-

egypt fwd
initiative

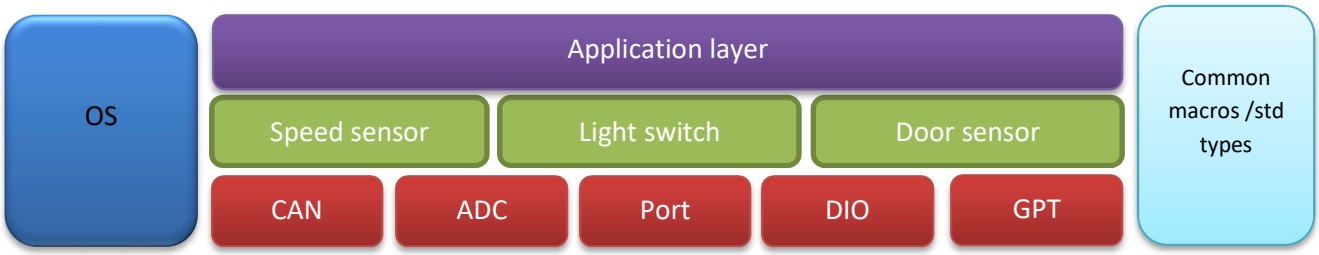
The design project

Static design

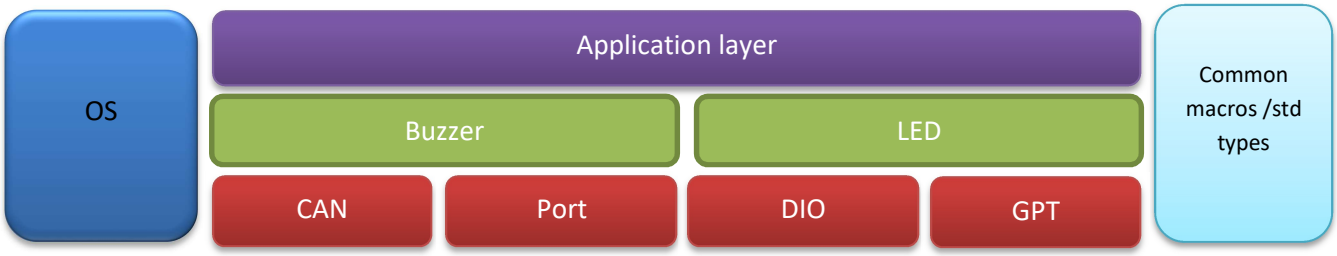
-

1- Layerd architecture and modules

For ECU 1



For ECU 2



-

2- detailed APIs for ECU1 and ECU2

a- common mcal layer for the two microcontrollers

PORT module

-Type definitions

1- Port_ConfigType

name	Port_ConfigType
Kind	Structure
Elements	Hardware Dependent Structure
Description	Type of the external data structure containing the initialization data for this module.
Available via	Port.h

2- Port_PinType

name	Port_PinType
Kind	Type
Derived from	uint
range	0 to number of port pins
Description	Data type for the symbolic name of a port pin.
Available via	Port.h

-

3- Port_PinDirectionType

Name	Port_PinDirectionType		
Type	Enumeration		
Range	PORT_PIN_IN	0x00	Sets port pin as input.
	PORT_PIN_OUT	0x01	Sets port pin as output.
Description	Possible directions of a port pin.		
Available via	Port.h		

4- Port_PinModeType

Name	Port_PinModeType
Derived from	uint
range	Implementation specific
Description	Different port pin modes.
Available via	Port.h

-Function definitions

Port_Init

Service Name	Port_Init		
Syntax	void Port_Init (const Port_ConfigType* ConfigPtr)		
Reentrancy	Non Reentrant		
Parameters	Inputs	ConfigPtr	Port_ConfigType
		Pointer to configuration set.	
	Output	None	
	Input/output	None	
Return value	None		
Description	Initializes the Port Driver module.		
Available via	Port.h		

-

DIO module

-Type definitions

1-Dio_ChannelType

name	Dio_ChannelType
Kind	Type
Derived from	uint
range	This is implementation specific but not all values may be valid within the type.
Description	Numeric ID of a DIO channel.
Available via	Dio.h

2- Dio_PortType

name	Dio_PortType
Kind	Type
Derived from	uint
range	0..<number of ports>
Description	Numeric ID of a DIO port.
Available via	Dio.h

3- Dio_LevelType

name	Dio_LevelType		
Kind	Type		
Derived from	Uint8		
range	STD_LOW	0x00	Physical state 0V
	STD_HIGH	0x01	Physical state 5V or 3.3V
Description	These are the possible levels a DIO channel can have (input or output)		

-

Available via	Dio.h

4- Dio_PortLevelType

name	Dio_PortLevelType
Kind	Type
Derived from	uint
range	0 .. xxx depends on the number of pins
Description	If the µC owns ports of different port widths (e.g. 4, 8,16...Bit) Dio_PortLevelType inherits the size of the largest port.
Available via	Dio.h

-Function definitions

1- Dio_ReadChannel

Service Name	Dio_ReadChannel		
Syntax	Dio_LevelType Dio_ReadChannel (Dio_ChannelType ChannelId)		
Reentrancy	Reentrant		
Parameters	Inputs	ChannelId	Dio_ChannelType
		ID of DIO channel	
	Output	None	
	Input/output	None	
Return value	Dio_Level-Type		
Description	Initializes the Port Driver module.		
Available via	Port.h		

-

2- Dio_WriteChannel

Service Name	Dio_WriteChannel		
Syntax	void Dio_WriteChannel (Dio_ChannelType ChannelId, Dio_LevelType Level)		
Reentrancy	Reentrant		
Parameters	Inputs	ChannelId	Dio_ChannelType
		ID of DIO channel	
		Level	Dio_LevelType
		Value to be written	
	Output	None	
	Input/output	None	
Return value	None		
Description	Service to set a level of a channel.		
Available via	Port.h		

3- Dio_ReadPort

Service Name	Dio_ReadPort		
Syntax	Dio_PortLevelType Dio_ReadPort (Dio_PortType PortId)		
Reentrancy	Reentrant		
Parameters	Inputs	PortId	Dio_PortType
		ID of DIO Port	
	Output	None	

-

	Input/output	None
Return value	Dio_PortLevelType	
Description	Returns the level of all channels of that port.	
Available via	Port.h	

4- Dio_WritePort

Service Name	Dio_WritePort		
Syntax	void Dio_WritePort (Dio_PortType PortId, Dio_PortLevelType Level)		
Reentrancy	Reentrant		
Parameters	Inputs	PortId	Dio_PortType
		ID of DIO Port	
		Level	PortLevelType
		Value to be written	
	Output	None	
	Input/output	None	
Return value	None		
Description	Service to set a value of the port.		
Available via	Port.h		

CAN module

-Type definitions

1-Can_ConfigType

name	Can_ConfigType
Kind	Structure
Description	This is the type of the external data structure containing the overall initialization data for the CAN driver and SFR settings affecting all controllers. Furthermore it contains pointers to controller configuration

-

	structures. The contents of the initialization data structure are CAN hardware specific.
Available via	Can.h

2-Can_ControllerStateType

name	Can_ControllerStateType		
Kind	Enumeration		
range	CAN_CS_UNINIT	0x00	CAN controller state UNINIT
	CAN_CS_STARTED	0x01	CAN controller state STARTED.
	CAN_CS_STOPPED	0x02	CAN controller state STOPPED.
	CAN_CS_SLEEP	0x03	CAN controller state SLEEP.
Description	States that are used by the several ControllerMode functions.		
Available via	Can_GeneralTypes.h		

-Function definitions

1-Can_Init

Service Name	Can_Init		
Syntax	void Can_Init (const Can_ConfigType* Config)		
Reentrancy	Non Reentrant		
Parameters	Inputs	Config	const Can_ConfigType*
		Pointer to driver configuration.	
	Output	None	
	Input/output	None	
Return value	None		

-

Description	This function initializes the module.
Available via	Can.h

2- Can_DeInit

Service Name	Can_DeInit	
Syntax	void Can_DeInit (void)	
Reentrancy	Non Reentrant	
	Inputs	None
	Output	None
	Input/output	None
Return value	None	
Description	This function de-initializes the module.	
Available via	Can.h	

3- Can_SetBaudrate

Service Name	Can_SetBaudrate		
Syntax	Std_ReturnType Can_SetBaudrate (uint8 Controller, uint16 BaudRateConfigID)		
Reentrancy	Reentrant for different Controllers. Non reentrant for the same Controller.		
Parameters	Inputs	Controller	uint8
		CAN controller, whose baud rate shall be set	
		BaudRateConfigID	uint16
		references a baud rate configuration by ID (see CanControllerBaudRateConfigID)	
	Output	None	

-

	Input/output	None
Return value	Std_ReturnType	-E_OK: Service request accepted, setting of (new) baud rate started -E_NOT_OK: Service request not accepted
Description	This service shall set the baud rate configuration of the CAN controller. Depending on necessary baud rate modifications the controller might have to reset.	
Available via	Can.h	

4-Can_SetControllerMode

Service Name	Can_SetControllerMode		
Syntax	Std_ReturnType Can_SetControllerMode (uint8 Controller, Can_ControllerStateType Transition)		
Reentrancy	Non Reentrant		
Parameters	Inputs	Controller	uint8
		CAN controller, whose baud rate shall be set	
		Transition	Can_ControllerStateType
		Transition value to request new CAN controller state	
	Output	None	
	Input/output	None	
Return value	Std_ReturnType	E_OK: Service request accepted, setting of (new) baud rate started E_NOT_OK: Service request not accepted	
Description	This function performs software triggered state transitions of the CAN controller State machine.		
Available via	Can.h		

5-Can_MainFunction_Write

-

Service Name	Can_MainFunction_Write	
Syntax	void Can_MainFunction_Write (void)	
Reentrancy	Non reentrant	
Parameters	Inputs	None
	Output	None
	Input/output	None
Return value	None	
Description	This function performs the polling of TX confirmation when CAN_TX_PROCESSING is set to POLLING.	
Available via	SchM_Can.h	

6-Can_MainFunction_Read

Service Name	Can_MainFunction_Read	
Syntax	void Can_MainFunction_Read (void)	
Reentrancy	Non reentrant	
Parameters	Inputs	None
	Output	None
	Input/output	None
Return value	None	
Description	This function performs the polling of RX indications when CAN_RX_PROCESSING is set to POLLING.	
Available via	SchM_Can.h	

GPT module

-

-Type definitions

1-Gpt_ConfigType

name	Gpt_ConfigType
Kind	Structure
Description	Implementation specific configuration data structure, see chapter 10 for configurable parameters.
Available via	Gpt.h

2- Gpt_ChannelType

name	Gpt_ChannelType
Kind	Type
Derived from	uint
range	Implementation specific. But not all values may be valid within this type. This type shall be chosen in order to have the most efficient implementation on a specific micro controller platform
Description	Numeric ID of a GPT channel.
Available via	Gpt.h

3-Gpt_ValueType

name	Gpt_ValueType
Kind	Type
Derived from	uint
range	The range of this type is µC dependent (width of the timer register) and has to be described by the supplier.
Description	Type for reading and setting the timer values (in number of ticks).
Available via	Gpt.h

-

4-Gpt_ModeType

name	Gpt_ModeType		
Kind	Enumeration		
range	GPT_MODE_NORMAL	0x00	Normal operation mode of the GPT
	GPT_MODE_SLEEP	0x01	Operation for reduced power operation mode. In sleep mode only wakeup capable channels are available.
Description	Modes of the GPT driver.		
Available via	Gpt.h		

-Function definitions

1-Gpt_Init

Service Name	Gpt_Init		
Syntax	void Gpt_Init (const Gpt_ConfigType* ConfigPtr)		
Reentrancy	Non Reentrant		
Parameters	Inputs	ConfigPtr	const Gpt_ConfigType*
		Pointer to a selected configuration structure	
	Output	None	
	Input/output	None	
Return value	None		
Description	Initializes the GPT driver.		
Available via	Gpt.h		

2- Gpt_StartTimer

-

Service Name	Gpt_StartTimer		
Syntax	void Gpt_StartTimer (Gpt_ChannelType Channel, Gpt_ValueType Value)		
Reentrancy	Reentrant (but not for the same timer channel)		
Parameters	Inputs	Channel	Gpt_ChannelType
		Numeric identifier of the GPT channel.	
		Value	Gpt_ValueType
		Target time in number of ticks.	
	Output	None	
	Input/output	None	
Return value	None		
Description	Starts a timer channel.		
Available via	Gpt.h		

3- Gpt_StopTimer

Service Name	Gpt_StopTimer	
Syntax	void Gpt_StopTimer (Gpt_ChannelType Channel)	
Reentrancy	Reentrant (but not for the same timer channel)	
Parameters	Inputs	None
	Output	None
	Input/output	None
Return value	None	
Description	Stops a timer channel.	
Available via	Gpt.h	

-

b- Adc module for ECU1

ADC module

-Type definitions

1-Adc_ConfigType

name	Adc_ConfigType
Kind	Structure
Elements	Implementation specific configuration data structure.
Description	Data structure containing the set of configuration parameters required for initializing the ADC Driver and ADC HW Unit(s).
Available via	Adc.h

2-Adc_ChannelType

Name	Adc_ChannelType
Kind	Type
Derived from	uint
range	The range of this type is µC specific and has to be described by the supplier.
Description	Numeric ID of an ADC channel.
Available via	Adc.h

-Function definitions

1- Adc_Init

Service Name	Adc_Init

-

Syntax	void Adc_Init (const Adc_ConfigType* ConfigPtr)		
Reentrancy	Non reentrant		
Parameters	Inputs	ConfigPtr	const Adc_ConfigType*
		Pointer to configuration set in Variant PB (Variant PC requires a NULL_PTR).	
	Output	None	
	Input/output	None	
Return value	None		
Description	Initializes the ADC hardware units and driver.		
Available via	Adc.h		

2- ADC_ReadChannel

Service Name	ADC_ReadChannel		
Syntax	uint32 ADC_readChannel(Adc_ChannelType CH_num)		
Reentrancy	Non reentrant		
Parameters	Inputs	CH_num	Adc_ChannelType
		The channel wanted to be read	
	Output	None	
	Input/output	None	
Return value	The channel value		
Description	Returns the value of the specified ADC Channel		
Available via	Adc.h		

c- On board layer for ECU1

Door_sensor module

-

Must include Dio.h , Port.h

-Type definitions

1-Door_State

Name	Door_State		
Kind	Type		
Derived from	Uint8		
range	open	0x00	When the door opened
	closed	0x01	When the door closed
Description	Descripes the state of the door		
Available via	Door_sensor.h		

-Function definitions

1-Door_Init

Service Name	Door_Init	
Syntax	void Door_Init (void)	
Reentrancy	Non reentrant	
Parameters	Inputs	None
	Output	None
	Input/output	None
Return value	None	
Description	Initializes the module	
Available via	Door_sensor.h	

2-Door_Get_state

-

Service Name	Door_Get_state	
Syntax	Door_state Door_Get_state (void)	
Reentrancy	Non reentrant	
Parameters	Inputs	None
	Output	None
	Input/output	None
Return value	Door_state	
Description	Returns the current state of the door	
Available via	Door_sensor.h	

Light_Switch module

Must include Dio.h , Port.h

-Type definitions

1-Switch_State

Name	switch_State		
Kind	Type		
Derived from	Uint8		
range	on	0x00	When the switch pressed
	off	0x01	When the switch released
Description	Descripes the state of the light switch		
Available via	Light_Switch.h		

-Function definitions

-

1-Switch_Init

Service Name	Switch_Init	
Syntax	void Switch_Init (void)	
Reentrancy	Non reentrant	
Parameters	Inputs	None
	Output	None
	Input/output	None
Return value	None	
Description	Initializes the module	
Available via	Light_Switch.h	

2-Switch_Get_state

Service Name	Switch_Get_state	
Syntax	Switch_state Switch_Get_state (void)	
Reentrancy	Non reentrant	
Parameters	Inputs	None
	Output	None
	Input/output	None
Return value	Switch_state	
Description	Returns the current state of the Light switch	
Available via	Door_sensor.h	

Speed_Sensor module

-

Must include Adc.h , Port.h

-Function definitions

1-Speed_Sensor_Init

Service Name	Speed_Sensor_Init	
Syntax	void Speed_Sensor_Init (void)	
Reentrancy	Non reentrant	
Parameters	Inputs	None
	Output	None
	Input/output	None
Return value	None	
Description	Initializes the module	
Available via	Speed_Sensor.h	

2-Speed_Get_Value

Service Name	Speed_Get_Value	
Syntax	Uint32 Speed_Get_Value (void)	
Reentrancy	Non reentrant	
Parameters	Inputs	None
	Output	None
	Input/output	None
Return value	Uint32	
Description	Returns the current car speed	
Available via	Speed_Sensor.h	

d- On board layer for ECU2

-

LED module

Must include Dio.h , Port.h

-Type definitions

1-LED_Type

Name	LED_Type		
Kind	enumeration		
range	Left	0x00	L light is chosen
	Right	0x01	R light is chosen
Description	Chooses which light		
Available via	Led.h		

-Function definitions

1-LED_Init

Service Name	LED_Init	
Syntax	void LED_Init (void)	
Reentrancy	Non reentrant	
Parameters	Inputs	None
	Output	None
	Input/output	None
Return value	None	
Description	Initializes the Led module	
Available via	Led.h	

2-Turn_LED_On

-

Service Name	Turn_LED_On		
Syntax	void Turn_LED_On (LED_Type type)		
Reentrancy	reentrant		
Parameters	Inputs	type	determines the required led
	Output	None	
	Input/output	None	
Return value	None		
Description	Turn on the requird led		
Available via	Led.h		

3-Turn_LED_Off

Service Name	Turn_LED_Off		
Syntax	void Turn_LED_Off (LED_Type type)		
Reentrancy	reentrant		
Parameters	Inputs	type	determines the required led
	Output	None	
	Input/output	None	
Return value	None		
Description	Turn off the requird led		
Available via	Led.h		

Buzzer module

-

Must include Dio.h , Port.h

-Function definitions

1-Buzzer_Init

Service Name	Buzzer_Init	
Syntax	void Buzzer_Init (void)	
Reentrancy	Non reentrant	
Parameters	Inputs	None
	Output	None
	Input/output	None
Return value	None	
Description	Initializes the Buzzer module	
Available via	Buzzer.h	

2-Turn_Buzzer_On

Service Name	Turn_Buzzer_On	
Syntax	void Turn_Buzzer_On (void)	
Reentrancy	reentrant	
	Input	None
	Output	None
	Input/output	None
Return value	None	
Description	Turn on the Buzzer	
Available via	Buzzer.h	

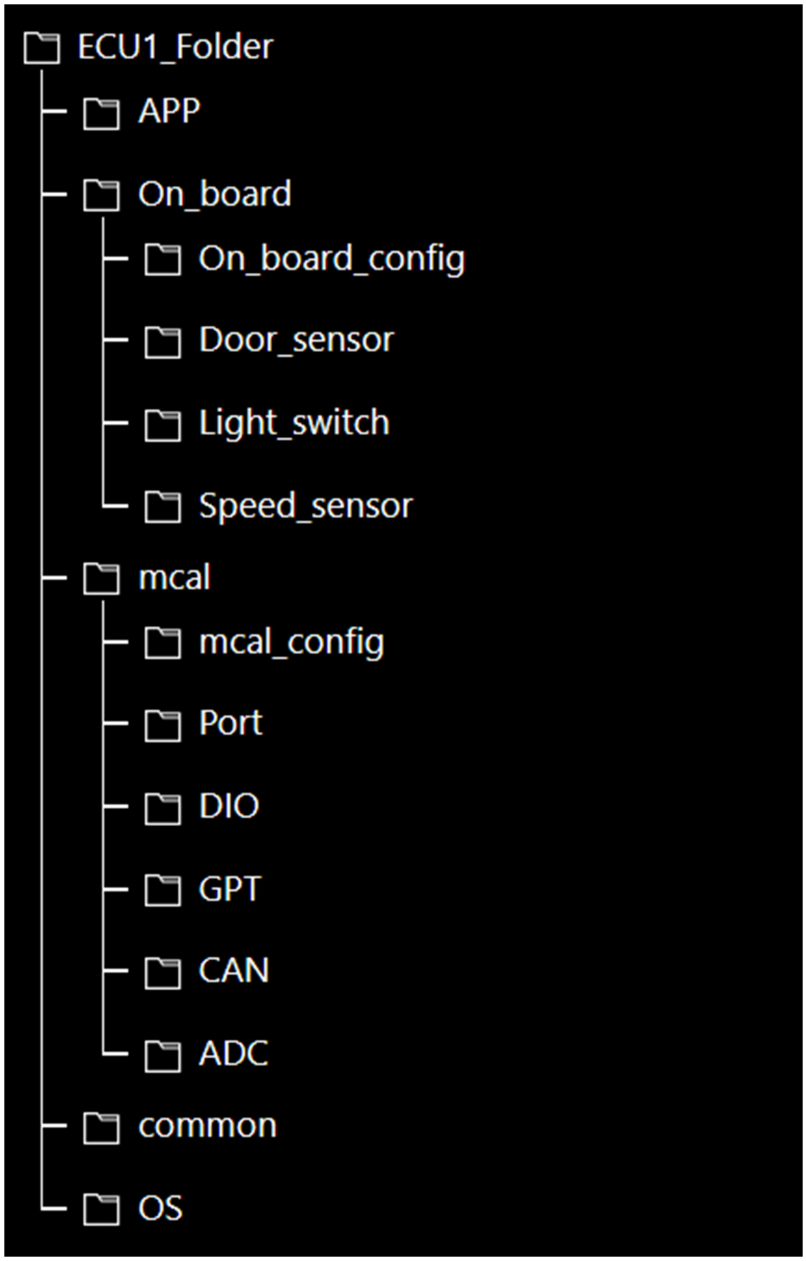
3-Turn_Buzzer_Off

-

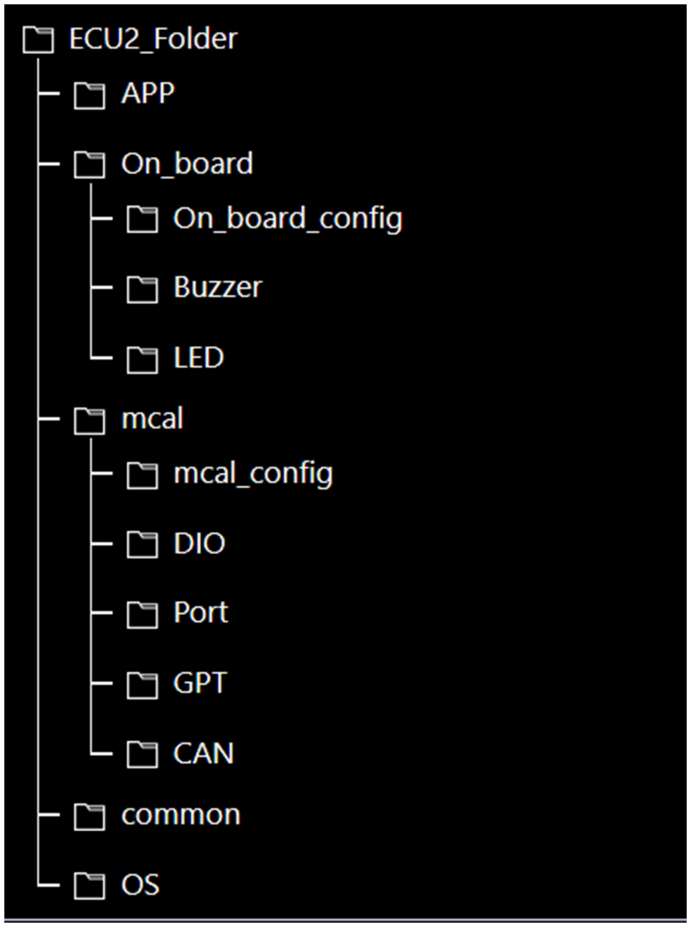
Service Name	Turn_Buzzer_Off	
Syntax	void Turn_Buzzer_Off (void)	
Reentrancy	reentrant	
	Input	None
	Output	None
	Input/output	None
Return value	None	
Description	Turn off the Buzzer	
Available via	Buzzer.h	

3- Folder structure for ECU1 and ECU2

-


ECU1_Folder
APP
On_board
On_board_config
Door_sensor
Light_switch
Speed_sensor
mcal
mcal_config
Port
DIO
GPT
CAN
ADC
common
OS

-



Each module will contain

1-.c file for the functions and data definitions

2-.h files for the module data types and function prototypes

3-Lconfig.c file for the linking configuration (contains definitions)

4-config.h file for the preprocessing configuration