# ECU1 Static Design

## Make the layered architecture

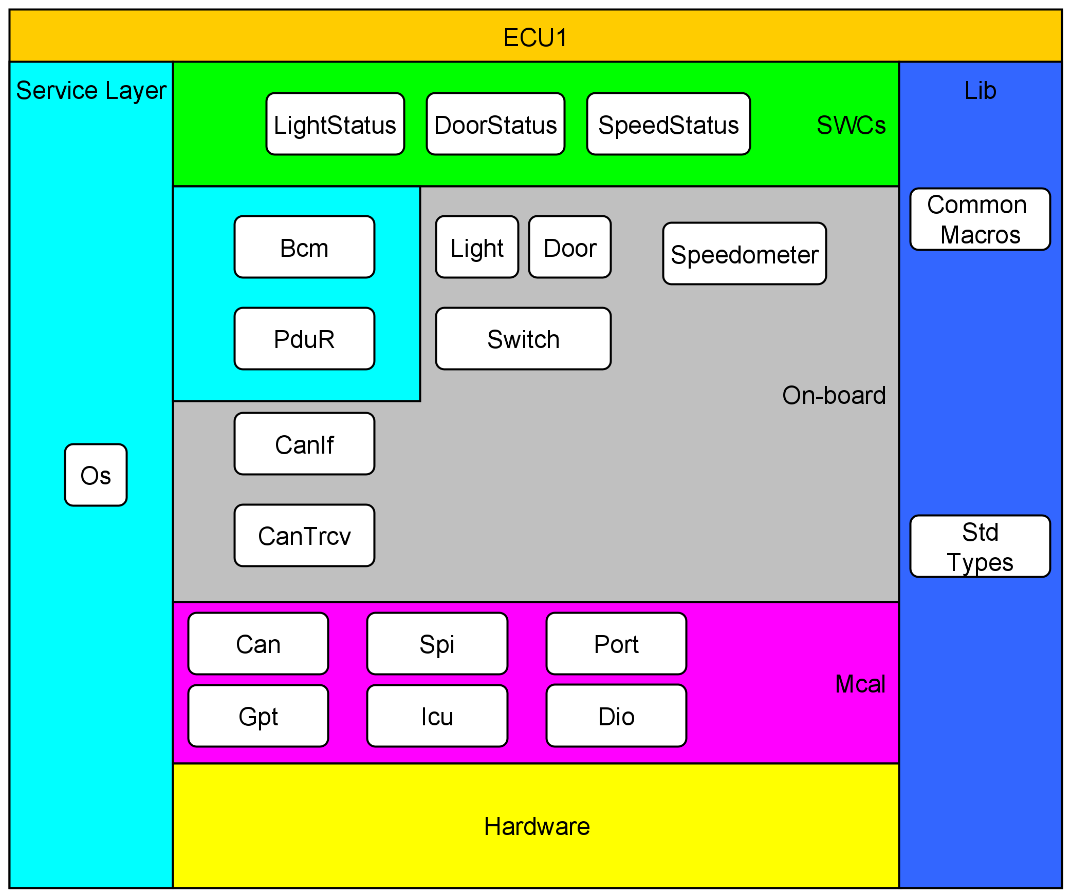


Figure 1 ECU1 static design

## Specify ECU components and modules

### Application layer

**LightStatus**

This software component is responsible of reading the light status and send it through CAN protocol.

**DoorStatus**

This software component is responsible of reading the door sensor status and send it through CAN protocol.

**SpeedStatus**

This software component is responsible of reading the speed sensor status and determine whether the car is stopped or moving. Then, it sends this status through CAN protocol.

### Service Layer

**Bcm**

This module is responsible of transmitting/receiving the CAN signals to/from the SWCs. It abstracts the Pdu Information from the application layer.

**PduR**

This module is responsible for routing the PDUs to the right communication interface e.g. (CanIf). It abstracts the ECU layer communication protocols from the upper layer.

### Ecu Layer (on board)

**Light**

This module is responsible to abstract the port and pin of the light switches from the application layer.

**Door**

This module is responsible to abstract the port and pin of the door sensor from the application layer.

**Switch**

This module is responsible to dealing with debouncing of the light switch and the door sensor.

**Speedometer**

This module is responsible for calculating the speed of the car from the Input Capture Unit value and sets the speed status.

**CanIf**

This module is responsible for abstracting the different CAN controllers from the upper layer.

**CanTrcv**

This module is responsible for abstracting the manipulation of the CAN transceiver configuration from the upper layer.

### Mcal Layer

**Port**

This driver is responsible for the configuration of GPIOs.

**Dio**

This driver is responsible for reading or writing from/to GPIO pins.

**Can**

This is the can driver that deal with the mailboxes and CAN controllers’ configuration.

**Spi**

This is the spi driver that’s used by CanTrcv module to configure the CAN transceivers.

**Gpt**

This is the general-purpose timer driver. It’s used by the scheduler as a tick source.

**Icu**

This is the driver of the Input Capture Unit which is used to interface with the speed sensor.

## Provide full detailed APIs for each module as well as a detailed description for the used typedefs

**Note: the following documentation is generated using DoxyGen**

### src/1-Appl/DoorStatus/inc/DoorStatus.h File Reference

This module is responsible for getting the door switch status (closed or open) and send it on the CAN bus.

#include "DoorStatus\_Types.h"

#### Functions

* void **DoorStatus\_Update** (void)

*Get the door status and send it through CAN protocol.*

#### Detailed Description

This module is responsible for getting the door switch status (closed or open) and send it on the CAN bus.

###### Author

Mohamed Hassanin (mohamed\_hassanin\_omran@yahoo.com)

###### Version

0.1

###### Date

2022-09-03

###### Copyright

Copyright (c) 2022

#### Function Documentation

##### void DoorStatus\_Update (void )

Get the door status and send it through CAN protocol.

### src/1-Appl/DoorStatus/inc/DoorStatus\_Types.h File Reference

This module is responsible for getting the door switch status (closed or open) and send it on the CAN bus.

#### Detailed Description

This module is responsible for getting the door switch status (closed or open) and send it on the CAN bus.

###### Author

Mohamed Hassanin (mohamed\_hassanin\_omran@yahoo.com)

###### Version

0.1

###### Date

2022-09-03

###### Copyright

Copyright (c) 2022

### src/1-Appl/LightStatus/inc/LightStatus.h File Reference

This module is responsible for getting the light switch status (pressed or released) and send it on the CAN bus.

#include "LightStatus\_Types.h"

#### Functions

* void **LightStatus\_Update** (void)

*Get the light switch status and send it through CAN protocol.*

#### Detailed Description

This module is responsible for getting the light switch status (pressed or released) and send it on the CAN bus.

###### Author

Mohamed Hassanin (mohamed\_hassanin\_omran@yahoo.com)

###### Version

0.1

###### Date

2022-09-03

###### Copyright

Copyright (c) 2022

#### Function Documentation

##### void LightStatus\_Update (void )

Get the light switch status and send it through CAN protocol.

### src/1-Appl/LightStatus/inc/LightStatus\_Types.h File Reference

This module is responsible for getting the light switch status (pressed or released) and send it on the CAN bus.

#### Detailed Description

This module is responsible for getting the light switch status (pressed or released) and send it on the CAN bus.

###### Author

Mohamed Hassanin (mohamed\_hassanin\_omran@yahoo.com)

###### Version

0.1

###### Date

2022-09-03

###### Copyright

Copyright (c) 2022

### src/1-Appl/SpeedStatus/inc/SpeedStatus.h File Reference

This module is responsible for getting the speed switch status and send it on the CAN bus.

#include "SpeedStatus\_Types.h"

#### Functions

* void **SpeedStatus\_Update** (void)

*Get the speedometer status and send the car status (moving or stopped) signal through CAN protocol.*

#### Detailed Description

This module is responsible for getting the speed switch status and send it on the CAN bus.

###### Author

Mohamed Hassanin (mohamed\_hassanin\_omran@yahoo.com)

###### Version

0.1

###### Date

2022-09-03

###### Copyright

Copyright (c) 2022

#### Function Documentation

##### void SpeedStatus\_Update (void )

Get the speedometer status and send the car status (moving or stopped) signal through CAN protocol.

### src/1-Appl/SpeedStatus/inc/SpeedStatus\_Types.h File Reference

This module is responsible for getting the speed switch status and send it on the CAN bus.

#### Detailed Description

This module is responsible for getting the speed switch status and send it on the CAN bus.

###### Author

Mohamed Hassanin (mohamed\_hassanin\_omran@yahoo.com)

###### Version

0.1

###### Date

2022-09-03

###### Copyright

Copyright (c) 2022

### src/2-Service/Bcm/inc/Bcm.h File Reference

This module is responsible for handling basic communication and mapping signals to the right Pdu Ids. It abstracts the Pdu meta data e.g. PduId.

#include "Bcm\_Types.h"

#include "ComStack\_Types.h"

#### Functions

* void **Bcm\_Init** (void)

*This function is responsible for initializing the mapping between the signals and PDUs.*

* uint8 **Bcm\_SendSignal** (**Bcm\_SignalIdType** SignalId, const void \*SignalDataPtr)

*The service Bcm\_SendSignal updates the signal object identified by SignalId with the signal referenced by the SignalDataPtr parameter.*

* uint8 **Bcm\_ReceiveSignal** (**Bcm\_SignalIdType** SignalId, void \*SignalDataPtr)

*Bcm\_ReceiveSignal copies the data of the signal identified by SignalId to the location specified by SignalDataPtr.*

* void **Bcm\_RxIndication** (**PduIdType** RxPduId, const **PduInfoType** \*PduInfoPtr)

*Indication of a received PDU from a lower layer communication interface module.*

#### Detailed Description

This module is responsible for handling basic communication and mapping signals to the right Pdu Ids. It abstracts the Pdu meta data e.g. PduId.

###### Author

Mohamed Hassanin (mohamed\_hassanin\_omran@yahoo.com)

###### Version

0.1

###### Date

2022-09-03

###### Copyright

Copyright (c) 2022

#### Function Documentation

##### void Bcm\_Init (void )

This function is responsible for initializing the mapping between the signals and PDUs.

##### uint8 Bcm\_ReceiveSignal (Bcm\_SignalIdType *SignalId*, void \* *SignalDataPtr*)

Bcm\_ReceiveSignal copies the data of the signal identified by SignalId to the location specified by SignalDataPtr.

###### Parameters

|  |  |
| --- | --- |
| *SignalId* | Id of signal to be received. |
| *SignalDataPtr* | Reference to the location where the received signal data shall bestored |

###### Returns

uint8

E\_OK: service has been accepted

E\_NOT\_OK: service has been rejected

##### void Bcm\_RxIndication (PduIdType *RxPduId*, const PduInfoType \* *PduInfoPtr*)

Indication of a received PDU from a lower layer communication interface module.

###### Parameters

|  |  |
| --- | --- |
| *RxPduId* | ID of the received PDU. |
| *PduInfoPtr* | Contains the length of the received PDU, a pointer to a buffer containing the PDU, and the MetaData related to this PDU. |

##### uint8 Bcm\_SendSignal (Bcm\_SignalIdType *SignalId*, const void \* *SignalDataPtr*)

The service Bcm\_SendSignal updates the signal object identified by SignalId with the signal referenced by the SignalDataPtr parameter.

###### Parameters

|  |  |
| --- | --- |
| *SignalId* | Id of signal to be sent. |
| *SignalDataPtr* | Reference to the signal data to be transmitted |

###### Returns

uint8

E\_OK: service has been accepted

E\_NOT\_OK: service has been rejected

### src/2-Service/Bcm/inc/Bcm\_Types.h File Reference

This module is responsible for handling basic communication and mapping signals to the right Pdu Ids. It abstracts the Pdu meta data e.g. PduId.

#### Typedefs

* typedef uint16 **Bcm\_SignalIdType**

*The signal id.*

#### Detailed Description

This module is responsible for handling basic communication and mapping signals to the right Pdu Ids. It abstracts the Pdu meta data e.g. PduId.

###### Author

Mohamed Hassanin (mohamed\_hassanin\_omran@yahoo.com)

###### Version

0.1

###### Date

2022-09-03

###### Copyright

Copyright (c) 2022

#### Typedef Documentation

##### typedef uint16 Bcm\_SignalIdType

The signal id.

### src/2-Service/Os/inc/Os.h File Reference

This module is responsible for handling the operating system and scheduling tasks.

#include "Os\_Types.h"

#### Functions

* void **Os\_Init** (void)

*Initialize the OS tasks.*

* void **Os\_StartScheduler** (void)

*Start the scheduler.*

#### Detailed Description

This module is responsible for handling the operating system and scheduling tasks.

###### Author

Mohamed Hassanin (mohamed\_hassanin\_omran@yahoo.com)

###### Version

0.1

###### Date

2022-09-03

###### Copyright

Copyright (c) 2022

#### Function Documentation

##### void Os\_Init (void )

Initialize the OS tasks.

##### void Os\_StartScheduler (void )

Start the scheduler.

### src/2-Service/Os/inc/Os\_Types.h File Reference

This module is responsible for handling the operating system and scheduling tasks.

#### Detailed Description

This module is responsible for handling the operating system and scheduling tasks.

###### Author

Mohamed Hassanin (mohamed\_hassanin\_omran@yahoo.com)

###### Version

0.1

###### Date

2022-09-03

###### Copyright

Copyright (c) 2022

### src/2-Service/PduR/inc/PduR.h File Reference

This module is responsible for routing and mapping Pdus to the right communication interfaces e.g. CanIf.

#include "PduR\_Types.h"

#include "ComStack\_Types.h"

#### Functions

* void **PduR\_Init** (void)

*Initializes the PDU Router.*

* Std\_ReturnType **PduR\_Transmit** (**PduIdType** TxPduId, const **PduInfoType** \*PduInfoPtr)

*Requests transmission of a PDU.*

* void **PduR\_RxIndication** (**PduIdType** RxPduId, const **PduInfoType** \*PduInfoPtr)

*Indication of a received PDU from a lower layer communication interface module.*

#### Detailed Description

This module is responsible for routing and mapping Pdus to the right communication interfaces e.g. CanIf.

###### Author

Mohamed Hassanin (mohamed\_hassanin\_omran@yahoo.com)

###### Version

0.1

###### Date

2022-09-03

###### Copyright

Copyright (c) 2022

#### Function Documentation

##### void PduR\_Init (void )

Initializes the PDU Router.

##### void PduR\_RxIndication (PduIdType *RxPduId*, const PduInfoType \* *PduInfoPtr*)

Indication of a received PDU from a lower layer communication interface module.

###### Parameters

|  |  |
| --- | --- |
| *RxPduId* | ID of the received PDU. |
| *PduInfoPtr* | Contains the length (SduLength) of the received PDU, a pointer to a buffer (SduDataPtr) containing the PDU, and the MetaData related to this PDU. |

##### Std\_ReturnType PduR\_Transmit (PduIdType *TxPduId*, const PduInfoType \* *PduInfoPtr*)

Requests transmission of a PDU.

###### Parameters

|  |  |
| --- | --- |
| *TxPduId* | Identifier of the PDU to be transmitted |
| *PduInfoPtr* | Length of and pointer to the PDU data and pointer to MetaData. |

###### Returns

Std\_ReturnType

### src/2-Service/PduR/inc/PduR\_Types.h File Reference

This module is responsible for routing and mapping Pdus to the right communication interfaces e.g. CanIf.

#### Detailed Description

This module is responsible for routing and mapping Pdus to the right communication interfaces e.g. CanIf.

###### Author

Mohamed Hassanin (mohamed\_hassanin\_omran@yahoo.com)

###### Version

0.1

###### Date

2022-09-03

###### Copyright

Copyright (c) 2022

### src/3-Ecu/CanIf/inc/CanIf.h File Reference

This module is responsible for abstracting the can controllers from the upper layers.

#include "CanIf\_Types.h"

#include "ComStack\_Types.h"

#include "Can\_GeneralTypes.h"

#### Functions

* void **CanIf\_Init** (void)

*Initialize CanIf module.*

* Std\_ReturnType **CanIf\_Transmit** (**PduIdType** TxPduId, const **PduInfoType** \*PduInfoPtr)

*Requests transmission of a PDU.*

* void **CanIf\_RxIndication** (const **Can\_HwType** \*Mailbox, const **PduInfoType** \*PduInfoPtr)

*This service indicates a successful reception of a received CAN Rx LPDU to the CanIf after passing all filters and validation checks.*

#### Detailed Description

This module is responsible for abstracting the can controllers from the upper layers.

###### Author

Mohamed Hassanin (mohamed\_hassanin\_omran@yahoo.com)

###### Version

0.1

###### Date

2022-09-03

###### Copyright

Copyright (c) 2022

#### Function Documentation

##### void CanIf\_Init (void )

Initialize CanIf module.

##### void CanIf\_RxIndication (const Can\_HwType \* *Mailbox*, const PduInfoType \* *PduInfoPtr*)

This service indicates a successful reception of a received CAN Rx LPDU to the CanIf after passing all filters and validation checks.

###### Parameters

|  |  |
| --- | --- |
| *Mailbox* | Identifies the HRH and its corresponding CAN Controller. |
| *PduInfoPtr* | Pointer to the received L-PDU |

##### Std\_ReturnType CanIf\_Transmit (PduIdType *TxPduId*, const PduInfoType \* *PduInfoPtr*)

Requests transmission of a PDU.

###### Parameters

|  |  |
| --- | --- |
| *TxPduId* | Identifier of the PDU to be transmitted |
| *PduInfoPtr* | Length of and pointer to the PDU data and pointer to MetaData. |

###### Returns

Std\_ReturnType

### src/3-Ecu/CanIf/inc/CanIf\_Types.h File Reference

This module is responsible for abstracting the can controllers from the upper layers.

#### Detailed Description

This module is responsible for abstracting the can controllers from the upper layers.

###### Author

Mohamed Hassanin (mohamed\_hassanin\_omran@yahoo.com)

###### Version

0.1

###### Date

2022-09-03

###### Copyright

Copyright (c) 2022

### src/3-Ecu/CanTrcv/inc/CanTrcv.h File Reference

This module is responsible for handling the can transceivers.

#include "CanTrcv\_Types.h"

#### Functions

* void **CanTrcv\_Init** (void)

*Initialize the Can transceiver module.*

* Std\_ReturnType **CanTrcv\_SetOpMode** (uint8 Transceiver, CanTrcv\_TrcvModeType OpMode)

*Sets the mode of the Transceiver to the value OpMode.*

#### Detailed Description

This module is responsible for handling the can transceivers.

###### Author

Mohamed Hassanin (mohamed\_hassanin\_omran@yahoo.com)

###### Version

0.1

###### Date

2022-09-03

###### Copyright

Copyright (c) 2022

#### Function Documentation

##### void CanTrcv\_Init (void )

Initialize the Can transceiver module.

##### Std\_ReturnType CanTrcv\_SetOpMode (uint8 *Transceiver*, CanTrcv\_TrcvModeType *OpMode*)

Sets the mode of the Transceiver to the value OpMode.

###### Parameters

|  |  |
| --- | --- |
| *Transceiver* | CAN transceiver to which API call has to be applied |
| *OpMode* | This parameter contains the desired operating mode |

###### Returns

Std\_ReturnType

### src/3-Ecu/CanTrcv/inc/CanTrcv\_Types.h File Reference

This module is responsible for handling the can transceivers.

#### Macros

* #define **CANTRCV\_TRCVMODE\_NORMAL**  0
* #define **CANTRCV\_TRCVMODE\_SLEEP**  1
* #define **CANTRCV\_TRCVMODE\_STANDBY**  2

#### Typedefs

* typedef uint8 **CanTrcv\_TrcvModeType**

#### Detailed Description

This module is responsible for handling the can transceivers.

###### Author

Mohamed Hassanin (mohamed\_hassanin\_omran@yahoo.com)

###### Version

0.1

###### Date

2022-09-03

###### Copyright

Copyright (c) 2022

### src/3-Ecu/Door/inc/Door.h File Reference

This module is responsible for setting the door sensor status (pressed or released).

#include "Door\_Types.h"

#### Functions

* void **Door\_Init** (void)

*Initialize the Door sensor.*

* void **Door\_Update** (void)

*This function is called periodically to update the door sensor status.*

* **Door\_StatusType** **Door\_GetStatus** (void)

*Get the door switch status.*

#### Detailed Description

This module is responsible for setting the door sensor status (pressed or released).

###### Author

Mohamed Hassanin (mohamed\_hassanin\_omran@yahoo.com)

###### Version

0.1

###### Date

2022-09-03

###### Copyright

Copyright (c) 2022

#### Function Documentation

##### Door\_StatusType Door\_GetStatus (void )

Get the door switch status.

###### Returns

Door\_StatusType This Door sensor status

##### void Door\_Init (void )

Initialize the Door sensor.

##### void Door\_Update (void )

This function is called periodically to update the door sensor status.

### src/3-Ecu/Door/inc/Door\_Types.h File Reference

This module is responsible for setting the door sensor status (pressed or released).

#### Macros

* #define **DOOR\_STATUS\_CLOSED**  0
* #define **DOOR\_STATUS\_OPEN**  1

#### Typedefs

* typedef uint8 **Door\_StatusType**

*The car door status (CLOSED, OPEN)*

#### Detailed Description

This module is responsible for setting the door sensor status (pressed or released).

###### Author

Mohamed Hassanin (mohamed\_hassanin\_omran@yahoo.com)

###### Version

0.1

###### Date

2022-09-03

###### Copyright

Copyright (c) 2022

#### Typedef Documentation

##### typedef uint8 Door\_StatusType

The car door status (CLOSED, OPEN)

### src/3-Ecu/Light/inc/Light.h File Reference

This module is responsible for setting the light switch status (pressed or released).

#include "Light\_Types.h"

#### Functions

* void **Light\_Init** (void)

*Initialize the Light switches.*

* void **Light\_Update** (void)

*This function is called periodically to update the light switch status.*

* **Light\_StatusType** **Light\_GetStatus** (void)

*Get the light switch status.*

#### Detailed Description

This module is responsible for setting the light switch status (pressed or released).

###### Author

Mohamed Hassanin (mohamed\_hassanin\_omran@yahoo.com)

###### Version

0.1

###### Date

2022-09-03

###### Copyright

Copyright (c) 2022

#### Function Documentation

##### Light\_StatusType Light\_GetStatus (void )

Get the light switch status.

###### Returns

Light\_StatusType This Light switch status

##### void Light\_Init (void )

Initialize the Light switches.

##### void Light\_Update (void )

This function is called periodically to update the light switch status.

### src/3-Ecu/Light/inc/Light\_Types.h File Reference

This module is responsible for setting the light switch status (pressed or released).

#### Macros

* #define **LIGHT\_STATUS\_PRESSED**  1
* #define **LIGHT\_STATUS\_RELEASED**  0

#### Typedefs

* typedef uint8 **Light\_StatusType**

*The status of the light switch (Pressed or Released).*

#### Detailed Description

This module is responsible for setting the light switch status (pressed or released).

###### Author

Mohamed Hassanin (mohamed\_hassanin\_omran@yahoo.com)

###### Version

0.1

###### Date

2022-09-03

###### Copyright

Copyright (c) 2022

#### Typedef Documentation

##### typedef uint8 Light\_StatusType

The status of the light switch (Pressed or Released).

### src/3-Ecu/Speedometer/inc/Speedometer.h File Reference

This module is responsible for calculating and setting the Speedometer sensor speed in kmph.

#include "Speedometer\_Types.h"

#### Functions

* void **Speedometer\_Init** (void)

*Initialize the speedometer module.*

* void **Speedometer\_Update** (void)

*This function is called periodically to update the speed.*

* Speed\_StatusType **Speedometer\_GetStatus** (void)

*Get the speed.*

#### Detailed Description

This module is responsible for calculating and setting the Speedometer sensor speed in kmph.

###### Author

Mohamed Hassanin (mohamed\_hassanin\_omran@yahoo.com)

###### Version

0.1

###### Date

2022-09-03

###### Copyright

Copyright (c) 2022

#### Function Documentation

##### Speed\_StatusType Speedometer\_GetStatus (void )

Get the speed.

###### Returns

Speed\_StatusType The speed.

##### void Speedometer\_Init (void )

Initialize the speedometer module.

##### void Speedometer\_Update (void )

This function is called periodically to update the speed.

### src/3-Ecu/Speedometer/inc/Speedometer\_Types.h File Reference

This module is responsible for calculating and setting the Speedometer sensor speed in kmph.

#### Typedefs

* typedef uint16 **Speed\_StatusType**

#### Detailed Description

This module is responsible for calculating and setting the Speedometer sensor speed in kmph.

###### Author

Mohamed Hassanin (mohamed\_hassanin\_omran@yahoo.com)

###### Version

0.1

###### Date

2022-09-03

###### Copyright

Copyright (c) 2022

### src/3-Ecu/Switch/inc/Switch.h File Reference

This module is responsible for handling switches state machines and Switches state.

#include "Switch\_Types.h"

#### Functions

* void **Switch\_Init** (void)

*Initialize the switches state.*

* void **Switch\_Update** (void)

*This function is called periodically to update the switches state machines.*

* **Switch\_StateType** **Switch\_GetState** (**Switch\_IdType** SwitchId)

*Get the switch state.*

#### Detailed Description

This module is responsible for handling switches state machines and Switches state.

###### Author

Mohamed Hassanin (mohamed\_hassanin\_omran@yahoo.com)

###### Version

0.1

###### Date

2022-09-03

###### Copyright

Copyright (c) 2022

#### Function Documentation

##### Switch\_StateType Switch\_GetState (Switch\_IdType *SwitchId*)

Get the switch state.

###### Returns

Switch\_StateType The switch state.

##### void Switch\_Init (void )

Initialize the switches state.

##### void Switch\_Update (void )

This function is called periodically to update the switches state machines.

### src/3-Ecu/Switch/inc/Switch\_Types.h File Reference

This module is responsible for handling switches state machines and Switches state.

#### Macros

* #define **SWITCH\_STATE\_PRESSED**  0
* #define **SWITCH\_STATE\_PREPRESSED**  1
* #define **SWITCH\_STATE\_RELEASED**  2
* #define **SWITCH\_STATE\_PRERELEASED**  3
* #define **SWITCH\_ID\_LIGHT**  0
* #define **SWITCH\_ID\_DOOR**  1

#### Typedefs

* typedef uint8 **Switch\_StateType**

*The state of the switch (PRESSED, PREPRESSED, RELEASED, PRERELEASED)*

* typedef uint8 **Switch\_IdType**

*The switch.*

#### Detailed Description

This module is responsible for handling switches state machines and Switches state.

###### Author

Mohamed Hassanin (mohamed\_hassanin\_omran@yahoo.com)

###### Version

0.1

###### Date

2022-09-03

###### Copyright

Copyright (c) 2022

#### Typedef Documentation

##### typedef uint8 Switch\_IdType

The switch.

##### typedef uint8 Switch\_StateType

The state of the switch (PRESSED, PREPRESSED, RELEASED, PRERELEASED)

### src/4-Mcal/Can/inc/Can.h File Reference

CAN communication protocol driver.

#include "Can\_Types.h"

#include "Can\_GeneralTypes.h"

#### Functions

* void **Can\_Init** (void)

*Initialize the CAN driver.*

* Std\_ReturnType **Can\_Write** (**Can\_HwHandleType** Hth, const **Can\_PduType** \*PduInfo)

*This function is called by CanIf to pass a CAN message to CanDrv for transmission.*

#### Detailed Description

CAN communication protocol driver.

###### Author

Mohamed Hassanin (mohamed\_hassanin\_omran@yahoo.com)

###### Version

0.1

###### Date

2022-09-03

###### Copyright

Copyright (c) 2022

#### Function Documentation

##### void Can\_Init (void )

Initialize the CAN driver.

##### Std\_ReturnType Can\_Write (Can\_HwHandleType *Hth*, const Can\_PduType \* *PduInfo*)

This function is called by CanIf to pass a CAN message to CanDrv for transmission.

###### Parameters

|  |  |
| --- | --- |
| *Hth* | information which HW-transmit handle shall be used for transmit. Implicitly this is also the information about the controller to use because the Hth numbers are unique inside one hardware unit. |
| *PduInfo* | Pointer to SDU user memory, Data Length and Identifier. |

###### Returns

Std\_ReturnType

### src/4-Mcal/Can/inc/Can\_Types.h File Reference

CAN communication protocol driver.

#### Detailed Description

CAN communication protocol driver.

###### Author

Mohamed Hassanin (mohamed\_hassanin\_omran@yahoo.com)

###### Version

0.1

###### Date

2022-09-03

###### Copyright

Copyright (c) 2022

### src/4-Mcal/Dio/inc/Dio.h File Reference

Digital input/output driver.

#include "Dio\_Types.h"

#### Functions

* **Dio\_LevelType** **Dio\_ReadChannel** (**Dio\_ChannelType** ChannelId)

#### Detailed Description

Digital input/output driver.

###### Author

Mohamed Hassanin (mohamed\_hassanin\_omran@yahoo.com)

###### Version

0.1

###### Date

2022-09-03

###### Copyright

Copyright (c) 2022

#### Function Documentation

##### Dio\_LevelType Dio\_ReadChannel (Dio\_ChannelType *ChannelId*)

###### Parameters

|  |  |
| --- | --- |
| *ChannelId* |  |

###### Returns

Dio\_LevelType

### src/4-Mcal/Dio/inc/Dio\_Types.h File Reference

Digital input/output driver.

#### Macros

* #define **STD\_LOW**  0
* #define **STD\_HIGH**  1

#### Typedefs

* typedef uint8 **Dio\_LevelType**

*These are the possible levels a DIO channel can have (input or output):*

*STD\_LOW*

*STD\_HIGH*

*.*

* typedef uint8 **Dio\_ChannelType**

*Numeric ID of a DIO channel.*

#### Detailed Description

Digital input/output driver.

###### Author

Mohamed Hassanin (mohamed\_hassanin\_omran@yahoo.com)

###### Version

0.1

###### Date

2022-09-03

###### Copyright

Copyright (c) 2022

#### Typedef Documentation

##### typedef uint8 Dio\_ChannelType

Numeric ID of a DIO channel.

##### typedef uint8 Dio\_LevelType

These are the possible levels a DIO channel can have (input or output):

STD\_LOW

STD\_HIGH

.

### src/4-Mcal/Gpt/inc/Gpt.h File Reference

General purpose timer driver.

#include "Gpt\_Types.h"

#### Functions

* void **Gpt\_Init** (void)

*Initialize the general purpose timer.*

* void **Gpt\_StartTimer** (**Gpt\_ChannelType** Channel, **Gpt\_ValueType** Value)

*Starts a timer channel.*

* void **Gpt\_EnableNotification** (**Gpt\_ChannelType** Channel)

*Enables the interrupt notification for a channel (relevant in normal mode).*

#### Detailed Description

General purpose timer driver.

###### Author

Mohamed Hassanin (mohamed\_hassanin\_omran@yahoo.com)

###### Version

0.1

###### Date

2022-09-03

###### Copyright

Copyright (c) 2022

#### Function Documentation

##### void Gpt\_EnableNotification (Gpt\_ChannelType *Channel*)

Enables the interrupt notification for a channel (relevant in normal mode).

###### Parameters

|  |  |
| --- | --- |
| *Channel* | Numeric identifier of the GPT channel |

##### void Gpt\_Init (void )

Initialize the general purpose timer.

##### void Gpt\_StartTimer (Gpt\_ChannelType *Channel*, Gpt\_ValueType *Value*)

Starts a timer channel.

###### Parameters

|  |  |
| --- | --- |
| *Channel* | Numeric identifier of the GPT channel. |
| *Value* | Target time in number of ticks |

### src/4-Mcal/Gpt/inc/Gpt\_Types.h File Reference

General purpose timer driver.

#### Typedefs

* typedef uint8 **Gpt\_ChannelType**

*Numeric ID of a GPT channel.*

* typedef uint8 **Gpt\_ValueType**

*Type for reading and setting the timer values (in number of ticks).*

#### Detailed Description

General purpose timer driver.

###### Author

Mohamed Hassanin (mohamed\_hassanin\_omran@yahoo.com)

###### Version

0.1

###### Date

2022-09-03

###### Copyright

Copyright (c) 2022

#### Typedef Documentation

##### typedef uint8 Gpt\_ChannelType

Numeric ID of a GPT channel.

##### typedef uint8 Gpt\_ValueType

Type for reading and setting the timer values (in number of ticks).

### src/4-Mcal/Icu/inc/Icu.h File Reference

Input capture unit driver.

#include "Icu\_Types.h"

#### Functions

* void **Icu\_Init** (void)

*Initialize the Input Capture Unit (ICU).*

* **Icu\_ValueType** **Icu\_GetTimeElapsed** (**Icu\_ChannelType** Channel)

*This function reads the elapsed Time for the given channel.*

#### Detailed Description

Input capture unit driver.

###### Author

Mohamed Hassanin (mohamed\_hassanin\_omran@yahoo.com)

###### Version

0.1

###### Date

2022-09-03

###### Copyright

Copyright (c) 2022

#### Function Documentation

##### Icu\_ValueType Icu\_GetTimeElapsed (Icu\_ChannelType *Channel*)

This function reads the elapsed Time for the given channel.

###### Parameters

|  |  |
| --- | --- |
| *Channel* | Numeric identifier of the ICU channel |

###### Returns

Icu\_ValueType

##### void Icu\_Init (void )

Initialize the Input Capture Unit (ICU).

### src/4-Mcal/Icu/inc/Icu\_Types.h File Reference

Input capture unit driver.

#### Typedefs

* typedef uint8 **Icu\_ChannelType**

*Numeric identifier of the ICU channel.*

* typedef uint8 **Icu\_ValueType**

*Width of the buffer for timestamp ticks and measured elapsed timeticks.*

#### Detailed Description

Input capture unit driver.

###### Author

Mohamed Hassanin (mohamed\_hassanin\_omran@yahoo.com)

###### Version

0.1

###### Date

2022-09-03

###### Copyright

Copyright (c) 2022

#### Typedef Documentation

##### typedef uint8 Icu\_ChannelType

Numeric identifier of the ICU channel.

##### typedef uint8 Icu\_ValueType

Width of the buffer for timestamp ticks and measured elapsed timeticks.

### src/4-Mcal/Port/inc/Port.h File Reference

Port driver.

#include "Port\_Types.h"

#### Functions

* void **Port\_Init** (void)

*Initialize the Port module.*

* void **Port\_SetPinDirection** (**Port\_PinType** Pin, **Port\_PinDirectionType** Direction)

*Sets the port pin direction.*

* void **Port\_SetPinMode** (**Port\_PinType** Pin, **Port\_PinModeType** Mode)

*Sets the port pin mode.*

#### Detailed Description

Port driver.

###### Author

Mohamed Hassanin (mohamed\_hassanin\_omran@yahoo.com)

###### Version

0.1

###### Date

2022-09-03

###### Copyright

Copyright (c) 2022

#### Function Documentation

##### void Port\_Init (void )

Initialize the Port module.

##### void Port\_SetPinDirection (Port\_PinType *Pin*, Port\_PinDirectionType *Direction*)

Sets the port pin direction.

###### Parameters

|  |  |
| --- | --- |
| *Pin* | Port Pin ID number |
| *Direction* | Port Pin Direction |

##### void Port\_SetPinMode (Port\_PinType *Pin*, Port\_PinModeType *Mode*)

Sets the port pin mode.

###### Parameters

|  |  |
| --- | --- |
| *Pin* | Port Pin ID number |
| *Mode* | New Port Pin mode to be set on port pin. |

### src/4-Mcal/Port/inc/Port\_Types.h File Reference

Port driver.

#### Typedefs

* typedef uint8 **Port\_PinDirectionType**

*Possible directions of a port pin.:*

*PORT\_PIN\_IN*

*PORT\_PIN\_OUT*

*.*

* typedef uint8 **Port\_PinType**

*Data type for the symbolic name of a port pin.*

* typedef uint8 **Port\_PinModeType**

*Different port pin modes.*

#### Detailed Description

Port driver.

###### Author

Mohamed Hassanin (mohamed\_hassanin\_omran@yahoo.com)

###### Version

0.1

###### Date

2022-09-03

###### Copyright

Copyright (c) 2022

#### Typedef Documentation

##### typedef uint8 Port\_PinDirectionType

Possible directions of a port pin.:

PORT\_PIN\_IN

PORT\_PIN\_OUT

.

##### typedef uint8 Port\_PinModeType

Different port pin modes.

##### typedef uint8 Port\_PinType

Data type for the symbolic name of a port pin.

### src/4-Mcal/Spi/inc/Spi.h File Reference

SPI communication protocol driver.

#include "Spi\_Types.h"

#### Functions

* void **Spi\_Init** (void)

*Initialize the SPI driver.*

* Std\_ReturnType **Spi\_Write** (**Spi\_ChannelType** Channel, const **Spi\_DataBufferType** \*DataBufferPtr)

*Service for writing synchronously one or more data from a SPI Handler/Driver Channel specified by parameter.*

* Std\_ReturnType **Spi\_Read** (**Spi\_ChannelType** Channel, **Spi\_DataBufferType** \*DataBufferPointer)

*Service for reading synchronously one or more data from a SPI Handler/Driver Channel specified by parameter.*

#### Detailed Description

SPI communication protocol driver.

###### Author

Mohamed Hassanin (mohamed\_hassanin\_omran@yahoo.com)

###### Version

0.1

###### Date

2022-09-03

###### Copyright

Copyright (c) 2022

#### Function Documentation

##### void Spi\_Init (void )

Initialize the SPI driver.

##### Std\_ReturnType Spi\_Read (Spi\_ChannelType *Channel*, Spi\_DataBufferType \* *DataBufferPointer*)

Service for reading synchronously one or more data from a SPI Handler/Driver Channel specified by parameter.

###### Parameters

|  |  |
| --- | --- |
| *Channel* | Channel ID. |
| *DataBufferPointer* | Pointer to destination data buffer in RAM |

###### Returns

Std\_ReturnType

##### Std\_ReturnType Spi\_Write (Spi\_ChannelType *Channel*, const Spi\_DataBufferType \* *DataBufferPtr*)

Service for writing synchronously one or more data from a SPI Handler/Driver Channel specified by parameter.

###### Parameters

|  |  |
| --- | --- |
| *Channel* | Channel ID. |
| *DataBufferPtr* | Pointer to source data buffer in RAM |

###### Returns

Std\_ReturnType

### src/4-Mcal/Spi/inc/Spi\_Types.h File Reference

SPI communication protocol driver.

#### Typedefs

* typedef uint8 **Spi\_ChannelType**

*Specifies the identification (ID) for a Channel.*

* typedef uint8 **Spi\_DataBufferType**

*Type of application data buffer elements.*

#### Detailed Description

SPI communication protocol driver.

###### Author

Mohamed Hassanin (mohamed\_hassanin\_omran@yahoo.com)

###### Version

0.1

###### Date

2022-09-03

###### Copyright

Copyright (c) 2022

#### Typedef Documentation

##### typedef uint8 Spi\_ChannelType

Specifies the identification (ID) for a Channel.

##### typedef uint8 Spi\_DataBufferType

Type of application data buffer elements.

### src/5-Common/inc/Can\_GeneralTypes.h File Reference

This file contains the CAN communication CAN\_GENERAL\_TYPES types definitions.

#include "Std\_Types.h"

#include "ComStack\_Types.h"

#### Data Structures

* struct **Can\_PduType**

*This type unites PduId (swPduHandle), SduLength (length), SduData (sdu), and CanId (id) for any CAN L-SDU.*

* struct **Can\_HwType**

*This type defines a data structure which clearly provides an Hardware Object Handle including its corresponding CAN Controller and therefore CanDrv as well as the specific CanId..*

#### Typedefs

* typedef struct **Can\_PduType** **Can\_PduType**

*This type unites PduId (swPduHandle), SduLength (length), SduData (sdu), and CanId (id) for any CAN L-SDU.*

* typedef uint32 **Can\_IdType**

*Represents the Identifier of an L-PDU. The two most significant bits specify the frame type:*

*00 CAN message with Standard CAN ID*

*01 CAN FD frame with Standard CAN ID*

*10 CAN message with Extended CAN ID*

*11 CAN FD frame with Extended CAN ID*

*.*

* typedef uint16 **Can\_HwHandleType**

*Represents the hardware object handles of a CAN hardware unit. For CAN hardware units with more than 255 HW objects use extended range. Ranges:*

*standard: 0..0x0FF*

*Extended: 0..0xFFFF*

*.*

* typedef struct **Can\_HwType** **Can\_HwType**

*This type defines a data structure which clearly provides an Hardware Object Handle including its corresponding CAN Controller and therefore CanDrv as well as the specific CanId..*

#### Detailed Description

This file contains the CAN communication CAN\_GENERAL\_TYPES types definitions.

###### Author

Mohamed Hassanin (mohamed\_hassanin\_omran@yahoo.com)

###### Version

0.1

###### Date

2022-09-03

###### Copyright

Copyright (c) 2022

#### Typedef Documentation

##### typedef uint16 Can\_HwHandleType

Represents the hardware object handles of a CAN hardware unit. For CAN hardware units with more than 255 HW objects use extended range. Ranges:

standard: 0..0x0FF

Extended: 0..0xFFFF

.

##### typedef struct Can\_HwType Can\_HwType

This type defines a data structure which clearly provides an Hardware Object Handle including its corresponding CAN Controller and therefore CanDrv as well as the specific CanId..

##### typedef struct Can\_PduType Can\_PduType

This type unites PduId (swPduHandle), SduLength (length), SduData (sdu), and CanId (id) for any CAN L-SDU.

### src/5-Common/inc/ComStack\_Types.h File Reference

This file contains the communication COM\_STACK\_TYPES types definitions.

#include "Std\_Types.h"

#### Data Structures

* struct **PduInfoType**

*The pdu information contains the SDU length and the payload.*

#### Typedefs

* typedef uint16 **PduIdType**

*The pdu id.*

* typedef uint16 **PduLengthType**
* typedef struct **PduInfoType** **PduInfoType**

*The pdu information contains the SDU length and the payload.*

#### Detailed Description

This file contains the communication COM\_STACK\_TYPES types definitions.

###### Author

Mohamed Hassanin (mohamed\_hassanin\_omran@yahoo.com)

###### Version

0.1

###### Date

2022-09-03

###### Copyright

Copyright (c) 2022

#### Typedef Documentation

##### typedef uint16 PduIdType

The pdu id.

##### typedef struct PduInfoType PduInfoType

The pdu information contains the SDU length and the payload.

## Prepare your folder structure according to the previous points

I ran tree command to print my folder structure and this is the output:

+---1-Appl

| +---DoorStatus

| | | DoorStatus.c

| | |

| | \---inc

| | DoorStatus.h

| | DoorStatus\_Types.h

| |

| +---LightStatus

| | | LightStatus.c

| | |

| | \---inc

| | LightStatus.h

| | LightStatus\_Types.h

| |

| \---SpeedStatus

| | SpeedStatus.c

| |

| \---inc

| SpeedStatus.h

| SpeedStatus\_Types.h

|

+---2-Service

| +---Bcm

| | | Bcm.c

| | |

| | \---inc

| | Bcm.h

| | Bcm\_Types.h

| |

| +---Os

| | | Os.c

| | |

| | \---inc

| | Os.h

| | Os\_Types.h

| |

| \---PduR

| | PduR.c

| |

| \---inc

| PduR.h

| PduR\_Types.h

|

+---3-Ecu

| +---CanIf

| | | CanIf.c

| | |

| | \---inc

| | CanIf.h

| | CanIf\_Types.h

| |

| +---CanTrcv

| | | CanTrcv.c

| | |

| | \---inc

| | CanTrcv.h

| | CanTrcv\_Types.h

| |

| +---Door

| | | Door.c

| | |

| | \---inc

| | Door.h

| | Door\_Types.h

| |

| +---Light

| | | Light.c

| | |

| | \---inc

| | Light.h

| | Light\_Types.h

| |

| +---Speedometer

| | | Speedometer.c

| | |

| | \---inc

| | Speedometer.h

| | Speedometer\_Types.h

| |

| \---Switch

| | Switch.c

| |

| \---inc

| Switch.h

| Switch\_Types.h

|

+---4-Mcal

| +---Can

| | | Can.c

| | |

| | \---inc

| | Can.h

| | Can\_Types.h

| |

| +---Dio

| | | Dio.c

| | |

| | \---inc

| | Dio.h

| | Dio\_Types.h

| |

| +---Gpt

| | | Gpt.c

| | |

| | \---inc

| | Gpt.h

| | Gpt\_Types.h

| |

| +---Icu

| | | Icu.c

| | |

| | \---inc

| | Icu.h

| | Icu\_Types.h

| |

| +---Port

| | | Port.c

| | |

| | \---inc

| | Port.h

| | Port\_Types.h

| |

| \---Spi

| | Spi.c

| |

| \---inc

| Spi.h

| Spi\_Types.h

|

\---5-Common

+---inc

| Can\_GeneralTypes.h

| Compiler.h

| ComStack\_Types.h

| Mcu\_Hw.h

| Platform\_Types.h

| Std\_Types.h

|

\---src

# ECU2 Static Design

## Make the layered architecture

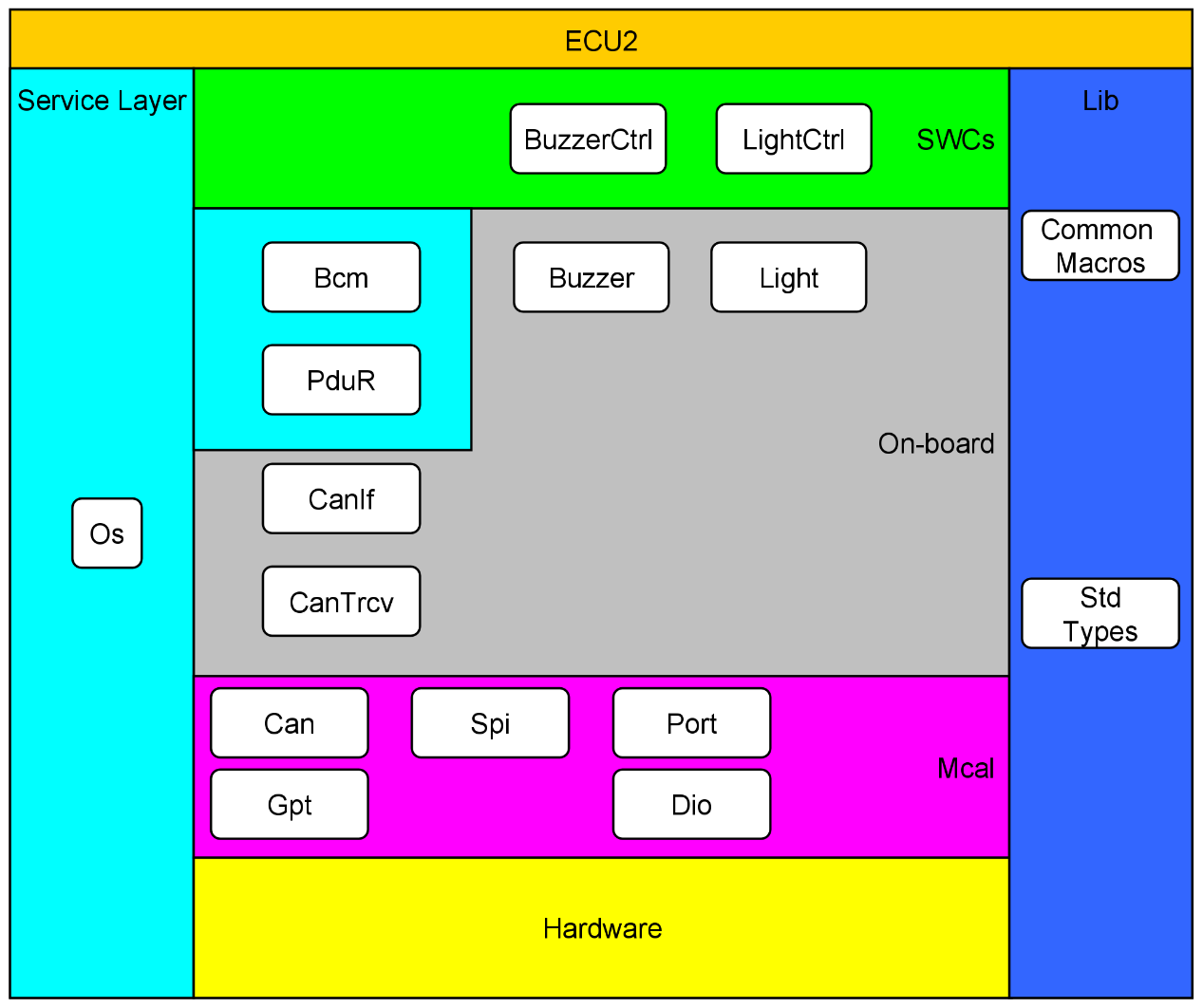


Figure 2 ECU2 static design

## Specify ECU components and modules

### Application layer

**BuzzerCtrl**

This software component is responsible of reading the Can signals and control the buzzer accordingly.

**LightCtrl**

This software component is responsible of reading the Can signals and control the light accordingly.

### Service Layer

**Bcm**

This module is responsible of transmitting/receiving the CAN signals to/from the SWCs. It abstracts the Pdu Information from the application layer.

**PduR**

This module is responsible for routing the PDUs to the right communication interface e.g. (CanIf). It abstracts the ECU layer communication protocols from the upper layer.

### Ecu Layer (on board)

**Light**

This module is responsible to abstract the port and pin of the light from the application layer.

**Buzzer**

This module is responsible to abstract the port and pin of the buzzer from the application layer.

**CanIf**

This module is responsible for abstracting the different CAN controllers from the upper layer.

**CanTrcv**

This module is responsible for abstracting the manipulation of the CAN transceiver configuration from the upper layer.

### Mcal Layer

**Port**

This driver is responsible for the configuration of GPIOs.

**Dio**

This driver is responsible for reading or writing from/to GPIO pins.

**Can**

This is the can driver that deal with the mailboxes and CAN controllers’ configuration.

**Spi**

This is the spi driver that’s used by CanTrcv module to configure the CAN transceivers.

**Gpt**

This is the general-purpose timer driver. It’s used by the scheduler as a tick source.

**Icu**

This is the driver of the Input Capture Unit which is used to interface with the speed sensor.

## Provide full detailed APIs for each module as well as a detailed description for the used typedefs

**Note: the following documentation is generated using DoxyGen**

### src/1-Appl/BuzzerCtrl/inc/BuzzerCtrl.h File Reference

This module is responsible for controlling the buzzer.

#include "BuzzerCtrl\_Types.h"

#### Functions

* void **BuzzerCtrl\_Init** (void)

*Initialize the Buzzer control module.*

* void **BuzzerCtrl\_Update** (void)

*This function runs periodically to control the buzzer.*

#### Detailed Description

This module is responsible for controlling the buzzer.

###### Author

Mohamed Hassanin (mohamed\_hassanin\_omran@yahoo.com)

###### Version

0.1

###### Date

2022-09-03

###### Copyright

Copyright (c) 2022

#### Function Documentation

##### void BuzzerCtrl\_Init (void )

Initialize the Buzzer control module.

##### void BuzzerCtrl\_Update (void )

This function runs periodically to control the buzzer.

### src/1-Appl/BuzzerCtrl/inc/BuzzerCtrl\_Types.h File Reference

This module is responsible for controlling the buzzer.

#### Detailed Description

This module is responsible for controlling the buzzer.

###### Author

Mohamed Hassanin (mohamed\_hassanin\_omran@yahoo.com)

###### Version

0.1

###### Date

2022-09-03

###### Copyright

Copyright (c) 2022

### src/1-Appl/LightCtrl/inc/LightCtrl.h File Reference

This module is responsible for controlling the light.

#include "LightCtrl\_Types.h"

#### Detailed Description

This module is responsible for controlling the light.

###### Author

Mohamed Hassanin (mohamed\_hassanin\_omran@yahoo.com)

###### Version

0.1

###### Date

2022-09-03

###### Copyright

Copyright (c) 2022

### src/1-Appl/LightCtrl/inc/LightCtrl\_Types.h File Reference

This module is responsible for controlling the light.

#### Detailed Description

This module is responsible for controlling the light.

###### Author

Mohamed Hassanin (mohamed\_hassanin\_omran@yahoo.com)

###### Version

0.1

###### Date

2022-09-03

###### Copyright

Copyright (c) 2022

### src/2-Service/Bcm/inc/Bcm.h File Reference

This module is responsible for handling basic communication and mapping signals to the right Pdu Ids. It abstracts the Pdu meta data e.g. PduId.

#include "Bcm\_Types.h"

#include "ComStack\_Types.h"

#### Functions

* void **Bcm\_Init** (void)

*This function is responsible for initializing the mapping between the signals and PDUs.*

* uint8 **Bcm\_SendSignal** (**Bcm\_SignalIdType** SignalId, const void \*SignalDataPtr)

*The service Bcm\_SendSignal updates the signal object identified by SignalId with the signal referenced by the SignalDataPtr parameter.*

* uint8 **Bcm\_ReceiveSignal** (**Bcm\_SignalIdType** SignalId, void \*SignalDataPtr)

*Bcm\_ReceiveSignal copies the data of the signal identified by SignalId to the location specified by SignalDataPtr.*

* void **Bcm\_RxIndication** (**PduIdType** RxPduId, const **PduInfoType** \*PduInfoPtr)

*Indication of a received PDU from a lower layer communication interface module.*

#### Detailed Description

This module is responsible for handling basic communication and mapping signals to the right Pdu Ids. It abstracts the Pdu meta data e.g. PduId.

###### Author

Mohamed Hassanin (mohamed\_hassanin\_omran@yahoo.com)

###### Version

0.1

###### Date

2022-09-03

###### Copyright

Copyright (c) 2022

#### Function Documentation

##### void Bcm\_Init (void )

This function is responsible for initializing the mapping between the signals and PDUs.

##### uint8 Bcm\_ReceiveSignal (Bcm\_SignalIdType *SignalId*, void \* *SignalDataPtr*)

Bcm\_ReceiveSignal copies the data of the signal identified by SignalId to the location specified by SignalDataPtr.

###### Parameters

|  |  |
| --- | --- |
| *SignalId* | Id of signal to be received. |
| *SignalDataPtr* | Reference to the location where the received signal data shall bestored |

###### Returns

uint8

E\_OK: service has been accepted

E\_NOT\_OK: service has been rejected

##### void Bcm\_RxIndication (PduIdType *RxPduId*, const PduInfoType \* *PduInfoPtr*)

Indication of a received PDU from a lower layer communication interface module.

###### Parameters

|  |  |
| --- | --- |
| *RxPduId* | ID of the received PDU. |
| *PduInfoPtr* | Contains the length of the received PDU, a pointer to a buffer containing the PDU, and the MetaData related to this PDU. |

##### uint8 Bcm\_SendSignal (Bcm\_SignalIdType *SignalId*, const void \* *SignalDataPtr*)

The service Bcm\_SendSignal updates the signal object identified by SignalId with the signal referenced by the SignalDataPtr parameter.

###### Parameters

|  |  |
| --- | --- |
| *SignalId* | Id of signal to be sent. |
| *SignalDataPtr* | Reference to the signal data to be transmitted |

###### Returns

uint8

E\_OK: service has been accepted

E\_NOT\_OK: service has been rejected

### src/2-Service/Bcm/inc/Bcm\_Types.h File Reference

This module is responsible for handling basic communication and mapping signals to the right Pdu Ids. It abstracts the Pdu meta data e.g. PduId.

#### Typedefs

* typedef uint16 **Bcm\_SignalIdType**

*The signal id.*

#### Detailed Description

This module is responsible for handling basic communication and mapping signals to the right Pdu Ids. It abstracts the Pdu meta data e.g. PduId.

###### Author

Mohamed Hassanin (mohamed\_hassanin\_omran@yahoo.com)

###### Version

0.1

###### Date

2022-09-03

###### Copyright

Copyright (c) 2022

#### Typedef Documentation

##### typedef uint16 Bcm\_SignalIdType

The signal id.

### src/2-Service/Os/inc/Os.h File Reference

This module is responsible for handling the operating system and scheduling tasks.

#include "Os\_Types.h"

#### Functions

* void **Os\_Init** (void)

*Initialize the OS tasks.*

* void **Os\_StartScheduler** (void)

*Start the scheduler.*

#### Detailed Description

This module is responsible for handling the operating system and scheduling tasks.

###### Author

Mohamed Hassanin (mohamed\_hassanin\_omran@yahoo.com)

###### Version

0.1

###### Date

2022-09-03

###### Copyright

Copyright (c) 2022

#### Function Documentation

##### void Os\_Init (void )

Initialize the OS tasks.

##### void Os\_StartScheduler (void )

Start the scheduler.

### src/2-Service/Os/inc/Os\_Types.h File Reference

This module is responsible for handling the operating system and scheduling tasks.

#### Detailed Description

This module is responsible for handling the operating system and scheduling tasks.

###### Author

Mohamed Hassanin (mohamed\_hassanin\_omran@yahoo.com)

###### Version

0.1

###### Date

2022-09-03

###### Copyright

Copyright (c) 2022

### src/2-Service/PduR/inc/PduR.h File Reference

This module is responsible for routing and mapping Pdus to the right communication interfaces e.g. CanIf.

#include "PduR\_Types.h"

#include "ComStack\_Types.h"

#### Functions

* void **PduR\_Init** (void)

*Initializes the PDU Router.*

* Std\_ReturnType **PduR\_Transmit** (**PduIdType** TxPduId, const **PduInfoType** \*PduInfoPtr)

*Requests transmission of a PDU.*

* void **PduR\_RxIndication** (**PduIdType** RxPduId, const **PduInfoType** \*PduInfoPtr)

*Indication of a received PDU from a lower layer communication interface module.*

#### Detailed Description

This module is responsible for routing and mapping Pdus to the right communication interfaces e.g. CanIf.

###### Author

Mohamed Hassanin (mohamed\_hassanin\_omran@yahoo.com)

###### Version

0.1

###### Date

2022-09-03

###### Copyright

Copyright (c) 2022

#### Function Documentation

##### void PduR\_Init (void )

Initializes the PDU Router.

##### void PduR\_RxIndication (PduIdType *RxPduId*, const PduInfoType \* *PduInfoPtr*)

Indication of a received PDU from a lower layer communication interface module.

###### Parameters

|  |  |
| --- | --- |
| *RxPduId* | ID of the received PDU. |
| *PduInfoPtr* | Contains the length (SduLength) of the received PDU, a pointer to a buffer (SduDataPtr) containing the PDU, and the MetaData related to this PDU. |

##### Std\_ReturnType PduR\_Transmit (PduIdType *TxPduId*, const PduInfoType \* *PduInfoPtr*)

Requests transmission of a PDU.

###### Parameters

|  |  |
| --- | --- |
| *TxPduId* | Identifier of the PDU to be transmitted |
| *PduInfoPtr* | Length of and pointer to the PDU data and pointer to MetaData. |

###### Returns

Std\_ReturnType

### src/2-Service/PduR/inc/PduR\_Types.h File Reference

This module is responsible for routing and mapping Pdus to the right communication interfaces e.g. CanIf.

#### Detailed Description

This module is responsible for routing and mapping Pdus to the right communication interfaces e.g. CanIf.

###### Author

Mohamed Hassanin (mohamed\_hassanin\_omran@yahoo.com)

###### Version

0.1

###### Date

2022-09-03

###### Copyright

Copyright (c) 2022

### src/3-Ecu/Buzzer/inc/Buzzer.h File Reference

This module is responsible for abstracting the dio of the buzzer.

#include "Buzzer\_Types.h"

#### Functions

* void **Buzzer\_Init** (void)

*Initialize the buzzer module.*

* void **Buzzer\_SetStatus** (**Buzzer\_StatusType** Status)

*Set the buzzer state (BUZZER\_STATUS\_OFF, BUZZER\_STATUS\_ON)*

#### Detailed Description

This module is responsible for abstracting the dio of the buzzer.

###### Author

Mohamed Hassanin (mohamed\_hassanin\_omran@yahoo.com)

###### Version

0.1

###### Date

2022-09-03

###### Copyright

Copyright (c) 2022

#### Function Documentation

##### void Buzzer\_Init (void )

Initialize the buzzer module.

##### void Buzzer\_SetStatus (Buzzer\_StatusType *Status*)

Set the buzzer state (BUZZER\_STATUS\_OFF, BUZZER\_STATUS\_ON)

###### Parameters

|  |  |
| --- | --- |
| *Status* |  |

### src/3-Ecu/Buzzer/inc/Buzzer\_Types.h File Reference

This module is responsible for abstracting the dio of the buzzer.

#### Macros

* #define **BUZZER\_STATUS\_OFF**  0
* #define **BUZZER\_STATUS\_ON**  1

#### Typedefs

* typedef uint8 **Buzzer\_StatusType**

*Buzzer status BUZZER\_STATUS\_OFF, BUZZER\_STATUS\_ON.*

#### Detailed Description

This module is responsible for abstracting the dio of the buzzer.

###### Author

Mohamed Hassanin (mohamed\_hassanin\_omran@yahoo.com)

###### Version

0.1

###### Date

2022-09-03

###### Copyright

Copyright (c) 2022

#### Typedef Documentation

##### typedef uint8 Buzzer\_StatusType

Buzzer status BUZZER\_STATUS\_OFF, BUZZER\_STATUS\_ON.

### src/3-Ecu/CanIf/inc/CanIf.h File Reference

This module is responsible for abstracting the can controllers from the upper layers.

#include "CanIf\_Types.h"

#include "ComStack\_Types.h"

#include "Can\_GeneralTypes.h"

#### Functions

* void **CanIf\_Init** (void)

*Initialize CanIf module.*

* Std\_ReturnType **CanIf\_Transmit** (**PduIdType** TxPduId, const **PduInfoType** \*PduInfoPtr)

*Requests transmission of a PDU.*

* void **CanIf\_RxIndication** (const **Can\_HwType** \*Mailbox, const **PduInfoType** \*PduInfoPtr)

*This service indicates a successful reception of a received CAN Rx LPDU to the CanIf after passing all filters and validation checks.*

#### Detailed Description

This module is responsible for abstracting the can controllers from the upper layers.

###### Author

Mohamed Hassanin (mohamed\_hassanin\_omran@yahoo.com)

###### Version

0.1

###### Date

2022-09-03

###### Copyright

Copyright (c) 2022

#### Function Documentation

##### void CanIf\_Init (void )

Initialize CanIf module.

##### void CanIf\_RxIndication (const Can\_HwType \* *Mailbox*, const PduInfoType \* *PduInfoPtr*)

This service indicates a successful reception of a received CAN Rx LPDU to the CanIf after passing all filters and validation checks.

###### Parameters

|  |  |
| --- | --- |
| *Mailbox* | Identifies the HRH and its corresponding CAN Controller. |
| *PduInfoPtr* | Pointer to the received L-PDU |

##### Std\_ReturnType CanIf\_Transmit (PduIdType *TxPduId*, const PduInfoType \* *PduInfoPtr*)

Requests transmission of a PDU.

###### Parameters

|  |  |
| --- | --- |
| *TxPduId* | Identifier of the PDU to be transmitted |
| *PduInfoPtr* | Length of and pointer to the PDU data and pointer to MetaData. |

###### Returns

Std\_ReturnType

### src/3-Ecu/CanIf/inc/CanIf\_Types.h File Reference

This module is responsible for abstracting the can controllers from the upper layers.

#### Detailed Description

This module is responsible for abstracting the can controllers from the upper layers.

###### Author

Mohamed Hassanin (mohamed\_hassanin\_omran@yahoo.com)

###### Version

0.1

###### Date

2022-09-03

###### Copyright

Copyright (c) 2022

### src/3-Ecu/CanTrcv/inc/CanTrcv.h File Reference

This module is responsible for handling the can transceivers.

#include "CanTrcv\_Types.h"

#### Functions

* void **CanTrcv\_Init** (void)

*Initialize the Can transceiver module.*

* Std\_ReturnType **CanTrcv\_SetOpMode** (uint8 Transceiver, CanTrcv\_TrcvModeType OpMode)

*Sets the mode of the Transceiver to the value OpMode.*

#### Detailed Description

This module is responsible for handling the can transceivers.

###### Author

Mohamed Hassanin (mohamed\_hassanin\_omran@yahoo.com)

###### Version

0.1

###### Date

2022-09-03

###### Copyright

Copyright (c) 2022

#### Function Documentation

##### void CanTrcv\_Init (void )

Initialize the Can transceiver module.

##### Std\_ReturnType CanTrcv\_SetOpMode (uint8 *Transceiver*, CanTrcv\_TrcvModeType *OpMode*)

Sets the mode of the Transceiver to the value OpMode.

###### Parameters

|  |  |
| --- | --- |
| *Transceiver* | CAN transceiver to which API call has to be applied |
| *OpMode* | This parameter contains the desired operating mode |

###### Returns

Std\_ReturnType

### src/3-Ecu/CanTrcv/inc/CanTrcv\_Types.h File Reference

This module is responsible for handling the can transceivers.

#### Macros

* #define **CANTRCV\_TRCVMODE\_NORMAL**  0
* #define **CANTRCV\_TRCVMODE\_SLEEP**  1
* #define **CANTRCV\_TRCVMODE\_STANDBY**  2

#### Typedefs

* typedef uint8 **CanTrcv\_TrcvModeType**

#### Detailed Description

This module is responsible for handling the can transceivers.

###### Author

Mohamed Hassanin (mohamed\_hassanin\_omran@yahoo.com)

###### Version

0.1

###### Date

2022-09-03

###### Copyright

Copyright (c) 2022

### src/3-Ecu/Light/inc/Light.h File Reference

This module is responsible for abstracting the dio of the light.

#include "Light\_Types.h"

#### Functions

* void **Light\_Init** (void)

*Initialize the Light module.*

* void **Light\_SetStatus** (**Light\_StatusType** Status)

*Set the Light state (LIGHT\_STATUS\_OFF, LIGHT\_STATUS\_ON)*

#### Detailed Description

This module is responsible for abstracting the dio of the light.

###### Author

Mohamed Hassanin (mohamed\_hassanin\_omran@yahoo.com)

###### Version

0.1

###### Date

2022-09-03

###### Copyright

Copyright (c) 2022

#### Function Documentation

##### void Light\_Init (void )

Initialize the Light module.

##### void Light\_SetStatus (Light\_StatusType *Status*)

Set the Light state (LIGHT\_STATUS\_OFF, LIGHT\_STATUS\_ON)

###### Parameters

|  |  |
| --- | --- |
| *Status* |  |

### src/3-Ecu/Light/inc/Light\_Types.h File Reference

This module is responsible for abstracting the dio of the light.

#### Macros

* #define **LIGHT\_STATUS\_OFF**  0
* #define **LIGHT\_STATUS\_ON**  1

#### Typedefs

* typedef uint8 **Light\_StatusType**

*Light status LIGHT\_STATUS\_OFF, LIGHT\_STATUS\_ON.*

#### Detailed Description

This module is responsible for abstracting the dio of the light.

###### Author

Mohamed Hassanin (mohamed\_hassanin\_omran@yahoo.com)

###### Version

0.1

###### Date

2022-09-03

###### Copyright

Copyright (c) 2022

#### Typedef Documentation

##### typedef uint8 Light\_StatusType

Light status LIGHT\_STATUS\_OFF, LIGHT\_STATUS\_ON.

### src/4-Mcal/Can/inc/Can.h File Reference

CAN communication protocol driver.

#include "Can\_Types.h"

#include "Can\_GeneralTypes.h"

#### Functions

* void **Can\_Init** (void)

*Initialize the CAN driver.*

* Std\_ReturnType **Can\_Write** (**Can\_HwHandleType** Hth, const **Can\_PduType** \*PduInfo)

*This function is called by CanIf to pass a CAN message to CanDrv for transmission.*

#### Detailed Description

CAN communication protocol driver.

###### Author

Mohamed Hassanin (mohamed\_hassanin\_omran@yahoo.com)

###### Version

0.1

###### Date

2022-09-03

###### Copyright

Copyright (c) 2022

#### Function Documentation

##### void Can\_Init (void )

Initialize the CAN driver.

##### Std\_ReturnType Can\_Write (Can\_HwHandleType *Hth*, const Can\_PduType \* *PduInfo*)

This function is called by CanIf to pass a CAN message to CanDrv for transmission.

###### Parameters

|  |  |
| --- | --- |
| *Hth* | information which HW-transmit handle shall be used for transmit. Implicitly this is also the information about the controller to use because the Hth numbers are unique inside one hardware unit. |
| *PduInfo* | Pointer to SDU user memory, Data Length and Identifier. |

###### Returns

Std\_ReturnType

### src/4-Mcal/Can/inc/Can\_Types.h File Reference

CAN communication protocol driver.

#### Detailed Description

CAN communication protocol driver.

###### Author

Mohamed Hassanin (mohamed\_hassanin\_omran@yahoo.com)

###### Version

0.1

###### Date

2022-09-03

###### Copyright

Copyright (c) 2022

### src/4-Mcal/Dio/inc/Dio.h File Reference

Digital input/output driver.

#include "Dio\_Types.h"

#### Functions

* **Dio\_LevelType** **Dio\_ReadChannel** (**Dio\_ChannelType** ChannelId)

#### Detailed Description

Digital input/output driver.

###### Author

Mohamed Hassanin (mohamed\_hassanin\_omran@yahoo.com)

###### Version

0.1

###### Date

2022-09-03

###### Copyright

Copyright (c) 2022

#### Function Documentation

##### Dio\_LevelType Dio\_ReadChannel (Dio\_ChannelType *ChannelId*)

###### Parameters

|  |  |
| --- | --- |
| *ChannelId* |  |

###### Returns

Dio\_LevelType

### src/4-Mcal/Dio/inc/Dio\_Types.h File Reference

Digital input/output driver.

#### Macros

* #define **STD\_LOW**  0
* #define **STD\_HIGH**  1

#### Typedefs

* typedef uint8 **Dio\_LevelType**

*These are the possible levels a DIO channel can have (input or output):*

*STD\_LOW*

*STD\_HIGH*

*.*

* typedef uint8 **Dio\_ChannelType**

*Numeric ID of a DIO channel.*

#### Detailed Description

Digital input/output driver.

###### Author

Mohamed Hassanin (mohamed\_hassanin\_omran@yahoo.com)

###### Version

0.1

###### Date

2022-09-03

###### Copyright

Copyright (c) 2022

#### Typedef Documentation

##### typedef uint8 Dio\_ChannelType

Numeric ID of a DIO channel.

##### typedef uint8 Dio\_LevelType

These are the possible levels a DIO channel can have (input or output):

STD\_LOW

STD\_HIGH

.

### src/4-Mcal/Gpt/inc/Gpt.h File Reference

General purpose timer driver.

#include "Gpt\_Types.h"

#### Functions

* void **Gpt\_Init** (void)

*Initialize the general purpose timer.*

* void **Gpt\_StartTimer** (**Gpt\_ChannelType** Channel, **Gpt\_ValueType** Value)

*Starts a timer channel.*

* void **Gpt\_EnableNotification** (**Gpt\_ChannelType** Channel)

*Enables the interrupt notification for a channel (relevant in normal mode).*

#### Detailed Description

General purpose timer driver.

###### Author

Mohamed Hassanin (mohamed\_hassanin\_omran@yahoo.com)

###### Version

0.1

###### Date

2022-09-03

###### Copyright

Copyright (c) 2022

#### Function Documentation

##### void Gpt\_EnableNotification (Gpt\_ChannelType *Channel*)

Enables the interrupt notification for a channel (relevant in normal mode).

###### Parameters

|  |  |
| --- | --- |
| *Channel* | Numeric identifier of the GPT channel |

##### void Gpt\_Init (void )

Initialize the general purpose timer.

##### void Gpt\_StartTimer (Gpt\_ChannelType *Channel*, Gpt\_ValueType *Value*)

Starts a timer channel.

###### Parameters

|  |  |
| --- | --- |
| *Channel* | Numeric identifier of the GPT channel. |
| *Value* | Target time in number of ticks |

### src/4-Mcal/Gpt/inc/Gpt\_Types.h File Reference

General purpose timer driver.

#### Typedefs

* typedef uint8 **Gpt\_ChannelType**

*Numeric ID of a GPT channel.*

* typedef uint8 **Gpt\_ValueType**

*Type for reading and setting the timer values (in number of ticks).*

#### Detailed Description

General purpose timer driver.

###### Author

Mohamed Hassanin (mohamed\_hassanin\_omran@yahoo.com)

###### Version

0.1

###### Date

2022-09-03

###### Copyright

Copyright (c) 2022

#### Typedef Documentation

##### typedef uint8 Gpt\_ChannelType

Numeric ID of a GPT channel.

##### typedef uint8 Gpt\_ValueType

Type for reading and setting the timer values (in number of ticks).

### src/4-Mcal/Icu/inc/Icu.h File Reference

Input capture unit driver.

#include "Icu\_Types.h"

#### Functions

* void **Icu\_Init** (void)

*Initialize the Input Capture Unit (ICU).*

* **Icu\_ValueType** **Icu\_GetTimeElapsed** (**Icu\_ChannelType** Channel)

*This function reads the elapsed Time for the given channel.*

#### Detailed Description

Input capture unit driver.

###### Author

Mohamed Hassanin (mohamed\_hassanin\_omran@yahoo.com)

###### Version

0.1

###### Date

2022-09-03

###### Copyright

Copyright (c) 2022

#### Function Documentation

##### Icu\_ValueType Icu\_GetTimeElapsed (Icu\_ChannelType *Channel*)

This function reads the elapsed Time for the given channel.

###### Parameters

|  |  |
| --- | --- |
| *Channel* | Numeric identifier of the ICU channel |

###### Returns

Icu\_ValueType

##### void Icu\_Init (void )

Initialize the Input Capture Unit (ICU).

### src/4-Mcal/Icu/inc/Icu\_Types.h File Reference

Input capture unit driver.

#### Typedefs

* typedef uint8 **Icu\_ChannelType**

*Numeric identifier of the ICU channel.*

* typedef uint8 **Icu\_ValueType**

*Width of the buffer for timestamp ticks and measured elapsed timeticks.*

#### Detailed Description

Input capture unit driver.

###### Author

Mohamed Hassanin (mohamed\_hassanin\_omran@yahoo.com)

###### Version

0.1

###### Date

2022-09-03

###### Copyright

Copyright (c) 2022

#### Typedef Documentation

##### typedef uint8 Icu\_ChannelType

Numeric identifier of the ICU channel.

##### typedef uint8 Icu\_ValueType

Width of the buffer for timestamp ticks and measured elapsed timeticks.

### src/4-Mcal/Port/inc/Port.h File Reference

Port driver.

#include "Port\_Types.h"

#### Functions

* void **Port\_Init** (void)

*Initialize the Port module.*

* void **Port\_SetPinDirection** (**Port\_PinType** Pin, **Port\_PinDirectionType** Direction)

*Sets the port pin direction.*

* void **Port\_SetPinMode** (**Port\_PinType** Pin, **Port\_PinModeType** Mode)

*Sets the port pin mode.*

#### Detailed Description

Port driver.

###### Author

Mohamed Hassanin (mohamed\_hassanin\_omran@yahoo.com)

###### Version

0.1

###### Date

2022-09-03

###### Copyright

Copyright (c) 2022

#### Function Documentation

##### void Port\_Init (void )

Initialize the Port module.

##### void Port\_SetPinDirection (Port\_PinType *Pin*, Port\_PinDirectionType *Direction*)

Sets the port pin direction.

###### Parameters

|  |  |
| --- | --- |
| *Pin* | Port Pin ID number |
| *Direction* | Port Pin Direction |

##### void Port\_SetPinMode (Port\_PinType *Pin*, Port\_PinModeType *Mode*)

Sets the port pin mode.

###### Parameters

|  |  |
| --- | --- |
| *Pin* | Port Pin ID number |
| *Mode* | New Port Pin mode to be set on port pin. |

### src/4-Mcal/Port/inc/Port\_Types.h File Reference

Port driver.

#### Typedefs

* typedef uint8 **Port\_PinDirectionType**

*Possible directions of a port pin.:*

*PORT\_PIN\_IN*

*PORT\_PIN\_OUT*

*.*

* typedef uint8 **Port\_PinType**

*Data type for the symbolic name of a port pin.*

* typedef uint8 **Port\_PinModeType**

*Different port pin modes.*

#### Detailed Description

Port driver.

###### Author

Mohamed Hassanin (mohamed\_hassanin\_omran@yahoo.com)

###### Version

0.1

###### Date

2022-09-03

###### Copyright

Copyright (c) 2022

#### Typedef Documentation

##### typedef uint8 Port\_PinDirectionType

Possible directions of a port pin.:

PORT\_PIN\_IN

PORT\_PIN\_OUT

.

##### typedef uint8 Port\_PinModeType

Different port pin modes.

##### typedef uint8 Port\_PinType

Data type for the symbolic name of a port pin.

### src/4-Mcal/Spi/inc/Spi.h File Reference

SPI communication protocol driver.

#include "Spi\_Types.h"

#### Functions

* void **Spi\_Init** (void)

*Initialize the SPI driver.*

* Std\_ReturnType **Spi\_Write** (**Spi\_ChannelType** Channel, const **Spi\_DataBufferType** \*DataBufferPtr)

*Service for writing synchronously one or more data from a SPI Handler/Driver Channel specified by parameter.*

* Std\_ReturnType **Spi\_Read** (**Spi\_ChannelType** Channel, **Spi\_DataBufferType** \*DataBufferPointer)

*Service for reading synchronously one or more data from a SPI Handler/Driver Channel specified by parameter.*

#### Detailed Description

SPI communication protocol driver.

###### Author

Mohamed Hassanin (mohamed\_hassanin\_omran@yahoo.com)

###### Version

0.1

###### Date

2022-09-03

###### Copyright

Copyright (c) 2022

#### Function Documentation

##### void Spi\_Init (void )

Initialize the SPI driver.

##### Std\_ReturnType Spi\_Read (Spi\_ChannelType *Channel*, Spi\_DataBufferType \* *DataBufferPointer*)

Service for reading synchronously one or more data from a SPI Handler/Driver Channel specified by parameter.

###### Parameters

|  |  |
| --- | --- |
| *Channel* | Channel ID. |
| *DataBufferPointer* | Pointer to destination data buffer in RAM |

###### Returns

Std\_ReturnType

##### Std\_ReturnType Spi\_Write (Spi\_ChannelType *Channel*, const Spi\_DataBufferType \* *DataBufferPtr*)

Service for writing synchronously one or more data from a SPI Handler/Driver Channel specified by parameter.

###### Parameters

|  |  |
| --- | --- |
| *Channel* | Channel ID. |
| *DataBufferPtr* | Pointer to source data buffer in RAM |

###### Returns

Std\_ReturnType

### src/4-Mcal/Spi/inc/Spi\_Types.h File Reference

SPI communication protocol driver.

#### Typedefs

* typedef uint8 **Spi\_ChannelType**

*Specifies the identification (ID) for a Channel.*

* typedef uint8 **Spi\_DataBufferType**

*Type of application data buffer elements.*

#### Detailed Description

SPI communication protocol driver.

###### Author

Mohamed Hassanin (mohamed\_hassanin\_omran@yahoo.com)

###### Version

0.1

###### Date

2022-09-03

###### Copyright

Copyright (c) 2022

#### Typedef Documentation

##### typedef uint8 Spi\_ChannelType

Specifies the identification (ID) for a Channel.

##### typedef uint8 Spi\_DataBufferType

Type of application data buffer elements.

### src/5-Common/inc/Can\_GeneralTypes.h File Reference

This file contains the CAN communication CAN\_GENERAL\_TYPES types definitions.

#include "Std\_Types.h"

#include "ComStack\_Types.h"

#### Data Structures

* struct **Can\_PduType**

*This type unites PduId (swPduHandle), SduLength (length), SduData (sdu), and CanId (id) for any CAN L-SDU.*

* struct **Can\_HwType**

*This type defines a data structure which clearly provides an Hardware Object Handle including its corresponding CAN Controller and therefore CanDrv as well as the specific CanId..*

#### Typedefs

* typedef struct **Can\_PduType** **Can\_PduType**

*This type unites PduId (swPduHandle), SduLength (length), SduData (sdu), and CanId (id) for any CAN L-SDU.*

* typedef uint32 **Can\_IdType**

*Represents the Identifier of an L-PDU. The two most significant bits specify the frame type:*

*00 CAN message with Standard CAN ID*

*01 CAN FD frame with Standard CAN ID*

*10 CAN message with Extended CAN ID*

*11 CAN FD frame with Extended CAN ID*

*.*

* typedef uint16 **Can\_HwHandleType**

*Represents the hardware object handles of a CAN hardware unit. For CAN hardware units with more than 255 HW objects use extended range. Ranges:*

*standard: 0..0x0FF*

*Extended: 0..0xFFFF*

*.*

* typedef struct **Can\_HwType** **Can\_HwType**

*This type defines a data structure which clearly provides an Hardware Object Handle including its corresponding CAN Controller and therefore CanDrv as well as the specific CanId..*

#### Detailed Description

This file contains the CAN communication CAN\_GENERAL\_TYPES types definitions.

###### Author

Mohamed Hassanin (mohamed\_hassanin\_omran@yahoo.com)

###### Version

0.1

###### Date

2022-09-03

###### Copyright

Copyright (c) 2022

#### Typedef Documentation

##### typedef uint16 Can\_HwHandleType

Represents the hardware object handles of a CAN hardware unit. For CAN hardware units with more than 255 HW objects use extended range. Ranges:

standard: 0..0x0FF

Extended: 0..0xFFFF

.

##### typedef struct Can\_HwType Can\_HwType

This type defines a data structure which clearly provides an Hardware Object Handle including its corresponding CAN Controller and therefore CanDrv as well as the specific CanId..

##### typedef struct Can\_PduType Can\_PduType

This type unites PduId (swPduHandle), SduLength (length), SduData (sdu), and CanId (id) for any CAN L-SDU.

### src/5-Common/inc/ComStack\_Types.h File Reference

This file contains the communication COM\_STACK\_TYPES types definitions.

#include "Std\_Types.h"

#### Data Structures

* struct **PduInfoType**

*The pdu information contains the SDU length and the payload.*

#### Typedefs

* typedef uint16 **PduIdType**

*The pdu id.*

* typedef uint16 **PduLengthType**
* typedef struct **PduInfoType** **PduInfoType**

*The pdu information contains the SDU length and the payload.*

#### Detailed Description

This file contains the communication COM\_STACK\_TYPES types definitions.

###### Author

Mohamed Hassanin (mohamed\_hassanin\_omran@yahoo.com)

###### Version

0.1

###### Date

2022-09-03

###### Copyright

Copyright (c) 2022

#### Typedef Documentation

##### typedef uint16 PduIdType

The pdu id.

##### typedef struct PduInfoType PduInfoType

The pdu information contains the SDU length and the payload.

## Prepare your folder structure according to the previous points

I ran tree command to print my folder structure and this is the output:

| FolderStructure.txt

|

+---1-Appl

| +---BuzzerCtrl

| | | BuzzerCtrl.c

| | |

| | \---inc

| | BuzzerCtrl.h

| | BuzzerCtrl\_Types.h

| |

| \---LightCtrl

| | LightCtrl.c

| |

| \---inc

| LightCtrl.h

| LightCtrl\_Types.h

|

+---2-Service

| +---Bcm

| | | Bcm.c

| | |

| | \---inc

| | Bcm.h

| | Bcm\_Types.h

| |

| +---Os

| | | Os.c

| | |

| | \---inc

| | Os.h

| | Os\_Types.h

| |

| \---PduR

| | PduR.c

| |

| \---inc

| PduR.h

| PduR\_Types.h

|

+---3-Ecu

| +---Buzzer

| | | Buzzer.c

| | |

| | \---inc

| | Buzzer.h

| | Buzzer\_Types.h

| |

| +---CanIf

| | | CanIf.c

| | |

| | \---inc

| | CanIf.h

| | CanIf\_Types.h

| |

| +---CanTrcv

| | | CanTrcv.c

| | |

| | \---inc

| | CanTrcv.h

| | CanTrcv\_Types.h

| |

| \---Light

| | Light.c

| |

| \---inc

| Light.h

| Light\_Types.h

|

+---4-Mcal

| +---Can

| | | Can.c

| | |

| | \---inc

| | Can.h

| | Can\_Types.h

| |

| +---Dio

| | | Dio.c

| | |

| | \---inc

| | Dio.h

| | Dio\_Types.h

| |

| +---Gpt

| | | Gpt.c

| | |

| | \---inc

| | Gpt.h

| | Gpt\_Types.h

| |

| +---Icu

| | | Icu.c

| | |

| | \---inc

| | Icu.h

| | Icu\_Types.h

| |

| +---Port

| | | Port.c

| | |

| | \---inc

| | Port.h

| | Port\_Types.h

| |

| \---Spi

| | Spi.c

| |

| \---inc

| Spi.h

| Spi\_Types.h

|

\---5-Common

+---inc

| Can\_GeneralTypes.h

| Compiler.h

| ComStack\_Types.h

| Mcu\_Hw.h

| Platform\_Types.h

| Std\_Types.h

|

\---src