Static Design Analysis

For ECU1 and ECU2

Abdelrahman Yasser

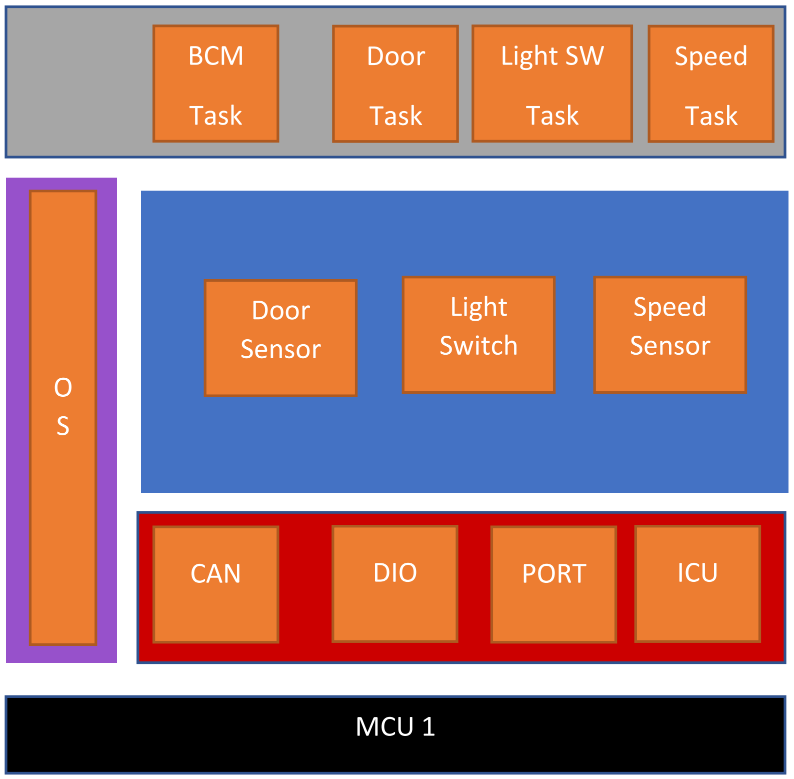
Sprints Embedded software design

FWD July cohort

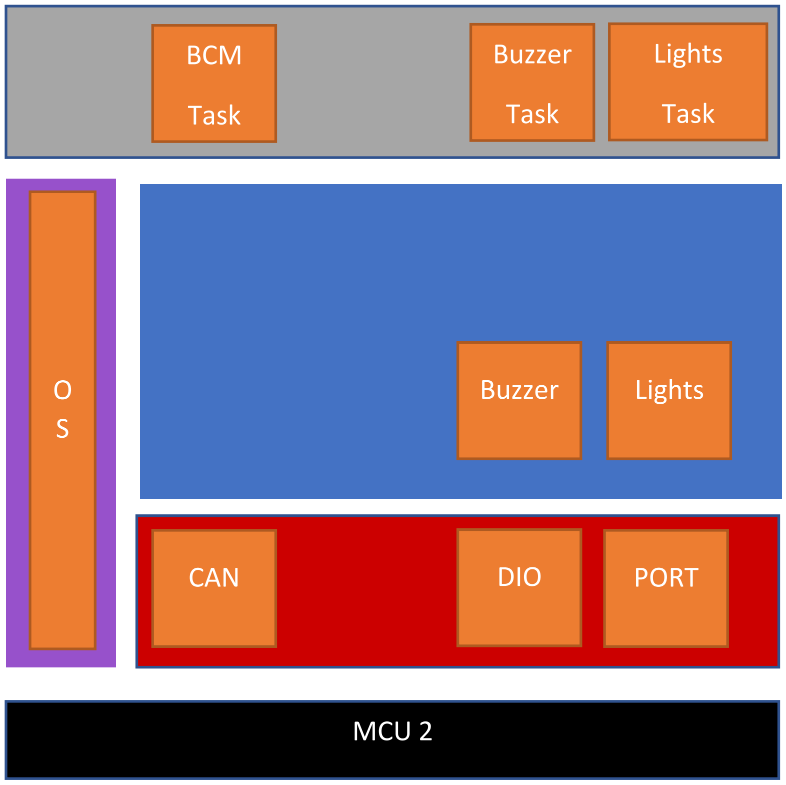
Contents

Layered Architecture

ECU 1:



ECU 2:



API specification

Speed sensor module

Type definitions

Speed_MeasuringUnitType

Name:	Speed_MeasuringUnitType	
Type:	Enum	
Range:	SPEED_KM_PER_HOUR	Speed is measured in km/h
	SPEED_M_PER_HOUR	Speed is measured in m/h
Description:	Speed units supported by module	

Speed_Type

Name:	Speed_Type
Type:	uint
Range:	< 0 – Max speed used in application >
Description:	Speed data

Function definitions

Speed_Init

Name:	Speed_Init	
Syntex:	`void Speed_Init( void )`	
Parameters:	None	
Return:	None	
Description:	Initialization function for speed module	

Speed_GetCurrentSpeed

Name:	Speed_GetCurrentSpeed	
Syntex:	`Speed_Type Speed_GetCurrentSpeed( void )`	
Parameters:	None	
Return:	Speed_Type	Speed of the vechicle
Description:	Function to read current speed	

Speed_SetSpeedUnit

Name:	Speed_SetSpeedUnit	
Syntex:	`void Speed_SetSpeedUnit ( Speed_MeasuringUnitType unit)`	
Parameters:	unit	Speed measuring unit (km/hr or m/hr)
Return:	None	
Description:	Function to set measuring unit of speed	

Switch module

Type definitions

Switch_ChannelType

Name:	Switch_ChannelType
Type:	uint
Range:	< 0 – number of switches used in application >
Description:	Numeric ID of a switch instance

Switch_StateType

Name:	Switch_StateType	
Type:	Enum	
Range:	SWITCH_IS_ON	Switch in on state
	SWITCH_IS_OFF	Switch in off state
Description:	Switch logic states	

Function definitions

Switch_Init

Name:	Switch_Init	
Syntex:	`void Switch_Init( void )`	
Parameters:	None	
Return:	None	
Description:	Initialization function for switch module	

Switch_GetSwitchState

Name:	Switch_GetSwitchState	
Syntex:	`Switch_StateType Switch_GetSwitchState( Switch_ChannelType channel)`	
Parameters:	channel	Switch ID
Return:	Switch_StateType	State of the switch
Description:	Function to get switch state	

Door Sensor module

Type definitions

Door_ChannelType

Name:	Door_ChannelType
Type:	uint8
Range:	< 0 – number of doors used in application >
Description:	Numeric ID of a door instance

Door_StateType

Name:	Door_StateType	
Type:	Enum	
Range:	DOOR_IS_OPEN	Door opened
	DOOR_IS_CLOSED	Door closed
Description:	Door logic states	

Function definitions

Door_Init

Name:	Door_Init	
Syntex:	`void Door_Init( void )`	
Parameters:	None	
Return:	None	
Description:	Initialization function for door module	

Door_GetDoorState

Name:	Door_GetDoorState	
Syntex:	`Door_StateType Door_GetDoorState(Door_ChannelType channel )`	
Parameters:	channel	Door ID
Return:	Door_StateType	State of the door
Description:	Function to get door state	

Buzzer module

Type definitions

Buzzer_ChannelType

Name:	Buzzer_ChannelType
Type:	uint
Range:	< 0 – number of buzzers used in application >
Description:	Numeric ID of a buzzer instance

Function definitions

Buzzer_Init

Name:	Buzzer_Init	
Syntex:	`void Buzzer_Init( void )`	
Parameters:	None	
Return:	None	
Description:	Initialization function for buzzer module	

Buzzer_SetBuzzerOn

Name:	Buzzer_SetBuzzerOn	
Syntex:	`void Buzzer_SetBuzzerOn ( Buzzer_ChannelType channel)`	
Parameters:	channel	Buzzer ID
Return:	None	
Description:	Function to turn on the buzzer	

Buzzer_SetBuzzerOff

Name:	Buzzer_SetBuzzerOff	
Syntex:	`void Buzzer_SetBuzzerOff ( Buzzer_ChannelType channel)`	
Parameters:	channel	Buzzer ID
Return:	None	
Description:	Function to turn off the buzzer	

Light module

Type definitions

Light_ChannelType

Name:	Light_ChannelType
Type:	uint
Range:	< 0 – number of lights used in application >
Description:	Numeric ID of a light instance

Light_LevelType

Name:	Light_LevelType	
Type:	uint8	
Range:	LIGHT_OFF	Depends on physical connection logic either 0 volts
	LIGHT _ON	Or (3.3 / 5) volt
Description:	Light levels	

Function definitions

Light_Init

Name:	Light_Init	
Syntex:	`void Light_Init( void )`	
Parameters:	None	
Return:	None	
Description:	Initialization function for light module	

Light_SetLightOn

Name:	Light_SetLightOn	
Syntex:	`void Light_SetLightOn ( Light_ChannelType channel)`	
Parameters:	channel	light ID
Return:	None	
Description:	Function to turn on the light	

Light_SetLightOff

Name:	Light_SetLightOff	
Syntex:	`void Light_SetLightOff ( Light_ChannelType channel)`	
Parameters:	channel	light ID
Return:	None	
Description:	Function to turn off the light	

Light_GetLightState

Name:	Light_GetLightState

Syntex:	`Light_LevelType Light_GetLightState (Light_ChannelType channel )`	
Parameters:	channel	Light ID
Return:	Light_LevelType	State of the light
Description:	Function to get light state	

Port module

Type definitions

Port_PinType

Name:	Port_PinType
Type:	uint
Range:	< 0 – number of hardware pins available >
Description:	Numeric ID of a pin instance

Port_PinDirectionType

Name:	Port_PinDirectionType	
Type:	Enum	
Range:	PORT_PIN_INPUT	Pin is input
	PORT_PIN_OUTPUT	Pin is output
Description:	Pin directions	

Port_PinModeType

Name:	Port_PinModeType
Type:	Implementation specific
Description:	Pin modes available on the target MCU

Port_PinInternalAttachType

Name:	Port_PinDirectionType
Type:	Implementation specific
Description:	Pin Pull modes available on the target MCU

Port_PinOutputCurrentType

Name:	Port_PinOutputCurrentType
Type:	Implementation specific
Description:	Pin output current modes available on the target MCU

Port_ConfigType

Name:	Port_ConfigType
Type:	struct
Range:	<Implementation specific>
Description:	Configuration structure that holds the port pin config

Function definitions

Port_Init

Name:	Port_Init	
Syntex:	`void Port_Init( void )`	
Parameters:	None	
Return:	None	
Description:	Initialization function for Port module	

Dio module

Type definitions

Dio_ChannelType

Name:	Dio_ChannelType
Type:	uint
Range:	< 0 – number of hardware pins available >
Description:	Numeric ID of a pin(channel) instance

Dio_PortType

Name:	Dio_PortType
Type:	uint8
Range:	< 0 – number of hardware ports available >
Description:	Numeric ID of a port instance

Dio_LevelType

Name:	Dio_LevelType	
Type:	uint8	
Range:	0x0	Pin logic Low
	0x1	Pin logic High
Description:	Pin logic levels	

Dio_PortLevelType

Name:	Dio_LevelType	
Type:	uint	
Range:	0 .. <2 ^ (port bits) >	Whole port level
Description:	Port logic levels	

Function definitions

Dio_ReadChannel

Name:	Dio_ReadChannel	
Syntex:	`Dio_LevelType Dio_ReadChannel( Dio_ChannelType channel)`	
Parameters:	channel	Pin ID
Return:	Dio_LevelType	
Description:	Function to read from a Dio pin	

Dio_WriteChannel

Name:	Dio_WriteChannel	
Syntex:	`void Dio_WriteChannel( Dio_ChannelType channel, Dio_LevelType level )`	
Parameters:	channel	Pin ID
	level	Pin Output level
Return:	None	
Description:	Function to write to a Dio pin	

Dio_ReadPort

Name:	Dio_ReadPort	
Syntex:	`Dio_PortLevelType Dio_ReadPort( Dio_PortType port)`	
Parameters:	port	Port ID
Return:	Dio_PortLevelType	
Description:	Function to read from a Dio port	

DioWritePort

Name:	Dio_WritePort	
Syntex:	`void Dio_WritePort( Dio_PortType port, Dio_PortLevelType level )`	
Parameters:	channel	Port ID
	level	Port Output level
Return:	None	
Description:	Function to write to a Dio Port	

DioFlipChannel

Name:	Dio_FlipChannel	
Syntex:	`void Dio_FlipChannel( Dio_ChannelType channel )`	
Parameters:	channel	Pin ID
Return:	None	
Description:	Function to flip a Dio pin	

Can module

This specification supports only 1 on-board CAN2.0 module for the sake of simplicity.

Data definitions

Can_ConfigType

Name:	Can_ConfigType
Type:	struct
Range:	<Implementation specific>
Description:	Configuration structure that holds the can config parameters

Can_MessageType

Name:	Can_MessageType		
Type:	struct		
Elements:	uint32	id	Can Id which fill up arbitration field
	uint8	length	Length of the data (Control field)
	uint8 *	data	Pointer to the data
Description:	Configuration structure that holds the can config parameters		

Can_ReturnType

Name:	Can_ReturnType	
Type:	enum	
Elements:	CAN_OK	No errors
	CAN_NOT_OK	Error
	CAN_BUSY	Can bus is busy
Description:	Can return type for error checking	

Function definitions

Can_Init

Name:	Can_Init	
Syntex:	`void Can_Init( Can_ConfigType *ConfigPtr )`	
Parameters:	ConfigPtr	Pointer to can configuration structure.
Return:	None	
Description:	Initialization function for Can module	

Can_Write

Name:	Can_Write	
Syntex:	`Can_ReturnType Can_Write ( Can_MessageType *message )`	
Parameters:	`message`	Pointer to can message buffer
Return:	`Can_ReturnType`	Transmition error status
Description:	Function to write on can bus	

Can_Read

Name:	Can_Read	
Syntex:	`Can_ReturnType Can_Read ( Can_MessageType *message )`	
Parameters:	`message`	Pointer to can message buffer
Return:	`Can_ReturnType`	Transmition error status
Description:	Function to read from can bus	