLE state.

8.3.3.4.7 Call-backs

None

8.3.3.4.8 Interrupts

None

8.3.3.5 Component Configuration

None

8.3.4 SCHED Component

The SCHEDular component is responsible for operating a mini real time system that controls the different tasks through handling the RTC (Real Time Counter) interrupt and running the tasks accordingly. It communicates with MCU, SSD, PBD, MODE, TIF and HIF components as it is present horizontally in the system.

8.3.4.1 Resources Constraints

None

8.3.4.2 External Interfaces (Dependencies)

{HLD-0130(0)} {Covers: }
{Assignation: SCHED} {Test Method: Static Test} {Integration Phase: } {Justification: }

The SCHED component shall include the microcontroller header file through “mc9s08jm32.h”.

{HLD-0131(0)} {Covers: }
{Assignation: SCHED} {Test Method: Static Test} {Integration Phase: } {Justification: }

The SCHED component shall include the basic types through the inclusion of “BTY_int.h”.

{HLD-0132(0)} {Covers: }
{Assignation: SCHED} {Test Method: Static Test} {Integration Phase: } {Justification: }

The SCHED component shall include the compiler types through the inclusion of “CTY_int.h”.

{HLD-0133(0)} {Covers: }
{Assignation: SCHED} {Test Method: Static Test} {Integration Phase: } {Justification: }

The SCHED component shall use the service “PBD_vidTask” from the PBD component through the inclusion of “PBD_int.h”.

{HLD-0134(0)} {Covers: }
{Assignation: SCHED} {Test Method: Static Test} {Integration Phase: } {Justification: }

The SCHED component shall use the service “SSD_vidTask” from the SSD component through the inclusion of “SSD_int.h”.

{HLD-0135(0)} {Covers: }
{Assignation: SCHED} {Test Method: Static Test} {Integration Phase: } {Justification: }

The SCHED component shall use the service “MODE_vidTask” from the MODE component through the inclusion of “MODE_int.h”.

{HLD-0136(0)} {Covers: }
{Assignation: SCHED} {Test Method: Static Test} {Integration Phase: } {Justification: }

The SCHED component shall use the service “HIF_vidTask” from the HIF component through the inclusion of “HIF_int.h”.

{HLD-0137(0)} {Covers: }

{Assignation: SCHED} {Test Method: Static Test} {Integration Phase: } {Justification: }

The SCHED component shall use the service “TIF_vidTask” from the TIF component through the inclusion of “TIF_int.h”.

8.3.4.3 Design Constraints

None

8.3.4.4 Component APIs

8.3.4.4.1 Symbols

None

8.3.4.4.2 Types

Type Name	**Type**	**Members**
Description		
{HLD-0138(0)}	{Covers: }	
{Assignation: SCHED} {Test Method: Integration Test}	{Integration Phase: }	{Justification: }
Task Control Block	structure	pointer a task to execute task delay (0 if not needed) task period
Structure to manage the different tasks that are available in the system.		

8.3.4.4.3 Constants

None

8.3.4.4.4 Data

None

8.3.4.4.5 Services

8.3.4.4.5.1 SCHED_vidInit

{HLD-0139(0)}	{Covers: }	
{Assignation: SCHED} {Test Method: Integration Test}	{Integration Phase: }	{Justification: }
Syntax:	`void SCHED_vidInit(void)`	
Description:	Initializes the SCHED component	
Sync/Async:	Synchronous	
Reentrancy:	non reentrant	
Parameters (in):	None	None
Parameters (out):	None	None
Return value:	None	None

Caveats:	None
Configuration:	None
MemSegment:	Default compiler memories will be used
MemClass:	Default compiler memories will be used

{HLD-0141(0)} {Covers: }
{Assignation: SCHED} {Test Method: Component Test} {Integration Phase: } {Justification: }

The Init function should initialize a local array with the configured values delays for each task and put an initial ACTIVE state for all tasks available.

8.3.4.4.5.2 SCHED_vidStartScheduler

{HLD-0142(0)}		{Covers: }	
{Assignation: SCHED}	{Test Method: Integration Test}	{Integration Phase: }	{Justification: }
Syntax:	`void SCHED_vidStartScheduler(void)`		
Description:	Starts the main scheduler of the system		
Sync/Async:	Synchronous		
Reentrancy:	non reentrant		
Parameters (in):	None	None	
Parameters (out):	None	None	
Return value:	None	None	
Caveats:	None		
Configuration:	None		
MemSegment:	Default compiler memories will be used		
MemClass:	Default compiler memories will be used		

{HLD-0143(0)} {Covers: }
{Assignation: SCHED} {Test Method: Component Test} {Integration Phase: } {Justification: }

This function shall check the current tick and calls the corresponding task if the task is active.

8.3.4.4.6 Tasks

None

8.3.4.4.7 Call-backs

None

8.3.4.4.8 Interrupts

8.3.4.4.8.1 SCHED_vidRtcInterrupt

{HLD-0144(0)}		{Covers: <Covered Requirements Tags>}	
{Assignation: SCHED}	{Test Method: Integration Test}	{Integration Phase: }	{Justification: }
Syntax:	`<CTY_INTERRUPT void SCHED_vidRtcInterrupt(void)`		
Description:	Implementation of the RTC interrupt		

Sync/Async:	Synchronous	
Reentrancy:	non reentrant	
Parameters (in):	None	None
Parameters (out):	None	None
Return value:	None	None
Caveats:	None	
Configuration:	None	
MemSegment:	Default compiler memories will be used	
MemClass:	Default compiler memories will be used	

{HLD-0145(0)} {Covers: }
{Assignation: SCHED} {Test Method: Component Test} {Integration Phase: } {Justification: }

This interrupt should increment the tick of the timer that is used by the scheduler to assist it in correctly running the different interrupts on time.

8.3.4.5 Component Configuration

Configuration Parameter	**Range**	**Value**
Description		
{HLD-0140(0)}	{Covers: }	
{Assignation: SCHED} {Test Method: Integration Test}	{Integration Phase: } {Justification: }	
Tasks Array	0..NumberOfTasks (=5)	N/A
Array of type “Task Control Block” that defines the different tasks and their periodicity.		

9 Sequence Management

9.1 Initialization Sequence

None

9.2 Sequence Diagrams

None

10 Software Integration Constraints

{HLD-0110(0)} {Covers: SRS-0047(0), SRS-0048(0), SRS-0049(0)}

{Assignation: } {Test Method: Static Test} {Integration Phase: } {Justification: }

Refer to section 7 in REF2 for more details.