ECU1 Static Design

Make the layered architecture

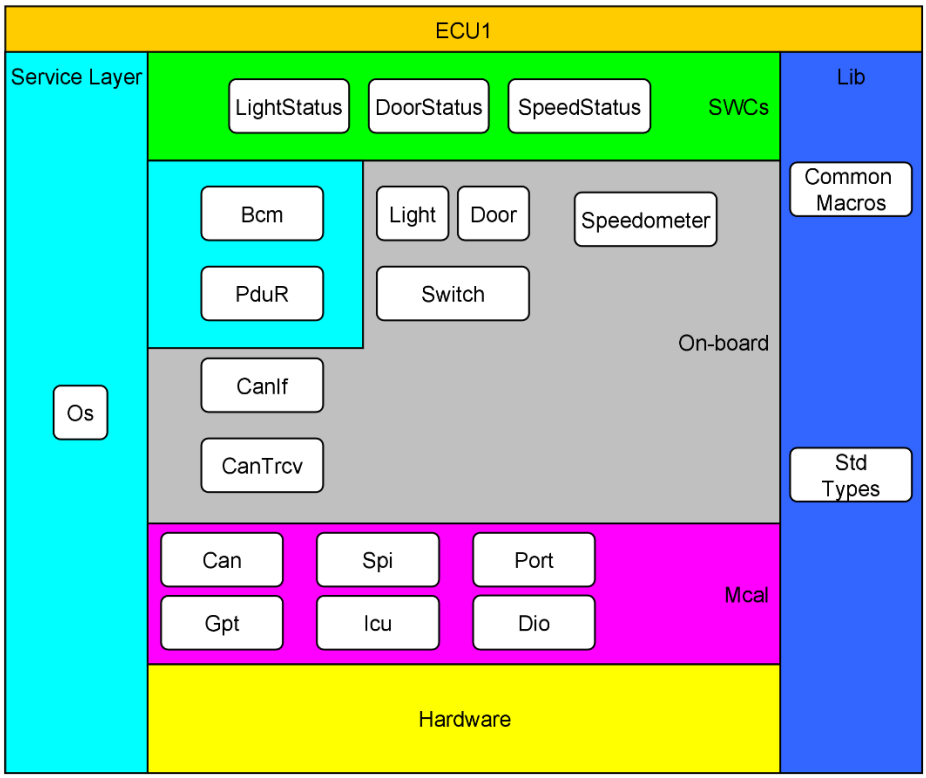

Figure 1 ECU1 static design

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

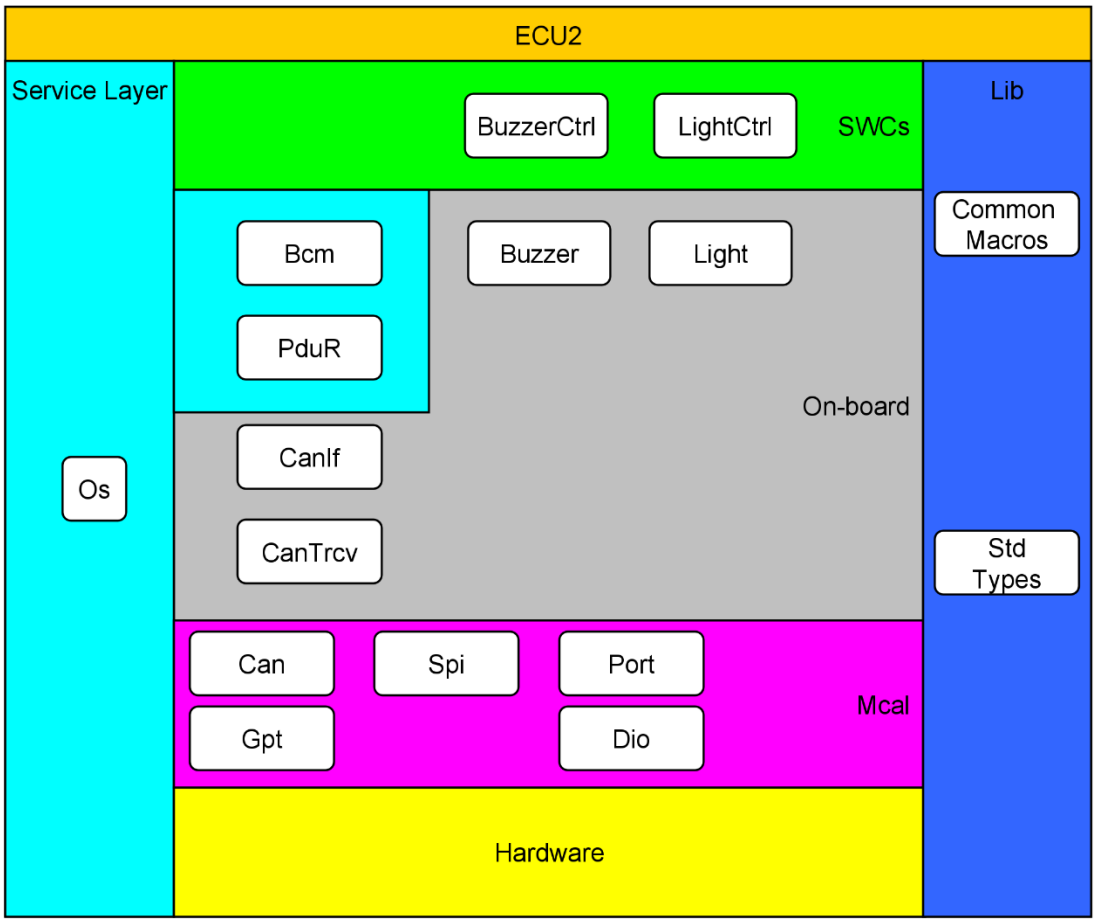

Figure 2 ECU2 static design

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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
```