**Autosar CAN Communication Stack**

# Summary:

The project code implements part of the Autosar Communication Stack (BASIC SOFTWARE LAYER) for understanding.

# File Structure & Architecture:

Source files can be grouped in modules/components as below:

A screenshot of a video game

Description automatically generated

|  |
| --- |
| **Autosar COM** **(Service Layer)** |
| COM.c |
| COM.h |
| COM\_types.h |
| COM\_GroupCfg.h |
| COM\_SignalCfg.h |

|  |
| --- |
| **PDU Router** **(Service Layer)** |
| PduR\_If.c |
| PduR\_If .h |
| PduR\_If\_types.h |
| PduR\_If\_Cfg.h |

|  |
| --- |
| **CAN Interface** **(ECU Abstraction Layer)** |
| CanIf.c |
| CanIf .h |
| CanIf\_Cfg.c |
| CanIf\_Cfg.h |

|  |
| --- |
| **CAN Driver (not implemented – Micro controller Abstraction Layer)** |
| CanDrv.c |
| CanDrv .h |

## Signal Transmission from Service Layer to Drivers

A picture containing screenshot

Description automatically generated

* Application Software Components (SWC) use the Run Time Environment API to send the signal to BSW (Basic Software).
* Depending on the Signal Configuration (timing, scaling, etc.), COM component sends the signal to the group.
* The group(message) is further transmitted to the lower layers via PDU router (network layer). PDU Router decides the network path and interface, and as per message configuration sends it to appropriate driver interface.
* Driver interface then decides the port and routes the messages to appropriate peripheral (in our case CAN controller)

A screenshot of a cell phone

Description automatically generated

## Signal Reception from Drivers to Service Layers

As expected for RX, exactly opposite path is followed. Please see below diagram.

A screenshot of a video game

Description automatically generated

A screenshot of a cell phone

Description automatically generated

# References:

<https://www.autosar.org/fileadmin/user_upload/standards/classic/4-3/AUTOSAR_SWS_SocketAdaptor.pdf>