**Automotive Door Control**

**System Design (ECU1)**

**Std**

**Types**

**Mcal Layer**

**On-Board Layer**

**Application Layer**

Speed Status

Switch Status

Door Sensor

Light Switch

Speed Sensor

**Free**

**RTOS**

Door Status

**Common**

**Macros**

CAN

ADC

DIO

**ECU1 has 3 abstraction layers: Application, Onboard and Mcal layers.**

For application layer, it contains 3 main tasks:

1. Door Status, which check the status of the door every 10 ms (opened or closed) and send to ECU 2.
2. Switch Status, which check the status of the switch every 20 ms (pressed or released) and send to ECU 2.
3. Speed Status, which check the status of the car every 5 ms (moved or stopped) and send to ECU 2.

For On-Board layer, it contains 3 main tasks:

1. Speed Sensor, which measure the speed of the car if it moves or not.
2. Light Switch, which is the output of the action according to the pressed switch or not.
3. Door Sensor, which check the door if its opened or closed for safety.

For Mcal layer, it contains 3 main tasks:

1. ADC, which responsible for converting the analog signal to digital for microcontroller.
2. DIO, which responsible for logic output and input (switch and door status).
3. CAN, which responsible for communicating with another ECU and give the status of the components.

For APIs, that will be used in the Projects:

1. Speed Sensor:
2. Light Switch:
3. Door Sensor:
4. DIO:
5. ADC:
6. CAN:

APIs Fully Detailed Description:

1. Speed Sensor:

Must include ADC Driver.

API Arguments:

|  |  |
| --- | --- |
| Name | SpeedSensor |
| Type | Enum |
| Range | 0 for SpeedSensor\_Off |
| 1 for SpeedSensor\_On |
| Description | Describe if the pin is high or low. |

API Functions:

void Speed\_Init (void)

|  |  |
| --- | --- |
| Name | Speed Init |
| API Type | Init |
| Arguments | void |
| Return Type | void |
| Description | that responsible for initialization of the sensor. |

SpeedSensor Speed\_Status (uint8 Ch-Num);

|  |  |  |
| --- | --- | --- |
| Name | Speed Status | |
| API Type | Getter |  |
| Arguments | Uint8 | Ch-Num |
|  | Select the channel of connected sensor | |
| Return Type | Speed Sensor | |
| Description | detect the variable voltage that obtained by the sensor. | |

1. Light Switch:

Must include DIO Driver.

API Arguments:

|  |  |
| --- | --- |
| Name | Lightswitch |
| Type | Enum |
| Range | 0 for LightSwitch\_Off |
| 1 for LightSwitch\_On |
| Description | Describe if the pin is high or low. |

API Functions:

void Switch\_Init (void);

|  |  |
| --- | --- |
| Name | Switch Init |
| API Type | Init |
| Arguments | void |
| Return Type | void |
| Description | that responsible for initialization of the switch. |

LightSwitch Switch\_Status (Port PortNo , PinID PinNo, Pin PinStatus);

|  |  |  |
| --- | --- | --- |
| Name | Switch Status | |
| API Type | Getter |  |
| Arguments | Port | PortNo |
| State which port is used | |
| PinID | PinNo |
| State which pin is used | |
| Pin | PinStatus |
| State the value of Pin (High or Low) | |
| Return Type | LightSwitch | |
| Description | Detect if switch is pressed or released | |
|  |  | |

1. Door Sensor:

Must include DIO Driver.

API Arguments:

|  |  |
| --- | --- |
| Name | DoorSensor |
| Type | Enum |
| Range | 0 for DoorSensor\_Off |
| 1 for DoorSensor\_On |
| Description | Describe if the pin is high or low. |

API Functions:

void Door\_Init (void);

|  |  |
| --- | --- |
| Name | Door Init |
| API Type | Init |
| Arguments | void |
| Return Type | void |
| Description | that responsible for initialization of the sensor. |

DoorSensor Door\_Status (Port PortNo , PinID PinNo, Pin PinStatus);

|  |  |  |
| --- | --- | --- |
| Name | Door Status | |
| API Type | Getter |  |
| Arguments | Port | PortNo |
| State which port is used | |
| PinID | PinNo |
| State which pin is used | |
| Pin | PinStatus |
| State the value of Pin (High or Low) | |
| Return Type | DoorSensor | |
| Description | detect the door if its closed or opened. | |

1. DIO:

API Arguments:

|  |  |
| --- | --- |
| Name | Pin |
| Type | Enum |
| Range | 0 for PIN\_IS\_LOW |
| 1 for PIN\_IS\_HIGH |
| Description | Describe if the pin is high or low. |

|  |  |
| --- | --- |
| Name | Port |
| Type | Enum |
| Range |
| Description | Describe which port is used. |

|  |  |
| --- | --- |
| Name | PINNo |
| Type | Enum |
| Range | 0 to 7 according to the No. of Pins Connected to Port ( PIN0, PIN1,…) |
| Description | Describe which port is used. |

|  |  |
| --- | --- |
| Name | DIO\_ConfigType |
| Type | Structure |
| Range | Uint8 |
| Description | Contain all configuration used to initialize the DIO port correctly. A pointer to structure is passed to the function with all information it needs. |

**API Functions:**

void DIO\_Init (DIO\_ConfigType \* ConfigStruct);

|  |  |
| --- | --- |
| Name | DIO Init |
| API Type | Init |
| Arguments | DIO\_ConfigType \* ConfigStruct |
| Structure for all configuration |
| Return Type | void |
| Description | initialize the DIO port with clock and determine which is input and output. |

Pin DIO\_Read ( Port PortNo , PinID PinNo);

|  |  |  |
| --- | --- | --- |
| Name | DIO Read | |
| API Type | Getter |  |
| Arguments | Port | PortNo |
| State which port is used. | |
| PinID | PinNo |
| State which Pin used to get Data | |
| Return Type | Pin | |
| Description | responsible for reading the status of the pin. | |

void DIO\_Write ( Port PortNo , PinID PinNo, Pin PinStatus );

|  |  |  |
| --- | --- | --- |
| Name | DIO Write | |
| API Type | Setter |  |
| Arguments | Port | PortNo |
| State which port is used | |
| PinID | PinNo |
| State which pin is used | |
| Pin | PinStatus |
| State the value of Pin (High or Low) | |
| Return Type | void | |
| Description | responsible for write on the pin for output. | |

1. ADC:

API Arguments:

|  |  |
| --- | --- |
| Name | ADC\_ConfigType |
| Type | Structure |
| Range | Uint8 |
| Description | Contain all configuration used to initialize the ADC correctly. A pointer to structure is passed to the function with all information it needs. |

|  |  |
| --- | --- |
| Name | ADC\_Prescaler |
| Type | Uint8 |
| Range |  |
| Description | Define the prescaler used for ADC to work Properly. |

|  |  |
| --- | --- |
| Name | ADC\_refvoltage |
| Type | Uint8 |
| Range |  |
| Description | Define the reference voltage to set the resolution of the ADC. |

API Functions:

void ADC\_Init (ADC\_ConfigType \* ConfigStruct);

|  |  |  |
| --- | --- | --- |
| Name | ADC Init | |
| API Type | Init |  |
| Arguments | ADC\_ConfigType \* | ConfigStruct |
|  | Determine the whole data needed to initialize the ADC. | |
| Return Type | void | |
| Description | : initialize the ADC with suitable ref. voltage and prescaler | |

Uint32 ADC\_ReadPolling (uint8 Ch-Num);

|  |  |  |
| --- | --- | --- |
| Name | ADC ReadPolling | |
| API Type | Getter |  |
| Arguments | Uint8 | Ch-Num |
|  | State which channel needed to get data from | |
| Return Type | Uint32 | |
| Description | determine which channel to read from and polling until get the result of the conversion. | |

Void ADC\_ReadInterrupt (uint8 Ch-Num);

|  |  |  |
| --- | --- | --- |
| Name | ADC ReadInterrupt | |
| API Type | Getter |  |
| Arguments | Uint8 | Ch-Num |
|  | State which channel needed to get data from | |
| Return Type | void | |
| Description | determine which channel to read from and get the data from ISR. | |

1. CAN:

API Arguments:

|  |  |
| --- | --- |
| Name | CAN\_ConfigType |
| Type | Structure |
| Range | Uint8 |
| Description | Contain all configuration used to initialize the CAN correctly. A pointer to structure is passed to the function with all information it needs. |

|  |  |
| --- | --- |
| Name | CANSTATUS |
| Type | Enum |
| Range | 0 for PdFalse |
| 1 for PdTrue |
| Description | Return the status of the data sent. |

API Functions:

void CAN\_Init (CAN\_Config \* ConfigStruct);

|  |  |  |
| --- | --- | --- |
| Name | CAN Init | |
| Arguments | CAN\_Config \* | ConfigStruct |
|  | Determine the whole data needed to initialize the CAN. | |
| Return Type | void | |
| Description | initialize the CAN protocol for communication. | |

CANStatus CAN\_SendData (uint32 Data);

|  |  |  |
| --- | --- | --- |
| Name | CAN SendData | |
| Arguments | Uint32 | Data |
|  | Contain data needed to be sent via CAN | |
| Return Type | CANStatus | |
| Description | responsible for encoding, send data and check if it’s completely sent. | |

Uint32 CAN\_ReceiveData ( void );

|  |  |
| --- | --- |
| Name | CAN ReceiveData |
| Arguments | void |
| Return Type | Uint32 |
| Description | responsible for encoding, send data and check if it’s completely sent. |