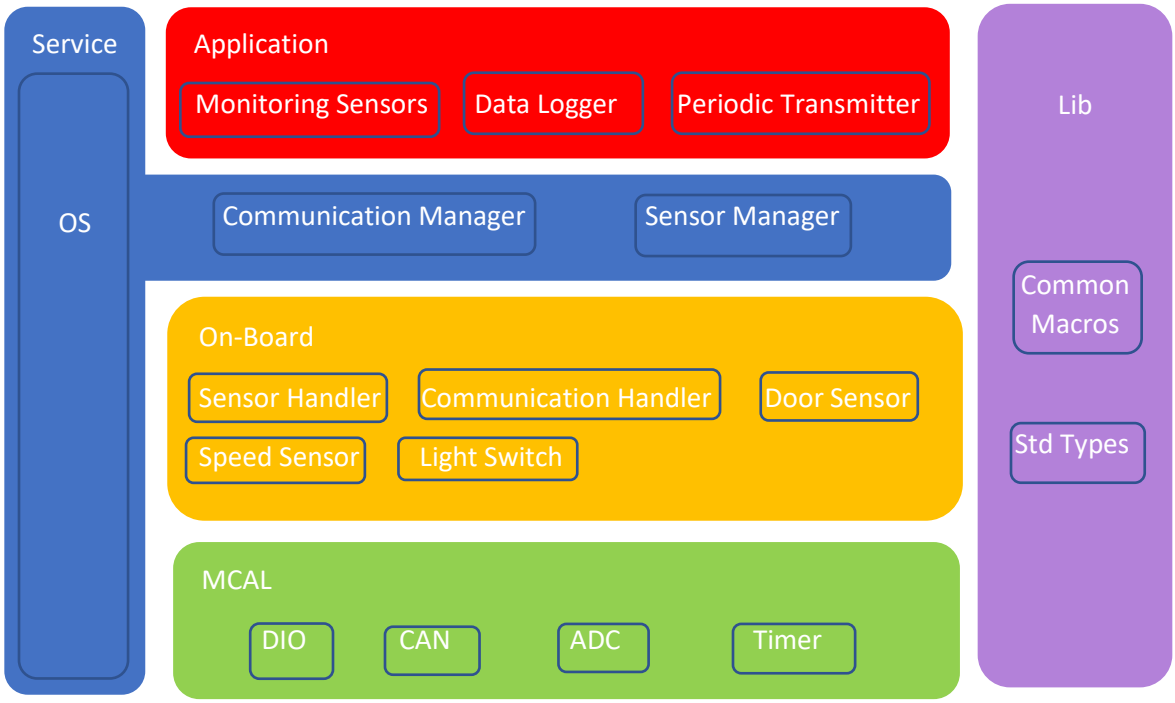
Static Design Analysis

ECU 1

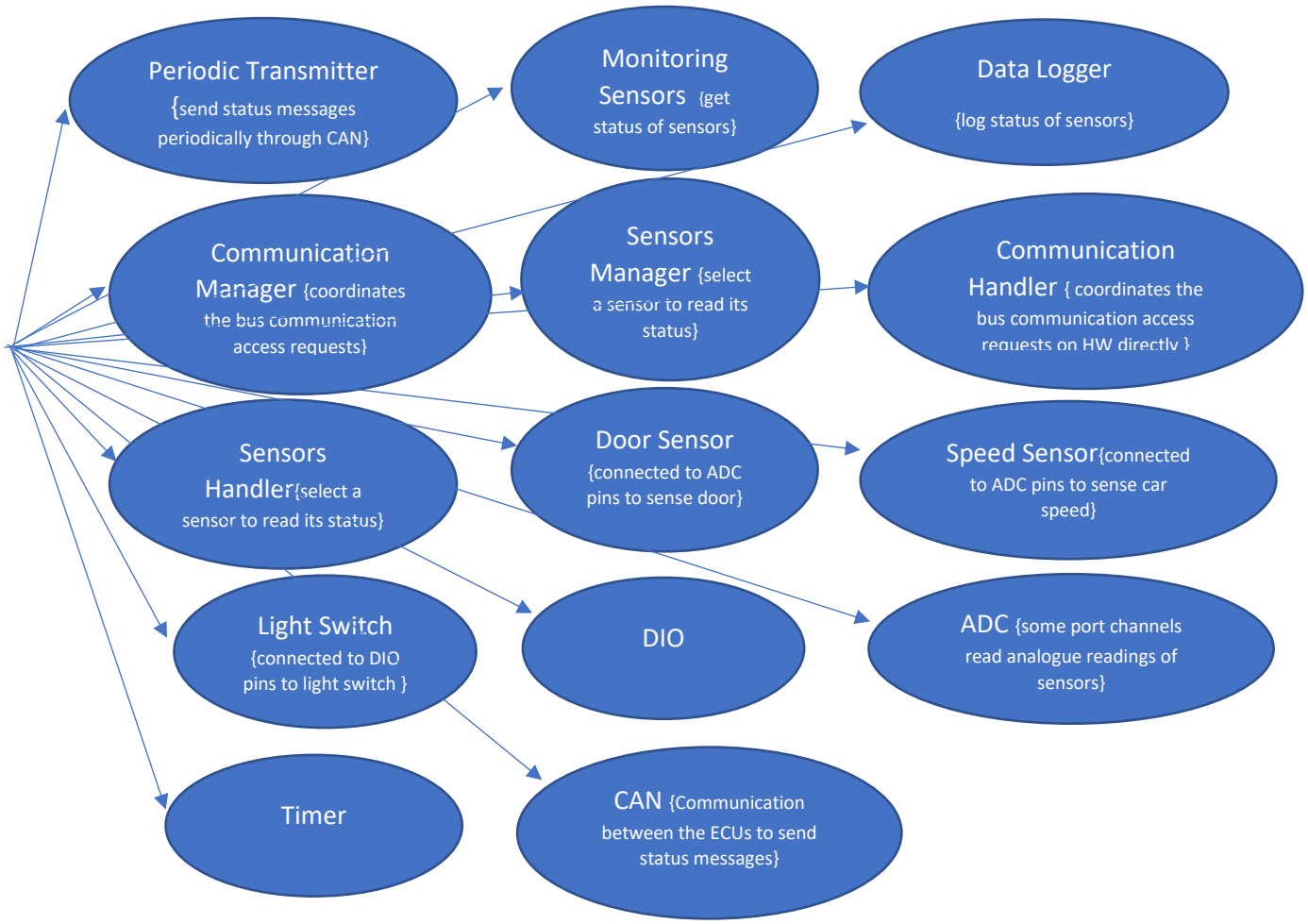
1. Layered Architecture:



Notes and Assumptions:

- Door sensor receives Digital Reading, High if closed and Low if opened
- Speed sensor receives Analogue Reading, Car speed then converted to Digital High if any non-zero value, and Low if speed equals zero.
- Light Switch is also controlled by the sensor manager and receives Digital Reading
- We Have three tasks (Door Sensor Task, Speed Sensor Task and Light Switch Task) each related to some sensor, each task does the following: Monitor Sensor (done by Sensor manager), Log Readings, and periodically Transmit via CAN bus (done by Communication manager)

2. Components/Modules



3. APIs

Module	APIs	API Details
Application Tasks (SW components)	DoorSensorTask SpeedSensorTask LightSwitchTask	Syntax: void DoorSensorTask(void); Sync/Async: Synchronous Reentrancy: Non-Reentrant Parameters: None Return: None Description: Door Sensor Task Syntax: void SpeedSensorTask(void); Sync/Async: Synchronous Reentrancy: Non-Reentrant Parameters: None Return: None Description: Speed Sensor Task

		Syntax:	void LightSwitchTask(void);
		Sync/Async:	Synchronous
		Reentrancy:	Non-Reentrant
		Parameters:	None
		Return:	None
		Description:	Light Switch Task
Data Logger	DataLogger_LogData	Syntax:	void DataLogger_LogData (uint64 data);
		Sync/Async:	Synchronous
		Reentrancy:	Non-Reentrant
		Parameters:	data
		Return:	None
		Description:	Logs data sent to it
Communication Manager	BCM_Manager	Syntax:	void BCM_Manager (uint64 data , uint8 bus);
		Sync/Async:	Synchronous
		Reentrancy:	Non-Reentrant
		Parameters:	Data , Selected Bus
		Return:	None
		Description:	coordinates the bus communication access requests
Sensor Manager (do Monitoring Sensors)	Sensor_Manager	Syntax:	Uint8 Sensor_Manager (uint8 sensorID);
		Sync/Async:	Synchronous
		Reentrancy:	Non-Reentrant
		Parameters:	Selected Sensor
		Return:	Sensor status
		Description:	Selects sensor to read status from
Communication Handler	BCM_Handler	Syntax:	void BCM_Handler (uint64 data , uint8 bus);
		Sync/Async:	Synchronous
		Reentrancy:	Non-Reentrant
		Parameters:	Data, Selected Bus
		Return:	None
		Description:	coordinates the bus communication access requests but deals with HW directly
Sensor Handler	Sensor_Handler	Syntax:	Uint8 Sensor_Handler (uint8 sensorID);
		Sync/Async:	Synchronous
		Reentrancy:	Non-Reentrant
		Parameters:	Selected Sensor
		Return:	Data
		Description:	Selects sensor to read status from but deals with HW directly
Door Sensor	DoorSensor_Init DoorSensor_ReadStatus	Syntax:	ERROR_STATE DoorSensor_Init (void);
		Sync/Async:	Synchronous

<table>
<tr><td></td><td></td><td>
<table>
<tr><td>Reentrancy:</td><td>Non-Reentrant</td></tr>
<tr><td>Parameters:</td><td>None</td></tr>
<tr><td>Return:</td><td>ERROR_STATE</td></tr>
<tr><td>Description:</td><td>Initialize the used Dio pins for Digital input</td></tr>
</table>
<table>
<tr><td>Syntax:</td><td>Uint8 DoorSensor_ReadStatus(void);</td></tr>
<tr><td>Sync/Async:</td><td>Synchronous</td></tr>
<tr><td>Reentrancy:</td><td>Non-Reentrant</td></tr>
<tr><td>Parameters:</td><td>None</td></tr>
<tr><td>Return:</td><td>Sensor Status (opened / closed)</td></tr>
<tr><td>Description:</td><td>Read the status of the sensor to check status of car component(door , car speed , light switch)</td></tr>
</table>
</td></tr>
<tr><td>Speed Sensor</td><td>SpeedSensor_Init
SpeedSensor_ReadStatus</td><td>
<table>
<tr><td>Syntax:</td><td>ERROR_STATE SpeedSensor_Init (void);</td></tr>
<tr><td>Sync/Async:</td><td>Synchronous</td></tr>
<tr><td>Reentrancy:</td><td>Non-Reentrant</td></tr>
<tr><td>Parameters:</td><td>None</td></tr>
<tr><td>Return:</td><td>ERROR_STATE</td></tr>
<tr><td>Description:</td><td>Initialize the used pins for Analogue input (input to ADC)</td></tr>
</table>
<table>
<tr><td>Syntax:</td><td>Uint8 SpeedSensor_ReadStatus(void);</td></tr>
<tr><td>Sync/Async:</td><td>Synchronous</td></tr>
<tr><td>Reentrancy:</td><td>Non-Reentrant</td></tr>
<tr><td>Parameters:</td><td>None</td></tr>
<tr><td>Return:</td><td>Sensor Status (moving/stopped)</td></tr>
<tr><td>Description:</td><td>Read the status of sensor to check status of car component(door , car speed , light switch)</td></tr>
</table>
</td></tr>
<tr><td>Light Switch</td><td>LightSwitch_Init
LightSwitch_ReadStatus</td><td>
<table>
<tr><td>Syntax:</td><td>ERROR_STATE LightSwitch_Init (void);</td></tr>
<tr><td>Sync/Async:</td><td>Synchronous</td></tr>
<tr><td>Reentrancy:</td><td>Non-Reentrant</td></tr>
<tr><td>Parameters:</td><td>None</td></tr>
<tr><td>Return:</td><td>ERROR_STATE</td></tr>
<tr><td>Description:</td><td>Initialize the used pins for Digital input (input to Dio)</td></tr>
</table>
<table>
<tr><td>Syntax:</td><td>Uint8 LightSwitch_ReadStatus(void);</td></tr>
<tr><td>Sync/Async:</td><td>Synchronous</td></tr>
<tr><td>Reentrancy:</td><td>Non-Reentrant</td></tr>
<tr><td>Parameters:</td><td>None</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td></td><td></td><td><table><tr><td>Return:</td><td>Sensor Status (Pressed/unpressed)</td></tr><tr><td>Description:</td><td>Read the status of sensor to check status of car component(door , car speed , light switch)</td></tr></table></td></tr>
<tr><td>DIO</td><td>Dio_Init
Dio_WriteChannel
Dio_ReadChannel</td><td><table><tr><td>Syntax:</td><td>ERROR_STATE Dio_Init (void);</td></tr><tr><td>Sync/Async:</td><td>Synchronous</td></tr><tr><td>Reentrancy:</td><td>Non-Reentrant</td></tr><tr><td>Parameters:</td><td>None</td></tr><tr><td>Return:</td><td>ERROR_STATE</td></tr><tr><td>Description:</td><td>Initialize Dio pins with required configuration</td></tr></table>
<table><tr><td>Syntax:</td><td>void Dio_WriteChannel (DioPinLevel value , DioPinType id);</td></tr><tr><td>Sync/Async:</td><td>Synchronous</td></tr><tr><td>Reentrancy:</td><td>Non-Reentrant</td></tr><tr><td>Parameters:</td><td>Value (pin value) , pin id</td></tr><tr><td>Return:</td><td>None</td></tr><tr><td>Description:</td><td>Write value on pin</td></tr></table>
<table><tr><td>Syntax:</td><td>DioPinLevel Dio_ReadChannel (DioPinType id);</td></tr><tr><td>Sync/Async:</td><td>Synchronous</td></tr><tr><td>Reentrancy:</td><td>Non-Reentrant</td></tr><tr><td>Parameters:</td><td>Id (pin id)</td></tr><tr><td>Return:</td><td>Pin value</td></tr><tr><td>Description:</td><td>Read value on pin</td></tr></table></td></tr>
<tr><td>ADC</td><td>ADC_Init
ADC_ReadChannel</td><td><table><tr><td>Syntax:</td><td>ERROR_STATE ADC_Init (uint8 id);</td></tr><tr><td>Sync/Async:</td><td>Synchronous</td></tr><tr><td>Reentrancy:</td><td>Non-Reentrant</td></tr><tr><td>Parameters:</td><td>Pin id</td></tr><tr><td>Return:</td><td>ERROR_STATE</td></tr><tr><td>Description:</td><td>Initialize pins to act as analogue inputs to read sensors</td></tr></table>
<table><tr><td>Syntax:</td><td>Uint8 ADC_ReadChannel (uint8 id);</td></tr><tr><td>Sync/Async:</td><td>Synchronous</td></tr><tr><td>Reentrancy:</td><td>Non-Reentrant</td></tr><tr><td>Parameters:</td><td>Id (pin id)</td></tr><tr><td>Return:</td><td>Pin value</td></tr><tr><td>Description:</td><td>Read value on pin</td></tr></table></td></tr>
<tr><td>Timer</td><td>Timer_Init</td><td><table><tr><td>Syntax:</td><td>ERROR_STATE Timer_Init</td></tr></table></td></tr>
</table>

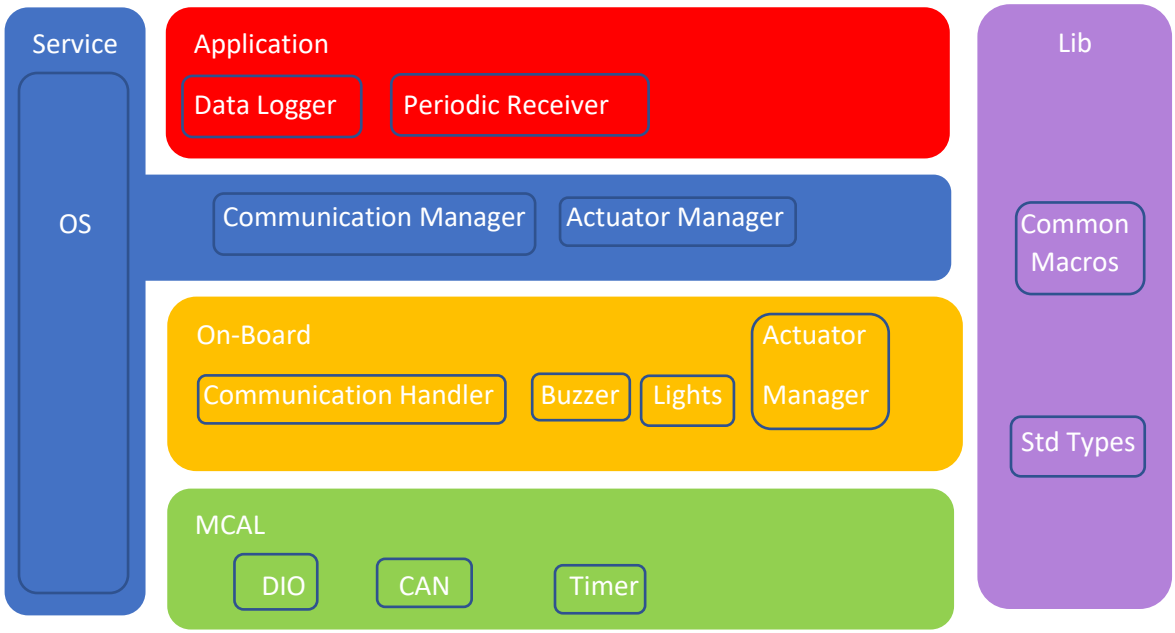
<table>
<tr><td></td><td>Timer_Start</td><td>
<table>
<tr><td></td><td>(void);</td></tr>
<tr><td>Sync/Async:</td><td>Synchronous</td></tr>
<tr><td>Reentrancy:</td><td>Non-Reentrant</td></tr>
<tr><td>Parameters:</td><td>None</td></tr>
<tr><td>Return:</td><td>ERROR_STATE</td></tr>
<tr><td>Description:</td><td>Initialize pins connected to timer internally , and initialize mode and other configurations</td></tr>
</table>
<table>
<tr><td>Syntax:</td><td>void Timer_start (TimerTickType ticks);</td></tr>
<tr><td>Sync/Async:</td><td>Synchronous</td></tr>
<tr><td>Reentrancy:</td><td>Non-Reentrant</td></tr>
<tr><td>Parameters:</td><td>Ticks</td></tr>
<tr><td>Return:</td><td>None</td></tr>
<tr><td>Description:</td><td>Make timer count till count = ticks to achieve required periodicity for sending status messages</td></tr>
</table>
</td></tr>
<tr><td>CAN</td><td>CAN_Init
CAN_Transmit</td><td>
<table>
<tr><td>Syntax:</td><td>ERROR_STATE CAN_Init (void);</td></tr>
<tr><td>Sync/Async:</td><td>Synchronous</td></tr>
<tr><td>Reentrancy:</td><td>Non-Reentrant</td></tr>
<tr><td>Parameters:</td><td>None</td></tr>
<tr><td>Return:</td><td>ERROR_STATE</td></tr>
<tr><td>Description:</td><td>Initialize CAN with required configuration</td></tr>
</table>
<table>
<tr><td>Syntax:</td><td>void CAN_transmit (uint64 data , DioPinType canPinid);</td></tr>
<tr><td>Sync/Async:</td><td>Synchronous</td></tr>
<tr><td>Reentrancy:</td><td>Non-Reentrant</td></tr>
<tr><td>Parameters:</td><td>data , CAN pin id</td></tr>
<tr><td>Return:</td><td>None</td></tr>
<tr><td>Description:</td><td>Transmit data through CAN</td></tr>
</table>
</td></tr>
</table>

4. Typedefs

Typedef unsigned char uint8	
Typedef unsigned long long uint64	used because max width of data in CAN frame is 64 bits
Typedef uint8 DioPinLevel	hold value of pin either HIGH = 1 or LOW = 0
Typedef uint8 DioPinType	hold value of pin id

ECU 2

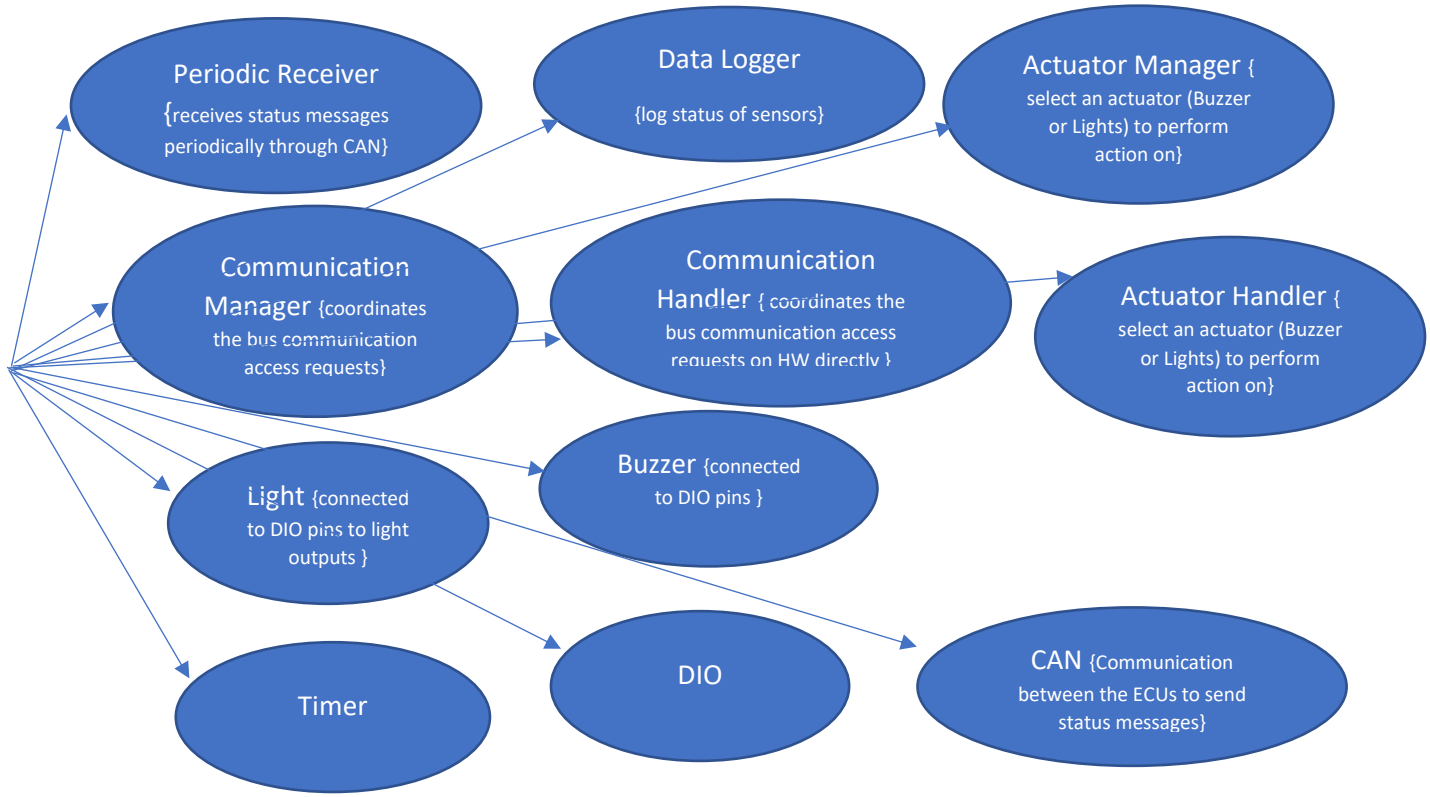
1. Layered Architecture:



Notes and Assumptions:

-We have one task that receives the status messages via CAN bus, checks their values versus the given cases, and uses actuators (Buzzer and Lights) to do some action based on this condition

2. Components/Modules



3. APIs

Module	APIs	API Details	
Periodic Receiver	PeriodicReceive_Status	Syntax:	void PeriodicReceive_Status (uint64* data , uint8* id);
		Sync/Async:	Synchronous
		Reentrancy:	Non-Reentrant
		Parameters:	Pointer to data act as buffer for data , pointer to id to save received id to check it
		Return:	None
		Description:	Receive status periodically from ECU 1 through CAN
Data Logger	DataLogger_LogData	Syntax:	void DataLogger_LogData (uint64 data);
		Sync/Async:	Synchronous
		Reentrancy:	Non-Reentrant
		Parameters:	data
		Return:	None
		Description:	Logs data sent to it
Communication Manager	BCM_Manager	Syntax:	void BCM_Manager (uint64 data , uint8 bus);
		Sync/Async:	Synchronous
		Reentrancy:	Non-Reentrant
		Parameters:	Data , Selected Bus
		Return:	None
		Description:	coordinates the bus communication access requests
Actuator Manager	Actuator_Manager	Syntax:	void Actuator_Manager (uint8 actuatorID , uint8 action);
		Sync/Async:	Synchronous
		Reentrancy:	Non-Reentrant
		Parameters:	ID of actuator , Action taken (ON , OFF)
		Return:	None
		Description:	Select actuator to do some action
Communication Handler	BCM_Handler	Syntax:	void BCM_Handler (uint64 data , uint8 bus);
		Sync/Async:	Synchronous
		Reentrancy:	Non-Reentrant
		Parameters:	Data , Selected Bus
		Return:	None
		Description:	coordinates the bus communication access requests but deals with HW directly
Actuator Handler	Actuator_Handler	Syntax:	void Actuator_Handler (uint8 actuatorID , uint8 action);

<table>
<tr><td></td><td></td><td><table>
<tr><td>Sync/Async:</td><td>Synchronous</td></tr>
<tr><td>Reentrancy:</td><td>Non-Reentrant</td></tr>
<tr><td>Parameters:</td><td>ID of actuator , Action taken (ON , OFF)</td></tr>
<tr><td>Return:</td><td>None</td></tr>
<tr><td>Description:</td><td>Select actuator to do some action</td></tr>
</table></td></tr>
<tr><td>Buzzer</td><td>Buzzer_Init
Buzzer_ON
Buzzer_OFF</td><td><table>
<tr><td>Syntax:</td><td>ERROR_STATE Buzzer_Init (void);</td></tr>
<tr><td>Sync/Async:</td><td>Synchronous</td></tr>
<tr><td>Reentrancy:</td><td>Non-Reentrant</td></tr>
<tr><td>Parameters:</td><td>None</td></tr>
<tr><td>Return:</td><td>ERROR_STATE</td></tr>
<tr><td>Description:</td><td>Initialize the used pins for Digital input (input to Dio)</td></tr>
</table>
<table>
<tr><td>Syntax:</td><td>void Buzzer_ON(void);</td></tr>
<tr><td>Sync/Async:</td><td>Synchronous</td></tr>
<tr><td>Reentrancy:</td><td>Non-Reentrant</td></tr>
<tr><td>Parameters:</td><td>None</td></tr>
<tr><td>Return:</td><td>None</td></tr>
<tr><td>Description:</td><td>Turn ON Buzzer</td></tr>
</table>
<table>
<tr><td>Syntax:</td><td>void Buzzer_OFF(void);</td></tr>
<tr><td>Sync/Async:</td><td>Synchronous</td></tr>
<tr><td>Reentrancy:</td><td>Non-Reentrant</td></tr>
<tr><td>Parameters:</td><td>None</td></tr>
<tr><td>Return:</td><td>None</td></tr>
<tr><td>Description:</td><td>Turn OFF Buzzer</td></tr>
</table></td></tr>
<tr><td>Lights</td><td>Lights_Init
Lights_ON
Lights_OFF</td><td><table>
<tr><td>Syntax:</td><td>ERROR_STATE Lights_Init (void);</td></tr>
<tr><td>Sync/Async:</td><td>Synchronous</td></tr>
<tr><td>Reentrancy:</td><td>Non-Reentrant</td></tr>
<tr><td>Parameters:</td><td>None</td></tr>
<tr><td>Return:</td><td>ERROR_STATE</td></tr>
<tr><td>Description:</td><td>Initialize the used pins for Digital input (input to Dio)</td></tr>
</table>
<table>
<tr><td>Syntax:</td><td>void Lights_ON(void);</td></tr>
<tr><td>Sync/Async:</td><td>Synchronous</td></tr>
<tr><td>Reentrancy:</td><td>Non-Reentrant</td></tr>
<tr><td>Parameters:</td><td>None</td></tr>
<tr><td>Return:</td><td>None</td></tr>
<tr><td>Description:</td><td>Turn ON lights</td></tr>
</table>
<table>
<tr><td>Syntax:</td><td>void Lights_OFF(void);</td></tr>
<tr><td>Sync/Async:</td><td>Synchronous</td></tr>
<tr><td>Reentrancy:</td><td>Non-Reentrant</td></tr>
<tr><td>Parameters:</td><td>None</td></tr>
<tr><td>Return:</td><td>None</td></tr>
</table></td></tr>
</table>

<table>
<tr><td></td><td></td><td><table><tr><td>Description:</td><td>Turn OFF lights</td></tr></table></td></tr>
<tr><td>DIO</td><td>Dio_Init
Dio_WriteChannel
Dio_ReadChannel</td><td><table><tr><td>Syntax:</td><td>ERROR_STATE Dio_Init (void);</td></tr><tr><td>Sync/Async:</td><td>Synchronous</td></tr><tr><td>Reentrancy:</td><td>Non-Reentrant</td></tr><tr><td>Parameters:</td><td>None</td></tr><tr><td>Return:</td><td>ERROR_STATE</td></tr><tr><td>Description:</td><td>Initialize Dio pins with required configuration</td></tr></table>
<table><tr><td>Syntax:</td><td>void Dio_WriteChannel (DioPinLevel value , DioPinType id);</td></tr><tr><td>Sync/Async:</td><td>Synchronous</td></tr><tr><td>Reentrancy:</td><td>Non-Reentrant</td></tr><tr><td>Parameters:</td><td>Value (pin value) , pin id</td></tr><tr><td>Return:</td><td>None</td></tr><tr><td>Description:</td><td>Write value on pin</td></tr></table>
<table><tr><td>Syntax:</td><td>DioPinLevel Dio_ReadChannel (DioPinType id);</td></tr><tr><td>Sync/Async:</td><td>Synchronous</td></tr><tr><td>Reentrancy:</td><td>Non-Reentrant</td></tr><tr><td>Parameters:</td><td>Id (pin id)</td></tr><tr><td>Return:</td><td>Pin value</td></tr><tr><td>Description:</td><td>Read value on pin</td></tr></table></td></tr>
<tr><td>Timer</td><td>Timer_Init
Timer_Start</td><td><table><tr><td>Syntax:</td><td>ERROR_STATE Timer_Init (void);</td></tr><tr><td>Sync/Async:</td><td>Synchronous</td></tr><tr><td>Reentrancy:</td><td>Non-Reentrant</td></tr><tr><td>Parameters:</td><td>None</td></tr><tr><td>Return:</td><td>ERROR_STATE</td></tr><tr><td>Description:</td><td>Initialize pins connected to timer internally , and initialize mode and other configurations</td></tr></table>
<table><tr><td>Syntax:</td><td>void Timer_start (TimerTickType ticks);</td></tr><tr><td>Sync/Async:</td><td>Synchronous</td></tr><tr><td>Reentrancy:</td><td>Non-Reentrant</td></tr><tr><td>Parameters:</td><td>Ticks</td></tr><tr><td>Return:</td><td>None</td></tr><tr><td>Description:</td><td>Make timer count till count = ticks to achieve required periodicity for sending status messages</td></tr></table></td></tr>
<tr><td>CAN</td><td>CAN_Init
CAN_Receive</td><td><table><tr><td>Syntax:</td><td>ERROR_STATE CAN_Init (void);</td></tr></table></td></tr>
</table>

		Sync/Async:	Synchronous
		Reentrancy:	Non-Reentrant
		Parameters:	None
		Return:	ERROR_STATE
		Description:	Initialize CAN with required configuration
		Syntax:	Uint64 CAN_receive (CanPinType canPinid);
		Sync/Async:	Synchronous
		Reentrancy:	Non-Reentrant
		Parameters:	CAN pin id
		Return:	data
		Description:	receive data through CAN

4. Typedefs

typedef unsigned char uint8	
typedef unsigned long long uint64	used because max width of data in CAN frame is 64 bits
typedef uint8 DioPinLevel	hold value of pin either HIGH = 1 or LOW = 0 , defined in Dio.h
typedef uint8 CanPinType	defined in CAN.h to specify the id of CAN pin
typedef uint8 DioPinType	hold value of pin id , defined in Dio.h