# Door Control Module GOC Motors Automotive

Generic OEM Motor has requested a New ECU that will be used o their car to control doors on vehicle. This system will control DriverDoor, PassengerDoor, RearRight Door and RearLeft Door.

Main functionalities for the system include:

* Door Locking
* Window Control
* HW Diagnostics

Details for requirements are described later on this document.

The system shall not cost more than 9 USD (U.S. Dollars) per unit.

The supplier shall be capable to produce 100,000 units per year.

The systems shall assure system proper behavior for at least 10 years.

The supplier shall produce the system for 5 years consecutive with the possibility to extend this period.

## System Description

**System** consists of 4 Different ECUs that are placed on the 4 doors of the vehicle.

* Driver Door
* Passenger Door
* Rear Right Door
* Rear Left Door.

This **system** will have Human Machine Interfaces that will interact with the user to control the system for **Door Locking** and **Window Control** functionalities.

The **system** shall be able to perform **Window Control** operation either Manual Mode or **CAN network** Request per individual Door.

The **system** shall be capable to **Determine Window Position** and **Report Window Position** **on CAN network** per individual Door.

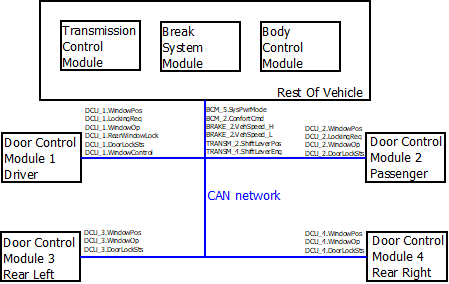
**Driver Door** shall be capable to request **Window Control** Operation on **CAN network** when user request via **HMI**.

The **system** shall be capable to **Determine Door Lock Position** and **Report Door Lock Position** on **CAN network** per individual Door.

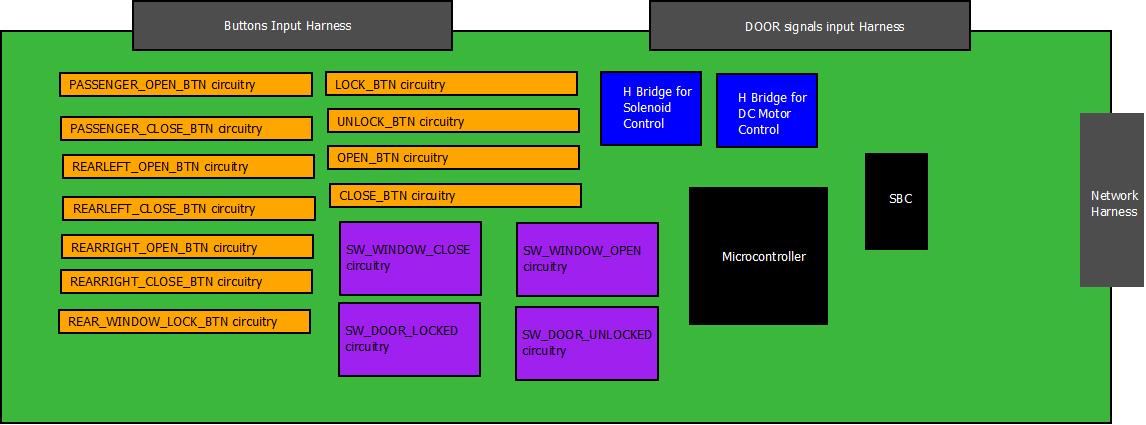
The system shall be able to attend **Door Locking** Operation requests via **CAN network** per individual Door.

**Driver Door** and **Passenger Door** shall be capable to request **Door Locking** Operation on **CAN network** when user request via **HMI**.

**Driver Door** shall be able to report **Block Rear Window** operation on **CAN network.**

****

The **system** will interact with the rest of the vehicle via **CAN network**. Details for CAN frames are described inside “Appendix CAN Data Base” inside this document.



Safety Goal

Sentence: “**Window Control** shall be capable to detect window obstacles while closing and stop operation”

Rational: Body parts can be pinch by the window when it is closing. The primary operation is to detect those obstacles in order to trigger a **CANCEL\_WINDOW\_ACTUATION** during a **CLOSE\_WINDOW\_ACTUATION** on the corresponding window.

Emergency operation: Perform **OPEN\_WINDOW\_ACTUATION** after **CLOSE\_WINDOW\_ACTUATION** on corresponding window**.**

ASILB

FTTI 100ms

## Functionalities

### Door Locking

**Door locking** function is used to control the corresponding Door Solenoid that is used to lock and unlock the corresponding Door.

#### LOCK\_DOOR\_ACTUATION

This function is executed when the operation to lock the door is executed.

Sequence for actuation shall be:

1. Turn off UNLOCK\_LED
2. Hold States during 100ms
3. Turn on LOCK\_LED



#### UNLOCK\_DOOR\_ ACTUATION

This function is executed when the operation to lock the door is executed.

Sequence for actuation shall be:

1. Turn off LOCK\_LED
2. Hold States during 100ms
3. Turn on UNLOCK\_LED



#### Manual Mode

**Driver Door** and **Passenger Door** are allowed to execute Door locking Manual Mode.

**LOCK\_DOOR\_ACTUATION** will be executed when **LOCK\_BTN** of the **DCU** transitions from **BTN\_NOT\_PRESSED** to **BTN\_PRESSED**.

**UNLOCK\_DOOR\_ACTUATION** will be executed when **UNLOCK\_BTN** of the **DCU** transitions from **BTN\_NOT\_PRESSED** to **BTN\_PRESSED**.

If both Buttons **LOCK\_BTN** and **UNLOCK\_BTN** are pressed then the operation shall not be executed.

If both Buttons **LOCK\_BTN** and **UNLOCK\_BTN** are not pressed then no action shall be done.

#### Remote operation

All Doors shall execute Door locking for Remote Operation from **BCM** request via **CAN network** if the message is authenticated from the correspondign source**.**

**LOCK\_DOOR\_ACTUATION** will be executed when signal **BCM\_2.ConfortCmd** transitions from **No Cmd** to **LockCmd**.

**UNLOCK\_DOOR\_ACTUATION** will be executed when signal **BCM\_2.ConfortCmd** transitions from **No Cmd** to **UnlockAllCmd** or transitions from **UnlockDrvrCmd** to **UnlockAllCmd.**

For **Driver Door** only **UNLOCK\_DOOR\_ACTUATION** will be executed when signal **BCM\_2.ConfortCmd** transitions from **No Cmd** to **UnlockDrvrCmd**.

Depending on the Door Locking operations shall be executed from requests from **Driver Door** or **Passenger Door**.

Passenger Door shall execute **LOCK\_DOOR\_ACTUATION** when signal **DCU\_1.LockingReq** transitions from **NO\_LOCKING\_REQ** to **LOCK\_REQ**.

Passenger Door shall execute **UNLOCK\_DOOR\_ACTUATION** when signal **DCU\_1.LockingReq** transitions from **NO\_LOCKING\_REQ** to **UNLOCK\_REQ**.

RearLeft Door shall execute **LOCK\_DOOR\_ACTUATION** when signal **DCU\_1.LockingReq** transitions from **NO\_LOCKING\_REQ** to **LOCK\_REQ**.

RearLeft Door shall execute **UNLOCK\_DOOR\_ACTUATION** when signal **DCU\_1.LockingReq** transitions from **NO\_LOCKING\_REQ** to **UNLOCK\_REQ**.

RearRightDoor shall execute **LOCK\_DOOR\_ACTUATION** when signal **DCU\_1.LockingReq** transitions from **NO\_LOCKING\_REQ** to **LOCK\_REQ**.

RearRightDoor shall execute **UNLOCK\_DOOR\_ACTUATION** when signal **DCU\_1.LockingReq** transitions from **NO\_LOCKING\_REQ** to **UNLOCK\_REQ**.

RearLeft Door shall execute **LOCK\_DOOR\_ACTUATION** when signal **DCU\_2.LockingReq** transitions from **NO\_LOCKING\_REQ** to **LOCK\_REQ**.

RearLeft Door shall execute **UNLOCK\_DOOR\_ACTUATION** when signal **DCU\_2.LockingReq** transitions from **NO\_LOCKING\_REQ** to **UNLOCK\_REQ**.

RearRightDoor shall execute **LOCK\_DOOR\_ACTUATION** when signal **DCU\_2.LockingReq** transitions from **NO\_LOCKING\_REQ** to **LOCK\_REQ**.

RearRightDoor shall execute **UNLOCK\_DOOR\_ACTUATION** when signal **DCU\_2.LockingReq** transitions from **NO\_LOCKING\_REQ** to **UNLOCK\_REQ**.

### Door Lock Status Determination

**DOOR\_LOCKED**

Door shall be considered as locked if **SW\_DOOR\_LOCKED** is determined as **SW\_ACTIVE**.



**DOOR\_UNLOCKED**

Door shall be considered as Unlocked if **SW\_DOOR\_UNLOCKED** is determined as **SW\_ACTIVE.**

**ERROR**



Door Lock shall be considered as **ERROR** if **SW\_DOOR\_LOCKED** and **SW\_DOOR\_UNLOCKED** are determined as **SW\_ACTIVE** or if **SW\_DOOR\_LOCKED and SW\_DOOR\_UNLOCKED are determined as SW\_INACTIVE.**

### Door Lock Status Report

For **Driver Door,** itshall report the determined Lock Status via **DCU\_1.DoorLockSts.**

For **Passenger Door,** itshall report the determined Lock Status via **DCU\_2.DoorLockSts.**

For **RearLeft Door,** itshall report the determined Lock Status via **DCU\_3.DoorLockSts.**

For **RearRight Door,** itshall report the determined Lock Status via **DCU\_4.DoorLockSts.**

### Auto Lock While Driving

If at least one of the **DoorLockSts (from DCU\_1, DCU\_2, DCU\_3 and DCU\_4)** is equal to **DOOR\_UNLOCK** and **Vehicle Speed** indicates equal or greater than 20km/h and **System Power Mode** is equal to **RUN**  then **Driver Door**  shall report command **LockingReq** as **LOCK\_REQ.**  This request will be received by **BCM** and reported as **ConfortCmd** with value **LockCmd**. This complete sequence shall not take more than 500ms since **Vehicle Speed** and **System Power Mode** are met.

### Window Control

**Window Control** function is used to control the corresponding **Window DC motor** that is used to Open and Close the corresponding Window.

If there is not a Window Control Actuation then ECU shall report IDLE Window operation on the corresponding CAN frame.

Driver Door shall report **DCU\_1.WindowOp** as **WINDOW\_IDLE.**

Passenger Door shall report **DCU\_2.WindowOp** as **WINDOW\_IDLE.**

RearLeft Door shall report **DCU\_3.WindowOp** as **WINDOW\_IDLE.**

RearRight Door shall report **DCU\_4.WindowOp** as **WINDOW\_IDLE.**

#### OPEN\_WINDOW\_ACTUATION

**OPEN\_WINDOW\_ACTUATION** shall be executed only when **WINDOW\_POSITION** is different than **COMPLETELY\_OPEN** and **WINDOW\_POSITION** is different than **ERROR**.

This function is executed when there are transitions from Picture in descendant order. There is a 500ms delay between every transition from one picture to another. Sequence will stop when conditions are no longer present.

During **OPEN\_WINDOW\_ ACTUATION** each ECU shall report its window operation on the corresponding CAN frame.

Driver Door shall report **DCU\_1.WindowOp** as **WINDOW\_DOWN.**

Passenger Door shall report **DCU\_2.WindowOp** as **WINDOW\_ DOWN.**

RearLeft Door shall report **DCU\_3.WindowOp** as **WINDOW\_ DOWN.**

RearRight Door shall report **DCU\_4.WindowOp** as **WINDOW\_ DOWN.**

#### GLOBAL\_OPEN\_WINDOW\_ACTUATION

**GLOBAL\_OPEN\_WINDOW\_ACTUATION** shall be executed only when **WINDOW\_POSITION** is different than **COMPLETELY\_OPEN** and **WINDOW\_POSITION** is different than **ERROR**.

This function is executed when there are transitions from Picture in descendant order. There is a 500ms delay between every transition from one picture to another. Sequence will stop when window is considered as **COMPLETELY\_OPEN**.

During **GLOBAL\_OPEN\_WINDOW\_ ACTUATION** each ECU shall report its window operation on the corresponding CAN frame.

Driver Door shall report **DCU\_1.WindowOp** as **WINDOW\_DOWN.**

Passenger Door shall report **DCU\_2.WindowOp** as **WINDOW\_ DOWN.**

RearLeft Door shall report **DCU\_3.WindowOp** as **WINDOW\_ DOWN.**

RearRight Door shall report **DCU\_4.WindowOp** as **WINDOW\_ DOWN.**

#### CLOSE\_WINDOW\_ ACTUATION

**GLOBAL\_CLOSE\_WINDOW\_ACTUATION** shall be executed only when **WINDOW\_POSITION** is different from **COMPLETELY\_CLOSE** and **WINDOW\_POSITION** is different than **ERROR**.

This function is executed when there are transitions from Picture in ascendant order. There is a 500ms delay between every transition from one picture to another. Sequence will stop when conditions are no longer present.

During **CLOSE\_WINDOW\_ ACTUATION** each ECU shall report its window operation on the corresponding CAN frame.

Driver Door shall report **DCU\_1.WindowOp** as **WINDOW\_UP.**

Passenger Door shall report **DCU\_2.WindowOp** as **WINDOW\_UP.**

RearLeft Door shall report **DCU\_3.WindowOp** as **WINDOW\_UP.**

RearRight Door shall report **DCU\_4.WindowOp** as **WINDOW\_UP.**

#### GLOBAL\_CLOSE\_WINDOW\_ ACTUATION

**CLOSE\_WINDOW\_ACTUATION** shall be executed only when **WINDOW\_POSITION** is different from **COMPLETELY\_CLOSE** and **WINDOW\_POSITION** is different than **ERROR**.

This function is executed when there are transitions from Picture in ascendant order. There is a 500ms delay between every transition from one picture to another. Sequence will stop when window is considered as **COMPLETELY\_CLOSED**.

During **GLOBAL\_CLOSE\_WINDOW\_ ACTUATION** each ECU shall report its window operation on the corresponding CAN frame.

Driver Door shall report **DCU\_1.WindowOp** as **WINDOW\_UP.**

Passenger Door shall report **DCU\_2.WindowOp** as **WINDOW\_UP.**

RearLeft Door shall report **DCU\_3.WindowOp** as **WINDOW\_UP.**

RearRight Door shall report **DCU\_4.WindowOp** as **WINDOW\_UP.**

#### CANCEL\_WINDOW\_ACTUATION

**CANCEL\_WINDOW\_ACTUATION** shall be executed to move the **Window Control** to IDLE. This actuation can be executed only when there is a Window control Actuation On Going and it abort current actuation.

This function is executed when all the LED stop keeping current position.

#### Manual Mode

##### Short Button Press

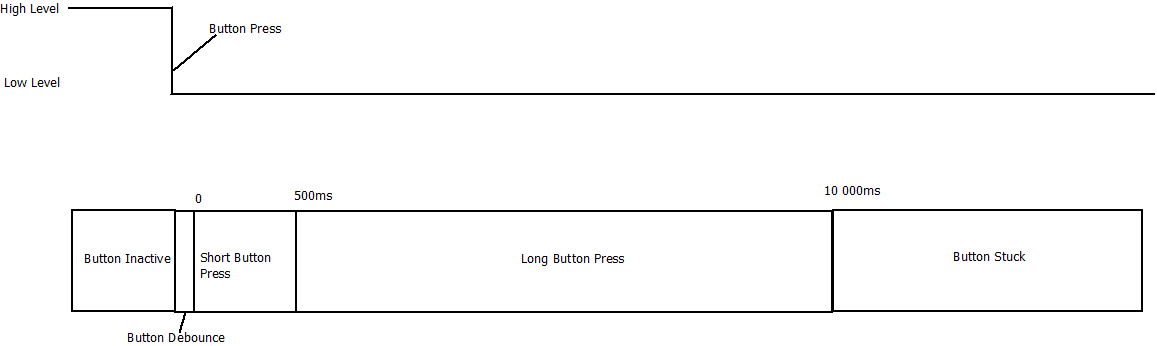
If a Button transitions from **BTN\_NOT\_PRESSED** to **BTN\_PRESSED** and **BTN\_NOT\_PRESSED** within a time <= 500ms then it shall be considered as a **SHORT\_BTN\_PRESS.**

##### Long Button Press

If a Button transitions from **BTN\_NOT\_PRESSED** to **BTN\_PRESSED** and **BTN\_NOT\_PRESSED** within a time > 500ms and <=10000ms then it shall be considered as a **LONG\_BTN\_PRESS.**

##### Button Stuck

If a Button transitions from **BTN\_NOT\_PRESSED** to **BTN\_PRESSED** and **BTN\_NOT\_PRESSED** within a time > 10000ms then it shall be considered as a **BTN\_STUCK.**

****

**All Doors** are allowed to execute manual **Window Control** from its corresponding **OPEN\_BTN** and **CLOSE\_BTN** states**.**

**Driver Door** and **Passenger Door** shall execute **GLOBAL\_OPEN\_WINDOW\_ACTUATION** when its **OPEN\_BTN** state is equal to **SHORT\_BTN\_PRESS.**

**Driver Door** and **Passenger Door** shall execute **GLOBAL\_CLOSE\_WINDOW\_ACTUATION** when its **CLOSE\_BTN** state is equal to **SHORT\_BTN\_PRESS.**

**Driver Door** and **Passenger Door** shall execute **OPEN\_WINDOW\_ACTUATION** when its **OPEN\_BTN** state is equal to **LONG\_BTN\_PRESS**.

**Driver Door** and **Passenger Door** shall execute **CLOSE\_WINDOW\_ACTUATION** when its **CLOSE\_BTN** state is equal to **LONG\_BTN\_PRESS**.

**RearLeft Door** and **RearRight Door** shall execute **GLOBAL\_OPEN\_WINDOW\_ACTUATION** when its **OPEN\_BTN** state is equal to **SHORT\_BTN\_PRESS** and **DCU\_1.RearWindowLock** is equal to REAR\_WINDOW\_UNBLOCK.

**RearLeft Door** and **RearRight Door** shall execute **GLOBAL\_CLOSE\_WINDOW\_ACTUATION** when its **CLOSE\_BTN** state is equal to **SHORT\_BTN\_PRESS** and **DCU\_1.RearWindowLock** is equal to REAR\_WINDOW\_UNBLOCK.

**RearLeft Door** and **RearRight Door** shall execute **OPEN\_WINDOW\_ACTUATION** when its **OPEN\_BTN** state is equal to **LONG\_BTN\_PRESS** and **DCU\_1.RearWindowLock** is equal to REAR\_WINDOW\_UNBLOCK.

**RearLeft Door** and **RearRight Door** shall execute **CLOSE\_WINDOW\_ACTUATION** when its **CLOSE\_BTN** state is equal to **LONG\_BTN\_PRESS** and **DCU\_1.RearWindowLock** is equal to REAR\_WINDOW\_UNBLOCK.

**Driver Door** shall report **WINDOW\_UP\_REQ** on **WindowControl\_Passenger** bits positionwhen **PASSENGER\_OPEN\_BTN** state is equal to **SHORT\_BTN\_PRESS**  or **LONG\_BTN\_PRESS.**

**Driver Door** shall report **WINDOW\_DOWN\_REQ** on **WindowControl\_** **Passenger** positionwhen **PASSENGER \_CLOSE\_BTN** state is equal to **SHORT\_BTN\_PRESS**  or **LONG\_BTN\_PRESS.**

**Driver Door** shall report **WINDOW\_UP\_REQ** on **WindowControl\_RearLeft** bits positionwhen **REARLEFT\_OPEN\_BTN** state is equal to **SHORT\_BTN\_PRESS**  or **LONG\_BTN\_PRESS.**

**Driver Door** shall report **WINDOW\_DOWN\_REQ** on **WindowControl\_** **RearLeft** positionwhen **REARLEFT \_CLOSE\_BTN** state is equal to **SHORT\_BTN\_PRESS**  or **LONG\_BTN\_PRESS.**

**Driver Door** shall report **WINDOW\_UP\_REQ** on **WindowControl\_RearRight** bits positionwhen **REARRIGHT\_OPEN\_BTN** state is equal to **SHORT\_BTN\_PRESS**  or **LONG\_BTN\_PRESS.**

**Driver Door** shall report **WINDOW\_DOWN\_REQ** on **WindowControl\_** **RearRight** positionwhen **REARRIGHT \_CLOSE\_BTN** state is equal to **SHORT\_BTN\_PRESS**  or **LONG\_BTN\_PRESS.**

**Driver Door** shall report **REAR\_WINDOW\_BLOCK** on **DCU\_1. RearWindowLock** bits positionwhile **REAR\_WINDOW\_LOCK\_BTN** state is equal to **BTN\_PRESSED.**

**Driver Door** shall report **REAR\_WINDOW\_UNBLOCK** on **DCU\_1. RearWindowLock** bits positionwhile **REAR\_WINDOW\_LOCK\_BTN** state is equal to **BTN\_NOT\_PRESSED.**

#### Remote Operation

All Doors are allowed to execute Door locking for Remote Operation from **BCM** request via **CAN network.**

**OPEN\_WINDOW\_ACTUATION** will be executed when signal **BCM\_2.ConfortCmd** is received consecutively at least during 500ms **UnlockAllCmd.**

**CANCEL\_WINDOW ACTUATION** will be executed when **BCM\_2.ConfortCmd** transitions to **No Cmd** or **WINDOW\_POSITION** is equal to **COMPLETELY\_OPEN** during an **OPEN\_WINDOW\_ACTUATION.**

**CLOSE\_WINDOW\_ACTUATION** will be executed when signal **BCM\_2.ConfortCmd** is received consecutively at least during 500ms **LockCmd.**

**CANCEL\_WINDOW ACTUATION** will be executed when **BCM\_2.ConfortCmd** transitions to **No Cmd** or **WINDOW\_POSITION** is equal to **COMPLETELY\_CLOSE** during an **CLOSE\_WINDOW\_ACTUATION.**

**Passenger Door** is allowed to execute Door locking for Remote Operation from **DCU 1** request via **CAN network.**

**OPEN\_WINDOW\_ACTUATION** will be executed on **Passenger Door** when signal **DCU\_1.** **WindowControl** is received with value **WINDOW\_DOWN\_REQ** on **WindowControl\_Passenger** bits position.

**CANCEL\_WINDOW\_ACTUATION** will be executed when signal **DCU\_1.** **WindowControl** is received with value **WINDOW\_NO\_REQ** on **WindowControl\_Passenger** bits position or Passenger Door **WINDOW\_POSITION** is equal to **COMPLETELY\_OPEN** during an **OPEN\_WINDOW\_ACTUATION.**

**CLOSE\_WINDOW\_ACTUATION** will be executed on **Passenger Door** when signal **DCU\_1.** **WindowControl** is received with value **WINDOW\_UP\_REQ** on **WindowControl\_Passenger** bits position.

**CANCEL\_WINDOW\_ACTUATION** will be executed when signal **DCU\_1.** **WindowControl** is received with value **WINDOW\_NO\_REQ** on **WindowControl\_Passenger** bits position or Passenger Door **WINDOW\_POSITION** is equal to **COMPLETELY\_CLOSE** during an **OPEN\_WINDOW\_ACTUATION.**

**RearLeft Door** is allowed to execute Door locking for Remote Operation from **DCU 1** request via **CAN network.**

**OPEN\_WINDOW\_ACTUATION** will be executed on **RearLeft Door** when signal **DCU\_1.** **WindowControl** is received with value **WINDOW\_DOWN\_REQ** on **WindowControl\_** **RearLeft** bits position.

**CANCEL\_WINDOW\_ACTUATION** will be executed when signal **DCU\_1.** **WindowControl** is received with value **WINDOW\_NO\_REQ** on **WindowControl\_** **RearLeft** bits position or RearLeft Door **WINDOW\_POSITION** is equal to **COMPLETELY\_OPEN** during an **OPEN\_WINDOW\_ACTUATION.**

**CLOSE\_WINDOW\_ACTUATION** will be executed on **RearLeft Door** when signal **DCU\_1.** **WindowControl** is received with value **WINDOW\_UP\_REQ** on **WindowControl\_** **RearLeft** bits position.

**CANCEL\_WINDOW\_ACTUATION** will be executed when signal **DCU\_1.** **WindowControl** is received with value **WINDOW\_NO\_REQ** on **WindowControl\_** **RearLeft** bits position or RearLeft Door **WINDOW\_POSITION** is equal to **COMPLETELY\_CLOSE** during an **OPEN\_WINDOW\_ACTUATION.**

**RearRight Door** is allowed to execute Door locking for Remote Operation from **DCU 1** request via **CAN network.**

**OPEN\_WINDOW\_ACTUATION** will be executed on **RearRight Door** when signal **DCU\_1.** **WindowControl** is received with value **WINDOW\_DOWN\_REQ** on **WindowControl\_** **RearRight** bits position.

**CANCEL\_WINDOW\_ACTUATION** will be executed when signal **DCU\_1.** **WindowControl** is received with value **WINDOW\_NO\_REQ** on **WindowControl\_** **RearRight** bits position or RearRight Door **WINDOW\_POSITION** is equal to **COMPLETELY\_OPEN** during an **OPEN\_WINDOW\_ACTUATION.**

**CLOSE\_WINDOW\_ACTUATION** will be executed on **RearRight Door** when signal **DCU\_1.** **WindowControl** is received with value **WINDOW\_UP\_REQ** on **WindowControl\_** **RearRight** bits position.

**CANCEL\_WINDOW\_ACTUATION** will be executed when signal **DCU\_1.** **WindowControl** is received with value **WINDOW\_NO\_REQ** on **WindowControl\_** **RearRight** bits position or RearRight Door **WINDOW\_POSITION** is equal to **COMPLETELY\_CLOSE** during an **OPEN\_WINDOW\_ACTUATION.**

#### AntiPinch Operation

**ANTIPINCH\_SIGNAL** is a Digital Input on the systems that reports when an Anti-pinch Event has occurred. **ANTIPINCH\_SIGNAL** has a dedicated instance per Door.

**ANTIPINCH\_SIGNAL** will report the Anti-pich Event using an Analog input 10 bits resolution. Anti pinc event shall be detected If the Analog signal transitions from below the threshold to above the threshold. Use 820 ADC counts as threshold reference.

**ANTIPINCH\_SIGNAL** will report the Anti-pinch Event only during **CLOSE\_WINDOW\_ACTUATION** or **GLOBAL\_CLOSE\_WINDOW\_ACTUATION.**

If **ANTIPINCH\_SIGNAL** is present then an immediate **CANCEL\_WINDOW\_ACTUATION** shall be executed. Then a **GLOBAL\_OPEN\_ACTUATION** shall be executed. After **GLOBAL\_OPEN\_ACTUATION** is finished the **CLOSE\_WINDOW\_ACTUATION** and **GLOBAL\_CLOSE\_WINDOW\_ACTUATION** shall be inhibit during 15 seconds for **Manual Mode** or **Remote Operation** for the corresponding Door.

### Window Position Determination

**WINDOW\_** **COMPLETELY\_OPEN**

Window shall be considered as **COMPLETELY\_OPEN** if **SW\_WINDOW\_OPEN** is determined as **SW\_INACTIVE** and **SW\_WINDOW\_CLOSE** is determined as **SW\_INACTIVE**.

**WINDOW\_** **COMPLETELY\_CLOSE**

Window shall be considered as **COMPLETELY\_CLOSE** if **SW\_WINDOW\_OPEN** is determined as **SW\_ACTIVE** and **SW\_WINDOW\_CLOSE** is determined as **SW\_ACTIVE**.

**WINDOW\_IN\_BETWEEN**

Window shall be considered as **IN\_BETWEEN** if **SW\_WINDOW\_OPEN**  is determined as **SW\_ACTIVE** and **SW\_WINDOW\_CLOSE**  is determined as **SW\_INACTIVE**.

**WINDOW\_ERROR**

Window shall be considered as **IN\_BETWEEN** if **SW\_WINDOW\_CLOSE** is determined as **SW\_ACTIVE** and **SW\_WINDOW\_OPEN** is determined as **SW\_INACTIVE**.



### Window Position Report

For **Driver Door,** itshall report the determined Lock Status via **DCU\_1.** **WindowPos.**

For **Passenger Door,** itshall report the determined Lock Status via **DCU\_2.** **WindowPos.**

For **RearLeft Door,** itshall report the determined Lock Status via **DCU\_3.** **WindowPos.**

For **RearRight Door,** itshall report the determined Lock Status via **DCU\_4.** **WindowPos.**

### Hardware - Software Requirements

#### Button Debounce

In order to use a mechanism to discard glitches on the buttons a debounce mechanism shall be used.

This mechanism implies to monitor periodically a signal and increment counters to mature the state of a signal.

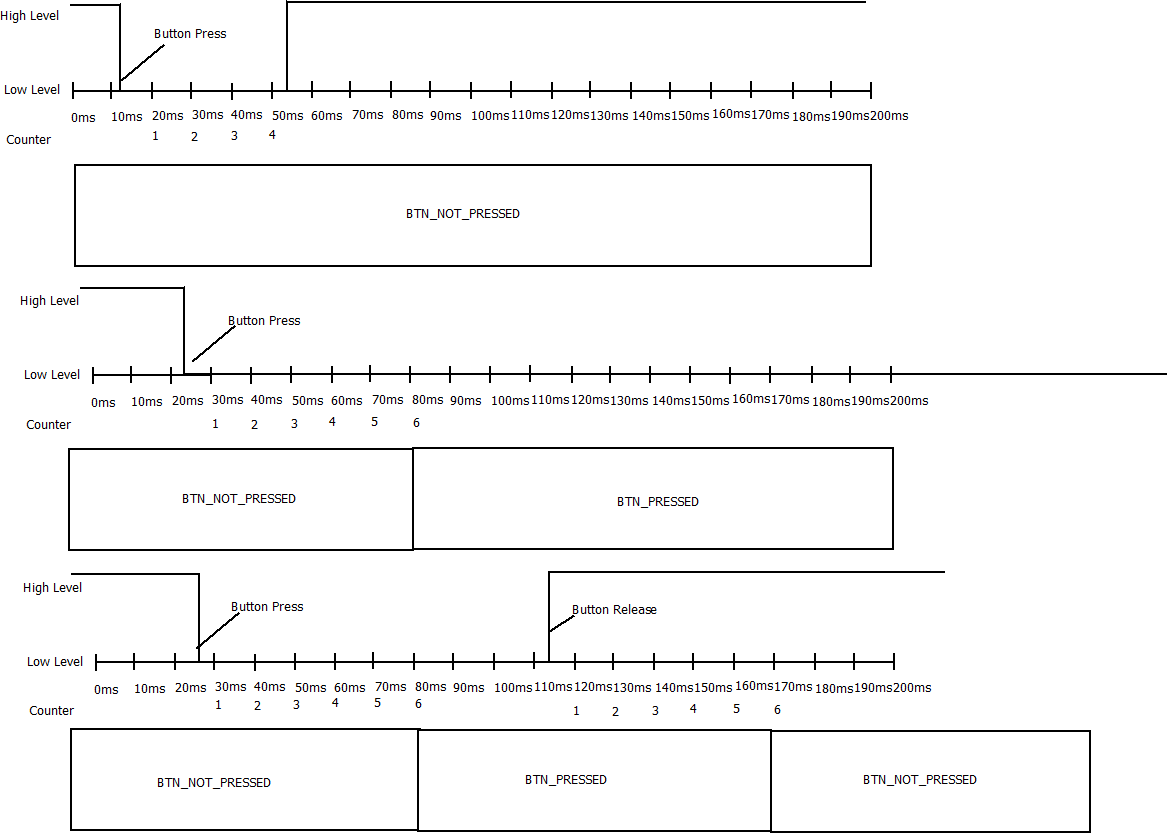
All the buttons used on the system shall use a inverted logic. This means they will be consider as ACTIVE when they are in Low state. Idle state shall be high.

Debounce mechanism consist to increment a counter if the Button State has not change from previous value. A threshold shall be used to indicate when the Button can be consider as matured (**BTN\_PRESSED** ) or Dematured (**BTN\_NOT\_PRESSED**).

Threshold value to determine a **BTN\_PRESSED** will be 50ms (6 counts).

Threshold value to determine a **BTN\_NOT\_ PRESSED** will be 50ms (6 counts).

The Button position will be evaluated periodically every 10ms.



#### Switch position Debounce

Similar to buttons it is required to use a mechanism to discard glitches on the Switches used to determine position.

This mechanism implies to monitor periodically a signal and increment counters to mature the state of a signal.

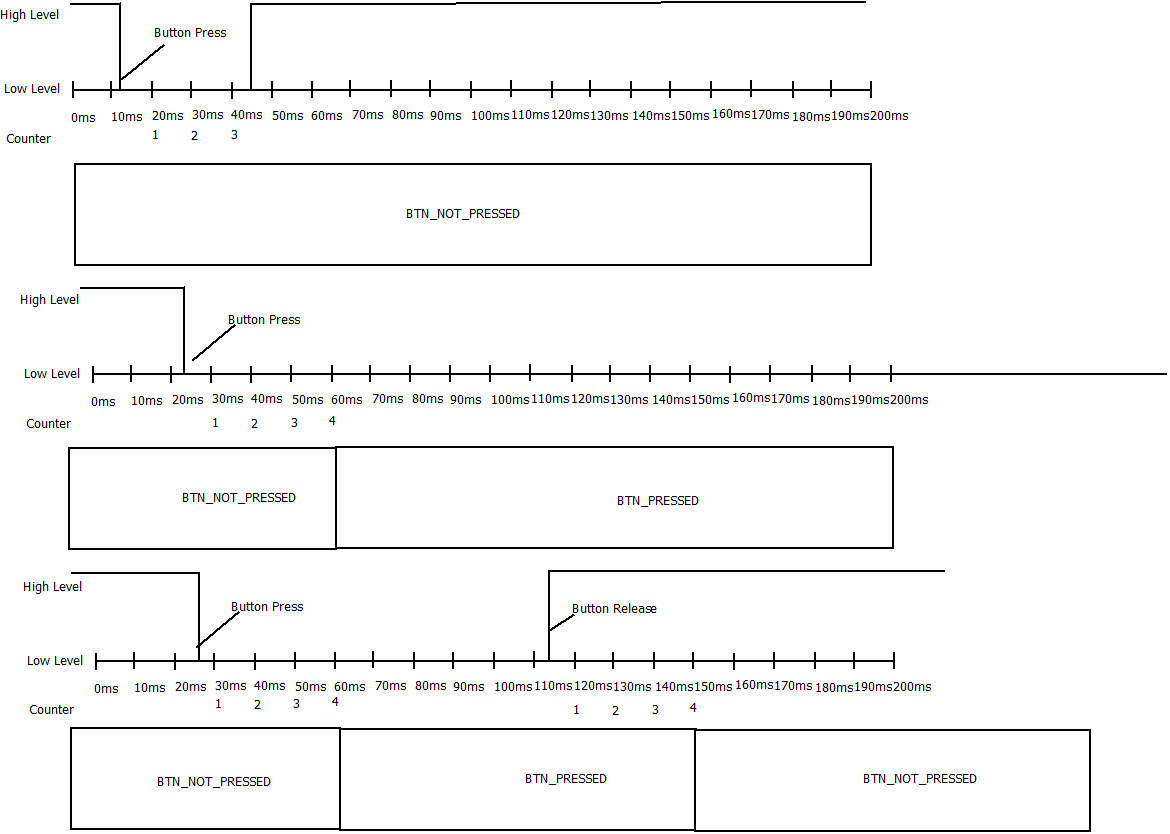
All the buttons used on the system shall use a positive logic. This means they will be consider as ACTIVE when they are in High state. Idle state shall be Low.

Debounce mechanism consist to increment a counter if the Button State has not change from previous value. A threshold shall be used to indicate when the Button can be considered as matured or Dematured.

Threshold value to determine a **BTN\_PRESSED** will be 30ms (4 counts).

Threshold value to determine a **BTN\_NOT\_ PRESSED** will be 30ms (4 counts).

The Switch position will be evaluated periodically every 10ms.



#### Solenoid Control

*Solenoid Behavior will be model with an RGB LED that will be used to indicate when the Door is LOCK or UNLOCK*

RGB will become Green (0,100,0) in order to indicate Door is UNLOCK

RGB will become Red (100,0,0) in order to indicate Door is LOCK

Any other value shall be considered invalid.



#### Window Control

*Window Behavior will be model with an 10 RED LED BAR that will be used to indicate window position and window operations*

It will be controlled using 10 digital outputs and the window actuation will be displayed as an animation with 500ms delay between transitions.

Window animation sequences are described in picture below.



Window Animation to close the window from Completely Open to Completely Close is from Picture 0 to Picture 10.

Window Animation to close the window from Completely Close to Completely Open is from Picture 10 to Picture 0.

### HW Diagnostics

#### Solenoid Error

In case Door Lock is considered as **ERROR** then Door Locking functionality shall be disable until next power cycle. (Transition OFF-> RUN)

#### Button Stuck

Inc ase a Buttin is considered as Stuck, thant button shall be ignore until next power Cycle (Transition OFF-> RUN)

#### Window Error

In case window is considered as **WINDOW\_ERROR** then Window Control functionality shall be disable until next power cycle. (Transition OFF-> RUN)

## System Variants

### SW Variant

There will be only 1 Software variant for all the DCU. The behavior of each one will depends on the configuration.

#### Configuration

SW Variant Configuration will be done through 2 configuration jumpers. Depending on the state of the jumpers is the intended behavior of the DCU.

|  |  |  |
| --- | --- | --- |
| Jumper 0 | Jumper 1 | Variant Behavior |
| 0 | 0 | Driver Door |
| 0 | 1 | Passenger Door |
| 1 | 0 | RearLeft Door |
| 1 | 1 | RearRight Door |

### HW Variant

#### Driver Door HW

Contains:

* Microcontroller
* SBC
* SW\_WINDOW\_CLOSE circuitry
* SW\_WINDOW\_OPEN circuitry
* SW\_DOOR\_LOCKED circuitry
* SW\_DOOR\_UNLOCKED circuitry
* H Bridge for Solenoid Control
* H Bridge for DC Motor Control
* LOCK\_BTN circuitry
* UNLOCK\_BTN circuitry
* OPEN\_BTN circuitry
* CLOSE\_BTN circuitry
* PASSENGER\_OPEN\_BTN circuitry
* PASSENGER \_ CLOSE \_BTN circuitry
* REARLEFT\_OPEN\_BTN circuitry
* REARLEFT\_CLOSE\_BTN circuitry
* REARRIGHT\_OPEN\_BTN circuitry
* REARRIGHT \_CLOSE\_BTN circuitry
* REAR\_WINDOW\_LOCK\_BTN circuitry

#### Passenger Door HW

Contains:

* Microcontroller
* SBC
* SW\_WINDOW\_CLOSE circuitry
* SW\_WINDOW\_OPEN circuitry
* SW\_DOOR\_LOCKED circuitry
* SW\_DOOR\_UNLOCKED circuitry
* H Bridge for Solenoid Control
* H Bridge for DC Motor Control
* LOCK\_BTN circuitry
* UNLOCK\_BTN circuitry
* OPEN\_BTN circuitry
* CLOSE\_BTN circuitry

#### RearDoor HW

Contains:

* Microcontroller
* SBC
* SW\_WINDOW\_CLOSE circuitry
* SW\_WINDOW\_OPEN circuitry
* SW\_DOOR\_LOCKED circuitry
* SW\_DOOR\_UNLOCKED circuitry
* H Bridge for Solenoid Control
* H Bridge for DC Motor Control
* OPEN\_BTN circuitry
* CLOSE\_BTN circuitry

## Appendix CAN Data Base

The system will be a CAN Node of the Comfort Network.

Physical Characteristics:

* Dual Wire (CAN High and CAN Low)
* Baud rate 125 Kbps

### Frame BCM\_5

#### Message Layout

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ID | Byte0 | Byte1 | Byte2 | Byte3 | Byte4 | Byte5 | Byte6 | Byte7 | Period |
| 0x110 | BCM\_5\_MC | x | x | SysPwrMode | BCM\_5\_CMAC | | | | 500ms |

#### Signals Description

**SysPwrMode:** This Signal contains the system power mode that will be used for the ECU to know the Ignition status of the system. Enumeration values are listed below:

* 0x00 (SNA) Signal Not Available.
* 0x01 (OFF) Vehicle is off.
* 0x02 (ACC) Vehicle is in Accessory.
* 0x03 (RUN) Vehicle is in Run Mode.
* 0x04 (CRANK) Vehicle is doing Ignition.
* Values different than this shall be considered as INVALID Data.

**BCM\_5\_MC:** This is the Message counter of the BCM\_5 frame. This signal shall go between 0 and 255 with increments of 1. This signal shall be updated every time the telegram is transmitted. If the signal reaches its limit(255), then the counter value shall initialize as 0.

**BCM\_5\_CMAC**: This is a 32 bytes CMAC used to authenticate the source of this message. If the CMAC is not valid then the content of the whole message shall be ignored.

### Frame BCM\_2

#### Message Layout

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ID | Byte0 | Byte1 | Byte2 | Byte3 | Byte4 | Byte5 | Byte6 | Byte7 | Period |
| 0x252 | BCM\_2\_MC | x | ConfortCmd | x | BCM\_2\_CMAC | | | | 500ms |

#### Signals Description

**ConfortCmd:** This Signal contains the Confort Command and it represents the confort operation for the Vehicle

* 0x00 (No Cmd) No command to execute.
* 0x01 (LockCmd) Represents Lock Command.
* 0x02 (UnlockAllCmd) Represents Unlock Command for all Doors.
* 0x03 (UnlockDrvrCmd) Represents Unlock Command for Driver Door only.
* Values different than this shall be considered as INVALID Data.

**BCM\_2\_MC:** This is the Message counter of the BCM\_2 frame. This signal shall go between 0 and 255 with increments of 1. This signal shall be updated every time the telegram is transmitted. If the signal reaches its limit (255), then the counter value shall initialize as 0.

**BCM\_2\_CMAC**: This is a 32 bytes CMAC used to authenticate the source of this message. If the CMAC is not valid then the content of the whole message shall be ignored.

### Frame BRAKE\_2

#### Message Layout

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ID | Byte0 | Byte1 | Byte2 | Byte3 | Byte4 | Byte5 | Byte6 | Byte7 | Period |
| 0x342 | x | x | VehSpeed\_H | VehSpeed\_L | x | x | BRAKE\_2\_MC | BRAKE\_2\_CRC | 100ms |

#### Signals Description

**Vehicle Speed\_H:** This signal contains the 2 bits more significant of the Vehicle Speed value.

**Vehicle Speed\_L**: This signal contains the 8 bits less significant of the Vehicle Speed value.

Vehicle Speed Signal is a 10 bits signal and is represented as follow

Vehicle Speed = (uint16) ((Vehicle Speed\_H <<8)| Vehicle Speed\_L)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Vehicle Speed\_H | | | | | | | | Vehicle Speed\_L | | | | | | | |
| B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| X | X | X | X | X | X | 1/0 | 1/0 | 1/0 | 1/0 | 1/0 | 1/0 | 1/0 | 1/0 | 1/0 | 1/0 |

Signal Range goes from 0 to 0x03FF. (0 to 1023 in Decimal)

Resolution bit is equal to 0.25Km/h.

This means a value of 0x0100 is equal to 64 Km/h.

**BRAKE\_2\_MC:** This is the Message counter of the BRAKE\_2 frame. This signal shall go between 0 and 255 with increments of 1. This signal shall be updated every time the telegram is transmitted. If the signal reaches its limit (255), then the counter value shall initialize as 0.

**BRAKE\_2\_CRC:** This is the Cyclic Redundancy Check for BRAKE\_2 frame. This signal shall be calculated with and CRC8 algorithm from Byte0 to Byte5.

### Frame TRANSM\_2

#### Message Layout

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ID | Byte0 | Byte1 | Byte2 | Byte3 | Byte4 | Byte5 | Byte6 | Byte7 | Period |
| 0x102 | x | x | ShiftLeverPos | x | x | x | TRANSM\_2\_MC | TRANSM\_2\_CRC | 20ms |

#### Signals Description

**ShiftLeverPos:** This Signal contains the Shift Lever Position

* 0x00 (PARK) Shift Lever Position is in PARK position.
* 0x01 (DRIVE\_1) Shift Lever Position is in DRIVE\_1 position.
* 0x02 (DRIVE\_2) Shift Lever Position is in DRIVE\_2 position.
* 0x03 (DRIVE\_3) Shift Lever Position is in DRIVE\_3 position.
* 0x04 (NEUTRAL) Shift Lever Position is in NEUTRAL position.
* 0x05 (MANUAL) Shift Lever Position is in MANUAL position.
* 0x06 (REVERSE) Shift Lever Position is in REVERSE position.
* 0x07 (SNA) Shift Lever Position is a Signal Not Available.
* Values different than this shall be considered as INVALID Data.

**TRANSM\_2\_MC:** This is the Message counter of the TRANSM\_2 frame. This signal shall go between 0 and 255 with increments of 1. This signal shall be updated every time the telegram is transmitted. If the signal reaches its limit (255), then the counter value shall initialize as 0.

**TRANSM\_2\_CRC:** This is the Cyclic Redundancy Check for TRANSM\_2 frame. This signal shall be calculated with and CRC8 algorithm from Byte0 to Byte5.

### Frame TRANSM \_4

#### Message Layout

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ID | Byte0 | Byte1 | Byte2 | Byte3 | Byte4 | Byte5 | Byte6 | Byte7 | Period |
| 0x104 | x | x | x | x | x | ShiftLeverEng | TRANSM\_4\_MC | TRANSM\_4\_CRC | 20ms |

#### Signals Description

**ShiftLeverEng:** This Signal contains the Shift Lever Position Engaged.

* 0x00 (SHIFT\_LEVER\_NOT\_ENGAGED) Shift Lever Position is not engaged.
* 0x01 (SHIFT\_LEVER\_ENGAGED) Shift Lever Position is engaged.
* Values different than this shall be considered as INVALID Data.

**TRANSM\_4\_MC:** This is the Message counter of the TRANSM\_4 frame. This signal shall go between 0 and 255 with increments of 1. This signal shall be updated every time the telegram is transmitted. If the signal reaches its limit (255), then the counter value shall initialize as 0.

**TRANSM\_4\_CRC:** This is the Cyclic Redundancy Check for TRANSM\_4 frame. This signal shall be calculated with and CRC8 algorithm from Byte0 to Byte5.

### Frame DCU\_1

#### Message Layout

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ID | Byte0 | Byte1 | Byte2 | Byte3 | Byte4 | Byte5 | Byte6 | Byte7 | Period |
| 0x200 | WindowPos | LockingReq | WindowOp | RearWindowLock | DoorLockSts | WindowControl | DCU\_1\_MC | DCU\_1\_CRC | 100ms |

#### Signals Description

**WindowPos:** This Signal reports to the network the Driver Window Position.

* 0x00 (IN\_BETWEEN) Windows is in between this means the window is not completely OPEN neither completely CLOSE.
* 0x01 (COMPLETELY\_OPEN) Window is Completely Open.
* 0x02 (COMPLETELY\_CLOSE) Window is Completely Close.
* 0x03 (ERROR) windows is on an ERROR state.
* Values different than this shall be considered as INVALID Data.

**LockingReq:** This signal reports to the network the Lock or Unlock Request to Body Control Module.

* 0x00 (NO\_LOCKING\_REQ) There is no Lock or unlock command requested.
* 0x01 (LOCK\_REQ) User has request a LOCK request operation.
* 0x02 (UNLOCK\_REQ) User has request an UNLOCK request operation.
* 0x03Values different than this shall be considered as INVALID Data.

**WindowOp:** This signal reports the windows current Operation.

* 0x00 (WINDOW\_IDLE) Window is not moving.
* 0x01 (WINDOW\_UP) Window is doing a Close Operation.
* 0x02 (WINDOW\_DOWN) Window is doing a Down Operation..
* Values different than this shall be considered as INVALID Data.

**RearWindowLock:** This signal reports the status to block the Window Control operation for Rear Windows.

* 0x00 (REAR\_WINDOW\_UNBLOCK) Rear Windows are allowed to operate.
* 0x01 (REAR\_WINDOW\_BLOCK) Rear Windows shall not Operate.
* Values different than this shall be considered as INVALID Data.

**DoorLockSts:** This signal reports the Door Lock Status.

* 0x00 (DOOR\_LOCK) Door is currently Locked.
* 0x01 (DOOR\_UNLOCK) Door is currently Unlocked.
* 0x02 (DOOR\_UNKNWON) Door is in an unknown State.
* 0x03 (ERROR) Door Position is on an ERROR state.
* Values different than this shall be considered as INVALID Data.

**WindowControl:** This signal is used to control other Door Control Modules on the network.

This signal Layout is described as follows:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
| x | x | WindowControl\_Passenger | | WindowControl\_RearLeft | | WindowControl\_RearRight | |

Commands:

* 0x00 (WINDOW\_NO\_REQ) No Window Request.
* 0x01 (WINDOW\_UP\_REQ) Indicated Window Close.
* 0x02 (WINDOW\_DOWN\_REQ) Indicated Window Open.
* 0x03 Values different than this shall be considered as INVALID Data.

**DCU\_1\_MC:** This is the Message counter of the DCU\_1 frame. This signal shall go between 0 and 255 with increments of 1. This signal shall be updated every time the telegram is transmitted. If the signal reaches its limit (255), then the counter value shall initialize as 0.

**DCU\_1\_CRC:** This is the Cyclic Redundancy Check for DCU\_1 frame. This signal shall be calculated with and CRC8 algorithm from Byte0 to Byte5.

### Frame DCU\_2

#### Message Layout

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ID | Byte0 | Byte1 | Byte2 | Byte3 | Byte4 | Byte5 | Byte6 | Byte7 | Period |
| 0x200 | WindowPos | LockingReq | WindowOp | RESERVED | DoorLockSts | RESERVED | DCU\_2\_MC | DCU\_2\_CRC | 100ms |

#### Signals Description

**WindowPos:** This Signal reports to the network the Driver Window Position.

* 0x00 (IN\_BETWEEN) Windows is in between this means the window is not completely OPEN neither completely CLOSE.
* 0x01 (COMPLETELY\_OPEN) Window is Completely Open.
* 0x02 (COMPLETELY\_CLOSE) Window is Completely Close.
* 0x03 (ERROR) windows is on an ERROR state.
* Values different than this shall be considered as INVALID Data.

**LockingReq:** This signal reports to the network the Lock or Unlock Request to Body Control Module.

* 0x00 (NO\_LOCKING\_REQ) There is no Lock or unlock command requested.
* 0x01 (LOCK\_REQ) User has request a LOCK request operation.
* 0x02 (UNLOCK\_REQ) User has request an UNLOCK request operation.
* 0x03Values different than this shall be considered as INVALID Data.

**WindowOp:** This signal reports the windows current Operation.

* 0x00 (WINDOW\_IDLE) Window is not moving.
* 0x01 (WINDOW\_UP) Window is doing a Close Operation.
* 0x02 (WINDOW\_DOWN) Window is doing a Down Operation..
* Values different than this shall be considered as INVALID Data.

**DoorLockSts:** This signal reports the Door Lock Status.

* 0x00 (DOOR\_LOCK) Door is currently locked.
* 0x01 (DOOR\_UNLOCK Door is currently Unlocked.
* 0x02 (DOOR\_UNKNWON) Door is in an unknown State.
* 0x03 (ERROR) Door Position is on an ERROR state.
* Values different than this shall be considered as INVALID Data.

**RESERVED**: Reserved bytes shall be transmitted as 0xFF.

**DCU\_2\_MC:** This is the Message counter of the DCU\_2 frame. This signal shall go between 0 and 255 with increments of 1. This signal shall be updated every time the telegram is transmitted. If the signal reaches its limit (255), then the counter value shall initialize as 0.

**DCU\_2\_CRC:** This is the Cyclic Redundancy Check for DCU\_2 frame. This signal shall be calculated with and CRC8 algorithm from Byte0 to Byte5.

### Frame DCU\_3

#### Message Layout

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ID | Byte0 | Byte1 | Byte2 | Byte3 | Byte4 | Byte5 | Byte6 | Byte7 | Period |
| 0x200 | WindowPos | RESERVED | WindowOp | RESERVED | DoorLockSts | RESERVED | DCU\_3\_MC | DCU\_3\_CRC | 100ms |

#### Signals Description

**WindowPos:** This Signal reports to the network the Driver Window Position.

* 0x00 (IN\_BETWEEN) Windows is in between this means the window is not completely OPEN neither completely CLOSE.
* 0x01 (COMPLETELY\_OPEN) Window is Completely Open.
* 0x02 (COMPLETELY\_CLOSE) Window is Completely Close.
* 0x03 (ERROR) windows is on an ERROR state.
* Values different than this shall be considered as INVALID Data.

**WindowOp:** This signal reports the windows current Operation.

* 0x00 (WINDOW\_IDLE) Window is not moving.
* 0x01 (WINDOW\_UP) Window is doing a Close Operation.
* 0x02 (WINDOW\_DOWN) Window is doing a Down Operation..
* Values different than this shall be considered as INVALID Data.

**DoorLockSts:** This signal reports the Door Lock Status.

* 0x00 (DOOR\_LOCK) Door is currently locked.
* 0x01 (DOOR\_UNLOCK Door is currently Unlocked.
* 0x02 (DOOR\_UNKNWON) Door is in an unknown State.
* 0x03 (ERROR) Door Position is on an ERROR state.
* Values different than this shall be considered as INVALID Data.

**RESERVED**: Reserved bytes shall be transmitted as 0xFF.

**DCU\_3\_MC:** This is the Message counter of the DCU\_3 frame. This signal shall go between 0 and 255 with increments of 1. This signal shall be updated every time the telegram is transmitted. If the signal reaches its limit (255), then the counter value shall initialize as 0.

**DCU\_3\_CRC:** This is the Cyclic Redundancy Check for DCU\_3 frame. This signal shall be calculated with and CRC8 algorithm from Byte0 to Byte5.

### Frame DCU\_4

#### Message Layout

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ID | Byte0 | Byte1 | Byte2 | Byte3 | Byte4 | Byte5 | Byte6 | Byte7 | Period |
| 0x200 | WindowPos | RESERVED | WindowOp | RESERVED | DoorLockSts | RESERVED | DCU\_4\_MC | DCU\_4\_CRC | 100ms |

#### Signals Description

**WindowPos:** This Signal reports to the network the Driver Window Position.

* 0x00 (IN\_BETWEEN) Windows is in between this means the window is not completely OPEN neither completely CLOSE.
* 0x01 (COMPLETELY\_OPEN) Window is Completely Open.
* 0x02 (COMPLETELY\_CLOSE) Window is Completely Close.
* 0x03 (ERROR) windows is on an ERROR state.
* Values different than this shall be considered as INVALID Data.

**WindowOp:** This signal reports the windows current Operation.

* 0x00 (WINDOW\_IDLE) Window is not moving.
* 0x01 (WINDOW\_UP) Window is doing a Close Operation.
* 0x02 (WINDOW\_DOWN) Window is doing a Down Operation..
* Values different than this shall be considered as INVALID Data.

**DoorLockSts:** This signal reports the Door Lock Status.

* 0x00 (DOOR\_LOCK) Door is currently locked.
* 0x01 (DOOR\_UNLOCK Door is currently Unlocked.
* 0x02 (DOOR\_UNKNWON) Door is in an unknown State.
* 0x03 (ERROR) Door Position is on an ERROR state.
* Values different than this shall be considered as INVALID Data.

**RESERVED**: Reserved bytes shall be transmitted as 0xFF.

**DCU\_4\_MC:** This is the Message counter of the DCU\_4 frame. This signal shall go between 0 and 255 with increments of 1. This signal shall be updated every time the telegram is transmitted. If the signal reaches its limit (255), then the counter value shall initialize as 0.

**DCU\_4\_CRC:** This is the Cyclic Redundancy Check for DCU\_4 frame. This signal shall be calculated with and CRC8 algorithm from Byte0 to Byte5.