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

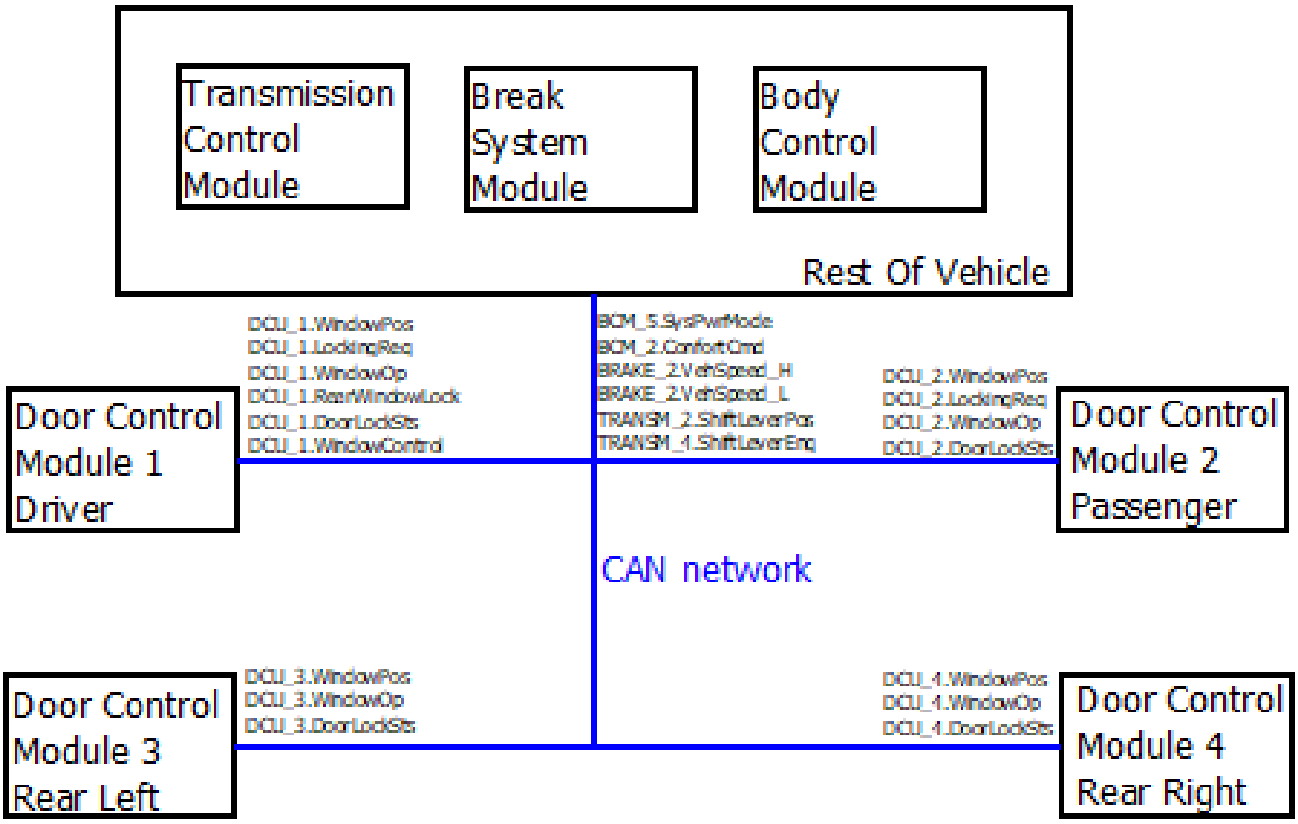
Details for requirements are described later this document.
The system shall not cost more than 9 USD (U.S. Dollars) per unit.
The supplier shall be capable to produce 100,000 units per year.
The systems shall assure system proper behavior for at least 10 years.
The supplier shall produce the system for 5 years consecutive with the possibility to extend this period.

System Description

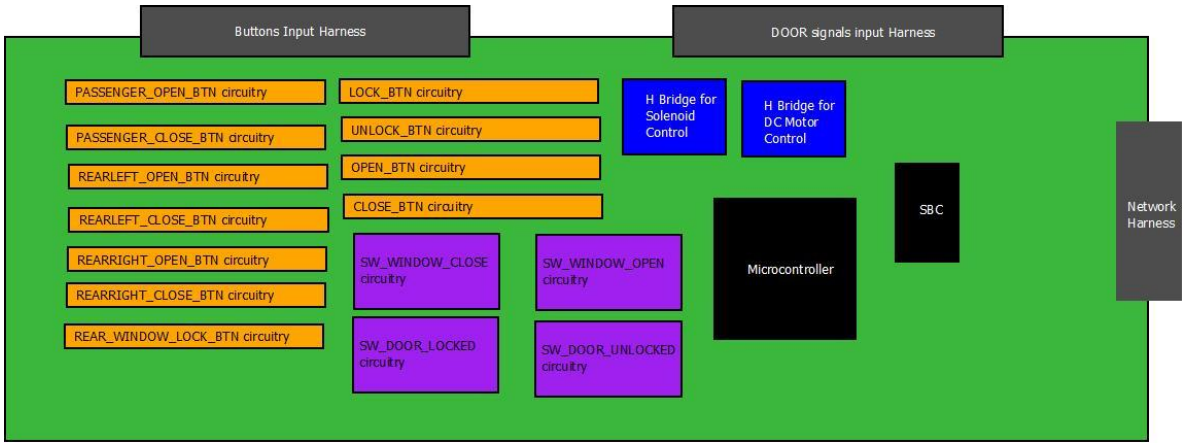
System consists of 4 Different ECUs that are placed on the 4 doors of the vehicle.

- 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

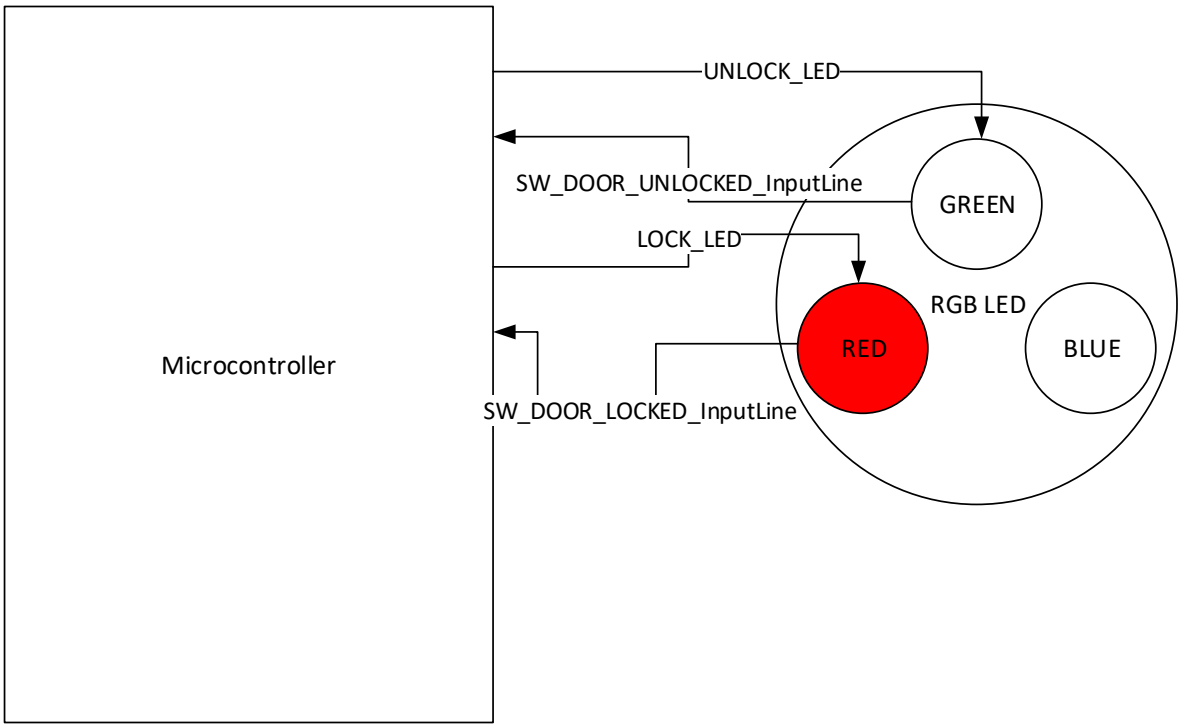
Door locking function is used to control the corresponding Door Solenoid that is used to lock and unlock the corresponding Door.

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 **DID_0180.LockBlinkTime milliseconds.**
3. Turn on LOCK_LED

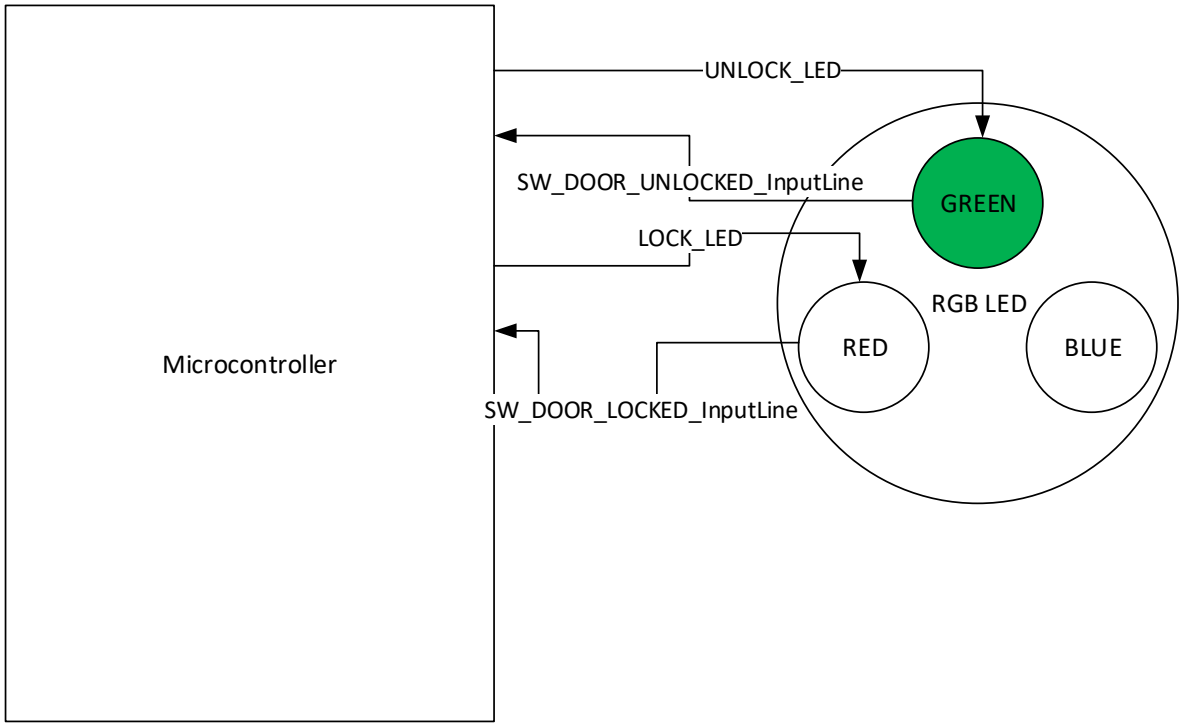


UNLOCK_DOOR_ ACTUATION

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

Sequence for actuation shall be:

1. Turn off LOCK_LED
2. Hold States during **DID_0180.UnlockBlinkTime milliseconds.**
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**.**

Driver Door shall execute **LOCK_DOOR_ACTUATION** when signal **DCU_2.LockingReq** transitions from **NO_LOCKING_REQ** to **LOCK_REQ**.

Driver Door shall execute **UNLOCK_DOOR_ACTUATION** when signal **DCU_2.LockingReq** transitions from **NO_LOCKING_REQ** to **UNLOCK_REQ**.

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**.

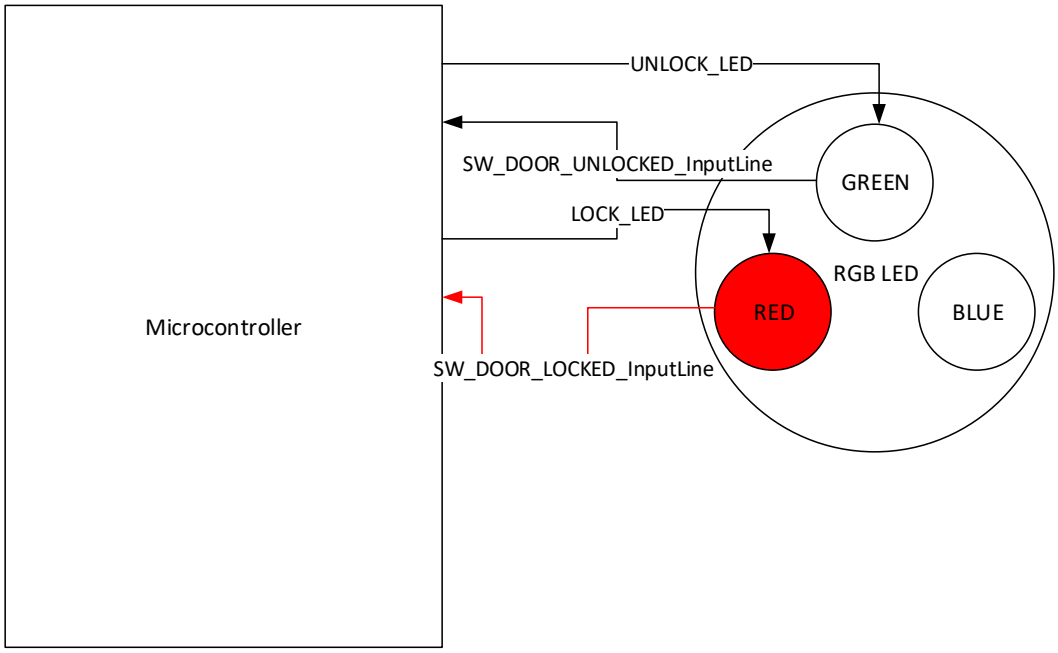
RearLeft Door shall execute **LOCK_DOOR_ACTUATION** when signal **DCU_2.LockingReq** transitions from **NO_LOCKING_REQ** to **LOCK_REQ**.
RearLeft Door shall execute **UNLOCK_DOOR_ACTUATION** when signal **DCU_2.LockingReq** transitions from **NO_LOCKING_REQ** to **UNLOCK_REQ**.

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

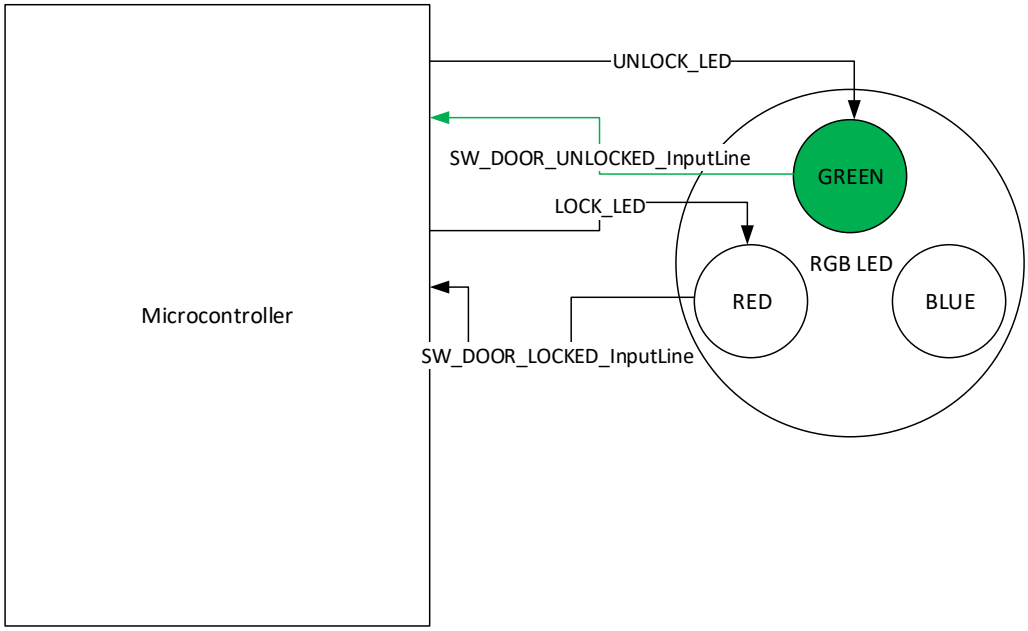
DOOR_LOCKED

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.**



DOOR_ERROR

Door Lock shall be considered as **DOOR_ERROR** if **SW_DOOR_LOCKED** and **SW_DOOR_UNLOCKED** are determined as **SW_ACTIVE** during 10 samples of 10 ms consecutively each.

DOOR_ERROR shall not be considered present when a **DOOR_LOCKED** or **DOOR_UNLOCKED** status has been determined without a **DOOR_ERROR** after Door Locking functionality is re-enable after a power cycle (See Chapter HW Diagnostics)

Door Lock Status Report

Driver Door shall transmit signal DCU_1.LockingReq with a value equal to 0x01 (LOCK_REQ) **only once** when LOCK_BTN of the DCU transitions from BTN_NOT_PRESSED to BTN_PRESSED. After that, Driver Door shall set DCU_1.LockingReq back to 0x00 (NO_LOCKING_REQ) and keep it like this as long as there is no lock/unlock request from LOCK_BTN/UNLOCK_BTN.

Driver Door shall transmit signal DCU_1.LockingReq with a value equal to 0x02 (UNLOCK_REQ) **only once** when UNLOCK_BTN of the DCU transitions from BTN_NOT_PRESSED to BTN_PRESSED. After that, Driver Door shall set DCU_1.LockingReq back to 0x00 (NO_LOCKING_REQ) and keep it like this as long as there is no lock/unlock request from LOCK_BTN/UNLOCK_BTN.

Passenger Door shall transmit signal DCU_2.LockingReq with a value equal to 0x01 (LOCK_REQ) **only once** when LOCK_BTN of the DCU transitions from BTN_NOT_PRESSED to BTN_PRESSED. After that, Driver Door shall set DCU_2.LockingReq back to 0x00 (NO_LOCKING_REQ) and keep it like this as long as there is no lock/unlock request from LOCK_BTN/UNLOCK_BTN.

Passenger Door shall transmit signal DCU_2.LockingReq with a value equal to 0x02 (UNLOCK_REQ) **only once** when UNLOCK_BTN of the DCU transitions from BTN_NOT_PRESSED to BTN_PRESSED. After that, Driver Door shall set DCU_2.LockingReq back to 0x00 (NO_LOCKING_REQ) and keep it like this as long as there is no lock/unlock request from LOCK_BTN/UNLOCK_BTN.

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

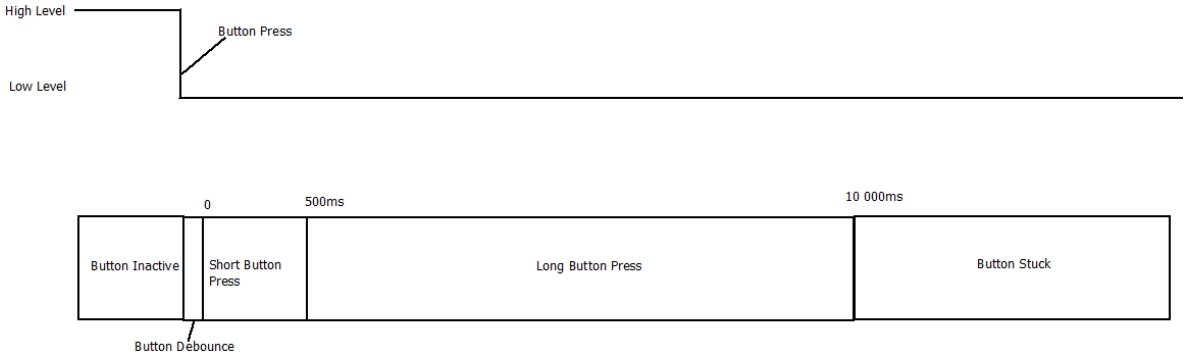
If a Button transitions from **BTN_NOT_PRESSED** to **BTN_PRESSED** and **BTN_NOT_PRESSED** within a time <= **DID_0111. LongBtnCfg** milliseconds then it shall be considered as a **SHORT_BTN_PRESS.**

Long Button Press

If a Button transitions from **BTN_NOT_PRESSED** to **BTN_PRESSED** and **BTN_NOT_PRESSED** within a time > **DID_0111. LongBtnCfg** and <= **DID_0111. StuckBtnCfg** 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 > **DID_0111. StuckBtnCfg** milliseconds 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 position when **PASSENGER_OPEN_BTN** state is equal to **SHORT_BTN_PRESS** or **LONG_BTN_PRESS.**
Driver Door shall report **WINDOW_DOWN_REQ** on **WindowControl_ Passenger** position when **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 position when **REARLEFT_OPEN_BTN** state is equal to **SHORT_BTN_PRESS** or **LONG_BTN_PRESS.**
Driver Door shall report **WINDOW_DOWN_REQ** on **WindowControl_ RearLeft** position when **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 position when **REARRIGHT_OPEN_BTN** state is equal to **SHORT_BTN_PRESS** or **LONG_BTN_PRESS.**
Driver Door shall report **WINDOW_DOWN_REQ** on **WindowControl_ RearRight** position when **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 position while **REAR_WINDOW_LOCK_BTN** state is equal to **BTN_PRESSED.**
Driver Door shall report **REAR_WINDOW_UNBLOCK** on **DCU_1. RearWindowLock** bits position while **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 (Do Not Implement until Module 4)

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.**

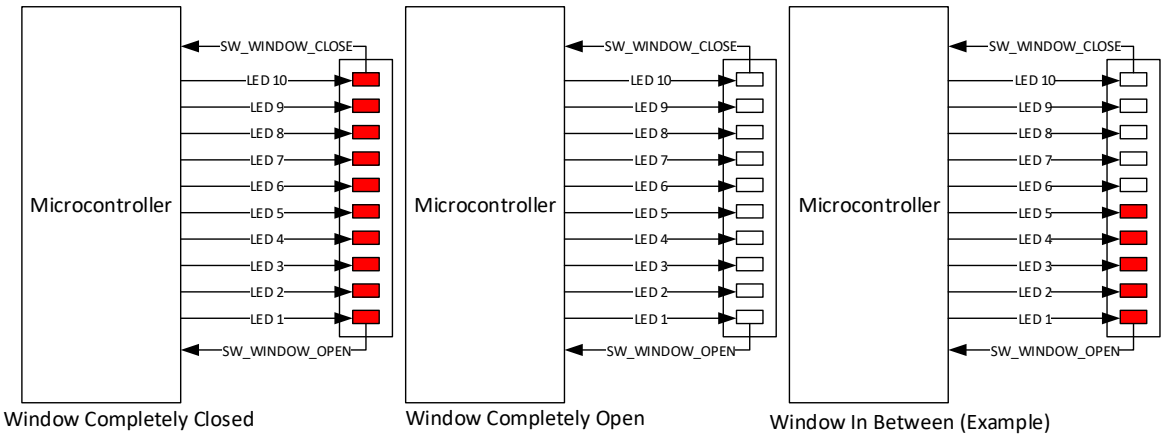
If **ANTIPINCH_SIGNAL** is present then an immediate **CANCEL_WINDOW_ACTUATION** shall be executed. Then a **GLOBAL_OPEN_ACTUATION** shall be executed. After **GLOBAL_OPEN_ACTUATION** is finished the **CLOSE_WINDOW_ACTUATION** and **GLOBAL_CLOSE_WINDOW_ACTUATION** shall be inhibited during 15 seconds 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 **WINDOW_ERROR** if **SW_WINDOW_CLOSE** is determined as **SW_ACTIVE** and **SW_WINDOW_OPEN** is determined as **SW_INACTIVE** during 10 samples of 10 ms consecutively each.

WINDOW_ERROR shall not be considered present if **SW_WINDOW_CLOSE** is determined as **SW_INACTIVE** and **SW_WINDOW_OPEN** is determined as **SW_ACTIVE** after Window Control functionality is re-enable after a power cycle (See Chapter HW Diagnostics)



Window Position Report

For **Driver Door,** it shall report the determined Lock Status via **DCU_1. WindowPos.**
For **Passenger Door,** it shall report the determined Lock Status via **DCU_2. WindowPos.**
For **RearLeft Door,** it shall report the determined Lock Status via **DCU_3. WindowPos.**
For **RearRight Door,** it shall 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.

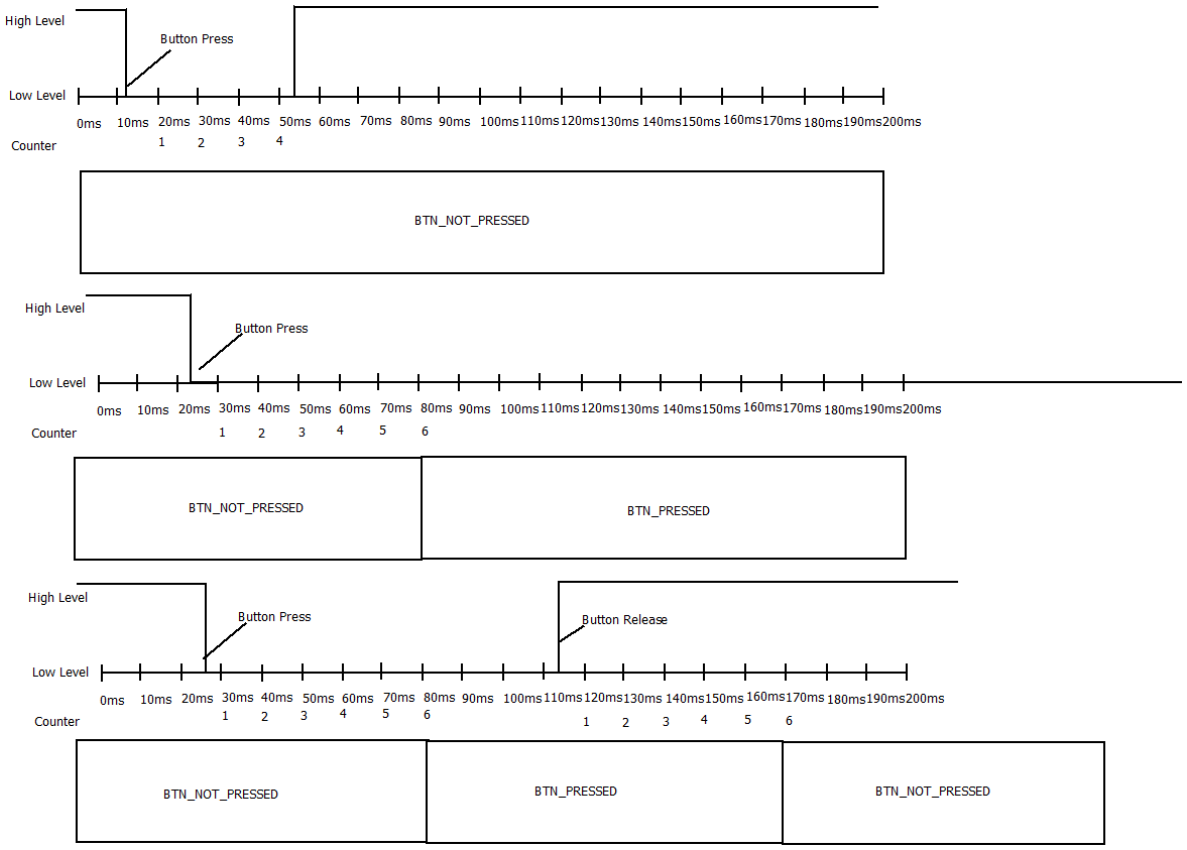
All the buttons used on the system shall use a inverted logic. This means they will be considered as ACTIVE when they are in Low state. Idle state shall be high.

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 considered 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.

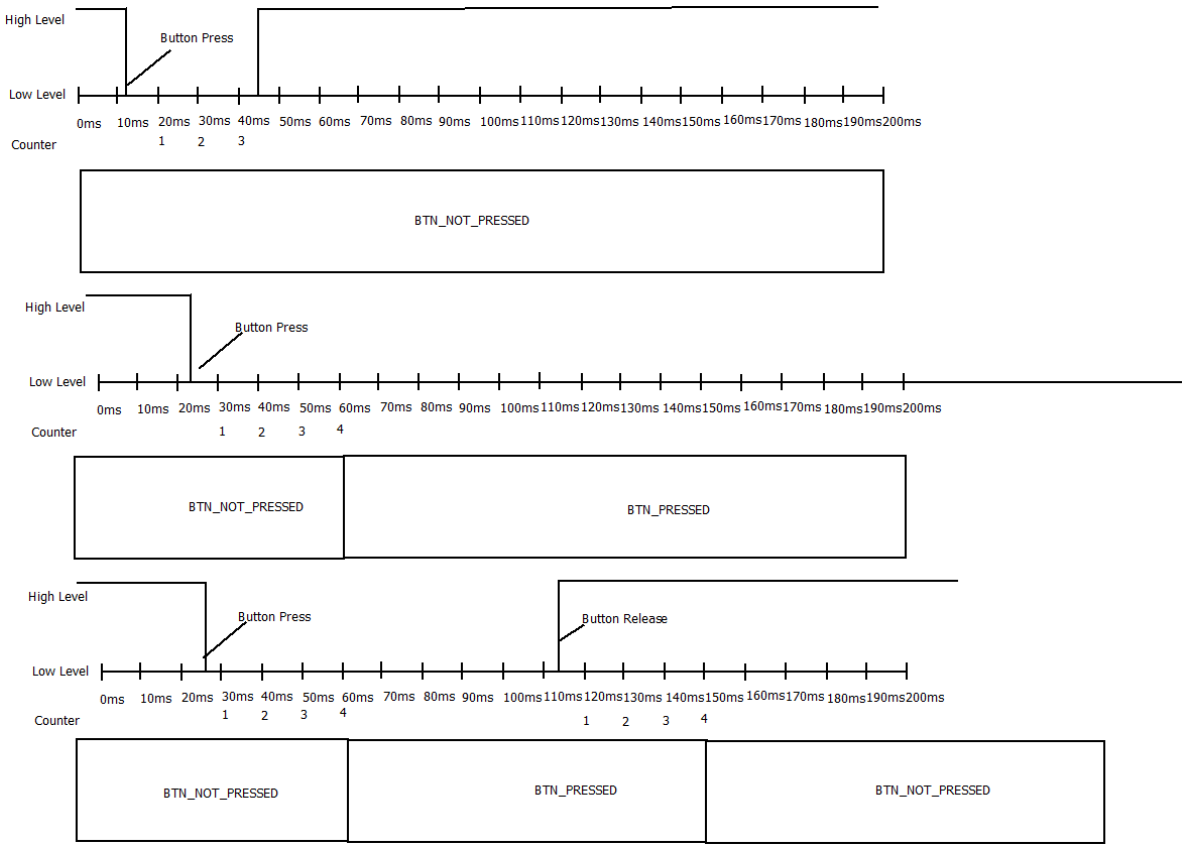
All the buttons used on the system shall use a positive logic. This means they will be considered as ACTIVE when they are in High state. Idle state shall be Low.

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.



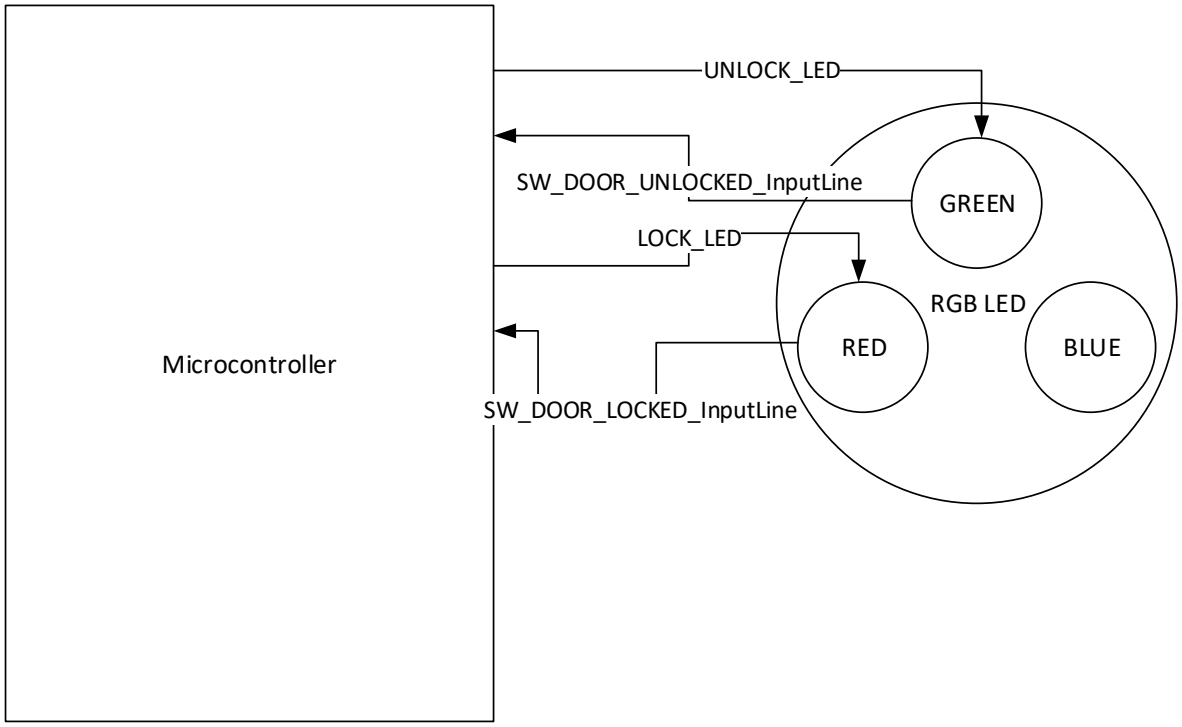
Solenoid Control

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.

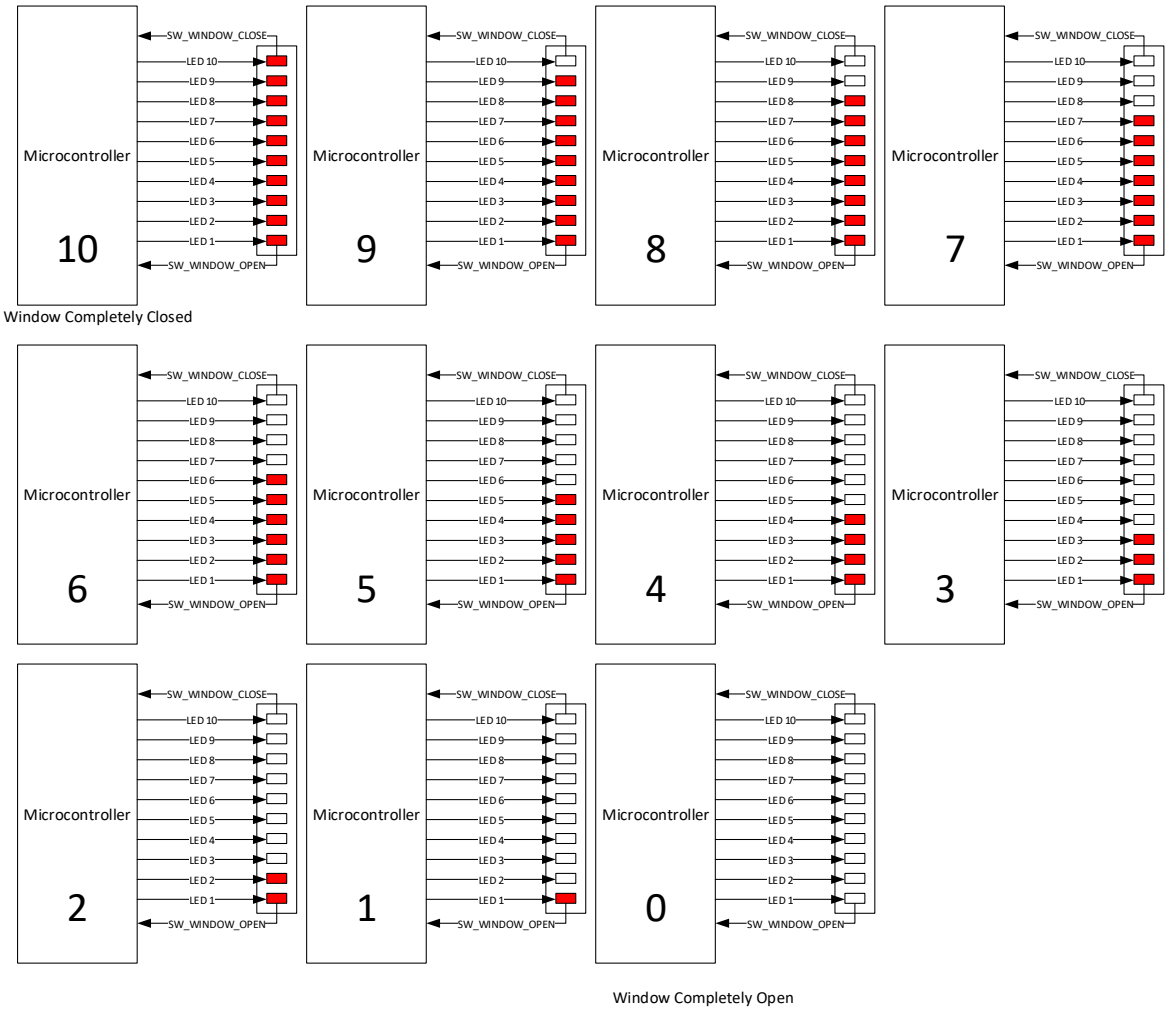


Window Control

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

Door Error

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

In case Door is considered as **DOOR_ERROR** corresponding DTC shall be set:

DTC: 0xA00100 for Driver Door Position Error Detected

DTC: 0xA00200 for Passenger Door Position Error Detected
DTC: 0xA00300 for Rear Right Door Position Error Detected
DTC: 0xA00400 for Rear Left Door Position Error Detected
In case DOOR_ERROR conditions are no longer detected then corresponding DTC shall be clear.

Button Stuck

In case a Button is considered as Stuck, that button shall be ignored until next power Cycle (Transition OFF-> RUN)

Depending on the Button Detected as Stuck, corresponding DTC shall be set:

DTC: 0x901100 for Driver Door - Window Control Buttons Error
DTC: 0x901200 for Driver Door – Door Locking Control Buttons Error
DTC: 0x902100 for Passenger Door - Window Control Buttons Error
DTC: 0x902200 for Passenger Door – Door Locking Control Buttons Error
DTC: 0x903100 for Rear Right Door - Window Control Buttons Error
DTC: 0x904100 for Rear Left Door - Window Control Buttons Error
In case Buttons are no longer considered as Stuck then, corresponding DTC shall be clear.

Window Error

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

In case a window is considered as **WINDOW_ERROR** corresponding DTC shall be set:

DTC: 0xB00100 for Driver Window Position Error Detected
DTC: 0xB00200 for Passenger Window Position Error Detected
DTC: 0xB00300 for Rear Right Window Position Error Detected
DTC: 0xB00400 for Rear Left Window Position Error Detected
In case WINDOW_ERROR conditions are no longer detected then corresponding DTC shall be clear.

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.

The SW variant to operate with shall be determined during initialization phase. Once determined, it shall not change until the next operating-cycle.

Configuration

SW Variant Configuration will be done through DID $011D. See appendix B.

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 A: 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
0x201	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
0x202	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
0x203	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
0x204	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.

Appendix B: Diagnostics

CAN IDs for Diagnostics

Functional Request ID: 0x18DBFE**F1**

Physical Request ID Driver Door: 0x18DA**C4**F1
Physical Response ID Driver Door: 0x18DAF1**C4**
Physical Request ID Passenger Door: 0x18DA**C5**F1
Physical Response ID Passenger Door: 0x18DAF1**C5**
Physical Request ID Rear Right Door: 0x18DA**C6**F1
Physical Response ID Rear Right Door: 0x18DAF1**C6**
Physical Request ID Rear Left Door: 0x18DA**C7**F1
Physical Response ID Rear Left Door: 0x18DAF1**C7**

CAN TP information for Diagnostics

addressingFormat: CANTP_STANDARD
cancelation: True
paddingActivation: CANTP_OFF
taType: CANTP_PHYSICAL
timeoutBr: 0.0
TimeoutBs: 0.075
TimeoutCr: 0.15
TimeoutCs: 0.01
CanTpNar:0.025

UDS Services supported

UDS services Supported

UDS service	Description
$10	Diagnostic Session Control
$11	ECU Reset
$14	Clear Diagnostic Information
$19	Read DTC Information
$22	Read Data By Identifier
$27	Security Access
$2E	Write Data By Identifier
$3E	Tester Present

Diagnostic Session Control [$10]

A Door Control Unit Has to Support for all variants the next sessions:

- Default session (0x01)
- Extended session(0x03)

If an DCU is in a diagnostic mode different from defaultSession and a timeout occurs, the DCU has to finish all pending actions/command and return into defaultSession
If the Tester requests the opening of a diagnostic session which is not active at the moment the request is made (e.g.: to change diagnostic Session) the DCU shall start the new session only after any pending service in the current diagnostic session has been aborted (if pending service is not support in requested session)
If a new Diagnostic session is requested, DCU shall maintain all active diagnostics already running in previous session only if are supported in both sessions (this means that Diagnostics are supported both sessions, previous and new session requested) and are not dependent upon security access (since security level is finished between session changes)

Service $10 shall be access via Diagnostic Physical Access.
Service $10 shall be access via Diagnostic Functional Access.

Default Session

Subfunction: DiagnosticSessionType ($01)
This diagnostic session enables the default diagnostic session in the server(s) and does not support any diagnostic application timeout handling provisions (e.g., Tester Present service is not necessary to keep the session active).
DCU shall always start the default diagnostic session when powered up.
If no other diagnostic session is started, then the default diagnostic session shall be running as long as the server is powered
Session Entry Criteria
DCU shall enter Default Session with the receipt of diagnostic Session Control with sub- function value of $01 (default Session)
Sesion Exit Criteria
DCU shall exit Default Session with the receipt of Diagnostic Session Control with a sub-function value other than $01 (assuming all entry conditions are met for the requested session).
DCU shall exit Default Session if Power is removed from the ECU

Extended Session

Subfunction: DiagnosticSessionType ($03)
This Diagnostic Session provides a timer-controlled environment in which all the supported diagnostic services can be executed (given that security access protected diagnostic services are properly unlocked before usage).

Session Entry Criteria
DCU shall enter Extended Session from:

- Diagnostic Session $01

Session Exit Criteria
DCU shall exit Extended Session if:

- ECU Reset occurs
- Receipt of Diagnostic Session Control with a sub-function value other than $03 (assuming all entry conditions are met for the requested session)
- 5 second Diagnostic Session timeout is reached (See Tester present)

ECU Reset [$11]

The ECU Reset service $11 is used to request a reset to the ECU from the tester tool.
Service $11 shall be access via Diagnostic Physical Access.
Service $11 shall not be access via Diagnostic Functional Access.

Hard Reset [$11 01]

This Subservice is used to request a Hard reset this means RAM init will be lost after this reset event is performed.

Soft Reset [$11 03]

This Subservice is used to request a Soft reset this means RAM no init won't be lost after this reset event is performed.

Clear Diagnostic Information [14 Hex]

The Clear Diagnostic Information service is used by an external test tool to clear diagnostic information (i.e. Clear DTC information) in one or multiple ECUs‟ memory

Service $14 shall be access via Diagnostic Physical Access.
Service $14 shall be access via Diagnostic Functional Access.

Read DTC Information [$19]

The Read DTC Information service is used in UDS protocol to read the DTC's from a vehicle or from a particular ECU or node.
Service $19 shall be access via Diagnostic Physical Access.
Service $19 shall be access via Diagnostic Functional Access.

Report Number of DTCs By Status Mask [$19 01]

This type specifies that the ECU shall transmit to the external test tool the number of DTCs matching an external test tool defined status mask. This functionality applies its DTC status filtering to the current status of all DTCs incl. Those stored in the Chrono-stack.

Report DTCs By Status Mask [$19 02]

This type specifies that the ECU shall transmit to the test device a list of DTCs and corresponding statuses matching an external test tool defined status mask. This functionality applies its DTC status filtering to the current status of all DTCs incl. Those stored in the Chrono-stack.

Report Number of DTC By Severity Mask Record [$19 07]

This parameter specifies that the DCU shall transmit to the client the number of DTCs matching a client-defined severity mask record

Report DTC By Severity Mask Record [$19 08]

This parameter specifies that the DCU shall transmit to the client a list of DTCs and corresponding statuses matching a client-defined severity mask record

Report Severity Information of DTC [$19 09]

This parameter specifies that the DCU shall transmit to the client the severity information of a specific DTC specified in the client request message

Report Supported DTCs [$19 0A]

This parameter specifies that the DCU shall transmit to the external test tool a list of all DTCs and corresponding statuses currently supported within the ECU

Read Data by Identifier [$22]

The ReadDataByIdentifier service allows the client to request data record values from the server identified by one or more data identifiers. The ECU may limit the number of data identifiers that can be read in one request. DataIdentifier – Identifies the ECU data record(s) that are being requested by the ECU.

Service $22 shall be access via Diagnostic Physical Access.
Service $22 shall be access via Diagnostic Functional Access.

Security Access [$27]

The unlocking of the security level needs to be realized through the Security Access Service (SID 0x27) of UDS. The specific format and parameters of the secure access service are not specifically introduced here. Please refer to ISO14229-1:2013(E) 9.4 SecurityAccess(0x27) service chapter.
DCU shall support subservices:
$01 requestSeed. It will include a 16-byte lenght number that will be the seed for the security access algorithm.
$02 sendKey: It will contain a 16-byte length number that is the result of the calculation using the seed provided by subservice $27 01. If the Key provided by the tester tool matches with the calculated by the DCU then security level 1 will be unlocked.
Service $27 shall be access via Diagnostic Physical Access.
Service $27 shall be access via Diagnostic Functional Access.

**Preliminar Security algorithm for DCU implies to receive a value different than 0x00000000000000000000000000000000 on sub-service $27 01 and receive a value different than 0x00000000000000000000000000000000 on sub service $27 02. This might change during module 5.

Write Data by Identifier [$2E]

This service allows the client/Tester to write information into the server/ECU at an internal location specified by the provided data identifier. The WriteDataByIdentifier service is used by the client to write a data Record to a server. The data is identified by a data identifier and may or may not be secured.

Service $2E shall be access via Diagnostic Physical Access.

Service $2E shall not be access via Diagnostic Functional Access.

Tester Present [$3E]

This service is used to keep one or multiple ECUs in a nondefault session.
When Diagnostic Sessions other than Default Session are started, session timeout timer is required.
Timeout time for Tester Present shall be 5000 ms.
DCU shall timeout and return to Default Session if after 5000 ms (ECU S3 time) no Tester Present diagnostic request have been received
Note: Timeout requirements above refers only to Tester Present but it could apply for all other Diagnostic Requests
Service $3E shall be access via Diagnostic Physical Access.
Service $3E shall be access via Diagnostic Functional Access.

Data Identier Lists

DID $0111: Button Configuration

This DID has the purpose to indicate the value for Stuck button detection.
This DID shall suspport service Read DID $22.
This DID shall suspport service Write DID $2E.
Write DID operation shall be protected by Extended sesssion.
Write DID operation shall be protected by security level $01.

ID	Byte0	Byte1	Byte2
0x0111	StuckBtnCfg		LongBtnCfg

DID_0111. StuckBtnCfg:
Resolution 10 ms.
Defaultvalue: 1000 (10 000ms)
DID_0111. LongBtnCfg:
Resolution 10 ms.
Defaultvalue: 50 (500ms)

DID $011D: ECU Variant Configuration

This DID has the purpose the ECU variant. Depending on the ECU variuant the SW shall behave on different ways.
This DID shall suspport service Read DID $22
This DID shall suspport service Write DID $2E.
Write DID operation shall be protected by Extended sesssion.
Write DID operation shall be protected by security level $01.

ID	Byte0
0x011D	ECU_Variant

DID_011D. ECU_Variant:

Value Range: 0 - 3.

Value Description:

Value	Variant Behavior
0	Driver Door
1	Passenger Door
2	RearLeft Door
3	RearRight Door

Default value: 0 (Driver Door)

DID $0180: Door Locking Configuration

This DID has the purpose to configure the duration of the LED on time for Locking and unlock actuations for Door Locking funcitonality.

This DID shall suspport service Read DID $22

This DID shall suspport service Write DID $2E

Write DID operation shall be protected by Extended sesssion.

Write DID operation shall be protected by security level $01.

ID	Byte0	Byte1
0x0180	UnlockBlinkTime	LockBlinkTime

DID_0180.UnlockBlinkTime:

Resolution 10 ms.

Defaultvalue: 10 (100ms)

DID_0180.LockBlinkTime milliseconds.

Resolution 10 ms.

Defaultvalue: 10 (100ms)

DID $0200: Developer ID

This DID has the purpose to indicate the ID for the developer that has created the code.

ID	Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7
0x0200	ID0	ID1	ID2	ID3	ID4	ID5	ID6	ID7

This DID has the purpose to indicate the value of the employee ID for the SW developer of the solution.

This DID shall suspport service Read DID $22

DID_0200. ID0:

Resolution: N/A

Defaultvalue: 0x30 (‘0’)

DID_0200. ID1:

Resolution: N/A

Defaultvalue: 0x30 (‘0’)

DID_0200. ID2:

Resolution: N/A

Defaultvalue: 0x30 (‘0’)

DID_0200. ID3:

Resolution: N/A

Defaultvalue: 0x30 (‘0’)

DID_0200. ID4:

Resolution: N/A

Defaultvalue: 0x30 (‘0’)

DID_0200. ID5:

Resolution: N/A

Defaultvalue: 0x30 (‘0’)

DID_0200. ID6:

Resolution: N/A

Defaultvalue: 0x30 (‘0’)

DID_0200. ID7:

Resolution: N/A

Defaultvalue: 0x30 (‘0’)