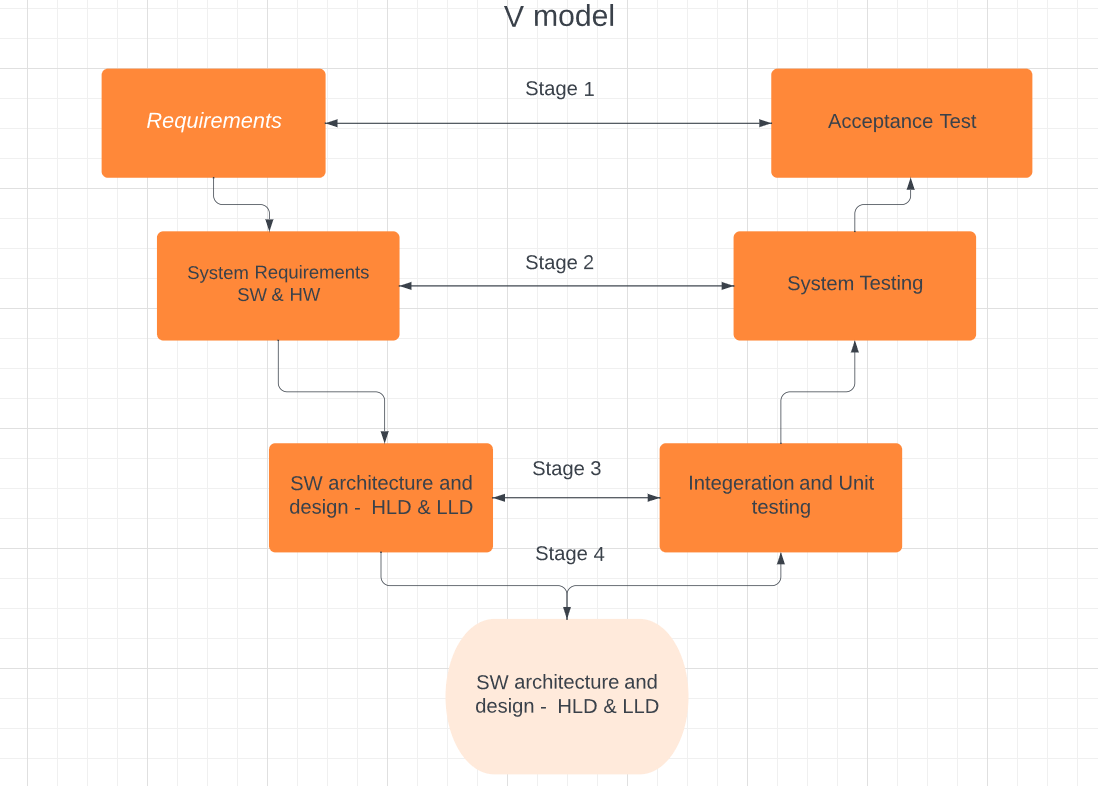
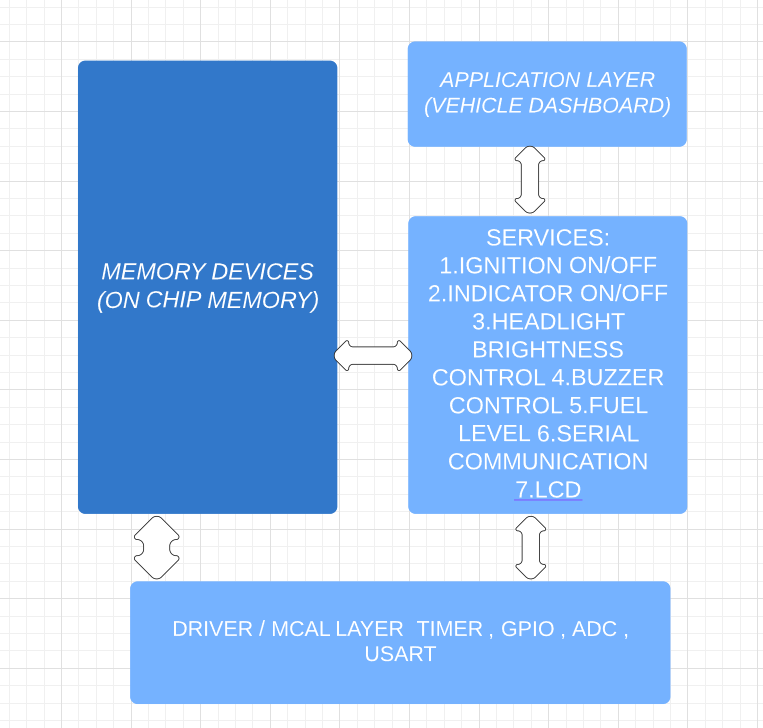
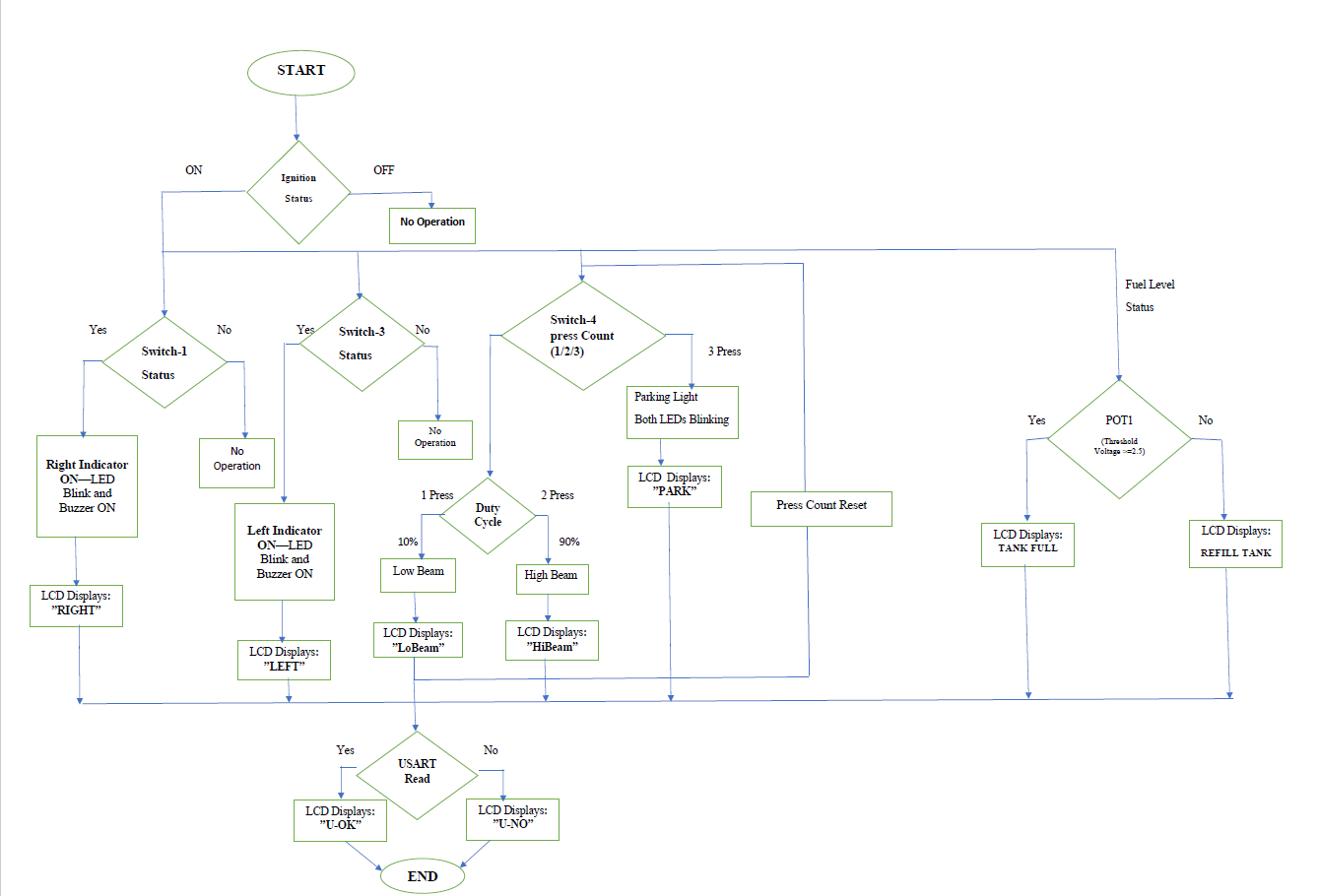
**MINI PROJECT**

**Connection Configuration:**

**Flowchart:**

****

****

****

**Requirements:**

Design Requirement 1

1- If Ignition OFF--> No operation

2- Ignition switch 1st press -> Ignition Status ON-> Next Press-> Ignition Status OFF (Toggle Mechanism)

3- If Ignition ON--> Ignition Status ON

i) If Right Indicator ON (Switch1 1st press-> Right Indicator Status ON -> next press-> Right Indicator Status Off)

- LED3 will blink continuously with a frequency of 0.5Hz 50% Duty).

- Buzzer will indicate with a tone with a frequency of 0.5Hz 50% Duty Buzzer should be in synch with the corresponding Indicator LED)

\*Above status won't work if both Left & Right Indicator Switches are ON

ii) If Left Indicator ON (Switch3 1st press-> Left Indicator Status ON -> next press-> Left Indicator Status Off)

- LED4 will blink continuously with a frequency of 0.5Hz 50% Duty.).

- Buzzer will indicate with a tone with a frequency of 0.5Hz 50% DutyBuzzer should be in synch with the corresponding Indicator LED)

\*Above status won't work if both Left & Right Indicator Switches are ON

iii) Simulated Fuel Indicator status to be continuously shown in Live Expression Window. Simulate the POT1 as Fuel Level Indicator & design respective ADC driver to display the Fuel Level in the STLink Debugger Live Window. Ex: Maximum position of pot is 100% and the minimum position of the pot is 0% and the display of fuel level should show with resolution of 1%

iv) If Head Light switch is pressed (Switch4 1st press->Light Low Beam ->next press-> Light High Beam-> next press-> Parking Light (both LED4 & LED3) blink at 1Hz-> next press OFF

- For Low Beam: LED1 with 10% PWM Duty Cycle

- For High Beam: LED1 with 90% PWM Duty Cycle

- For Parking Light: Both LED3 & LED4 blink & Buzzer to indicate with different tone (with a frequency of 2Hz 50% Duty) then indicator

\*Parking light won't operate, if either Left or Right Indicator is ON

v) UART is continuously passing a status message 'j' from Rx to Tx & the status has to be shown on STLink Debugger Live Window. Loop Back UART3 RX & TX pins to verify the data sent & received. (Find the Rx & Tx pins of UART3 with the help of datasheet). Every 500msec the UART status must be displayed on STLink Debugger Live Window.

Design Requirement II

1- If Ignition OFF--> No operation: LCD to display "CAROFF"

2- Ignition switch 1st press -> Ignition Status ON-> Next Press-> Ignition Status OFF (Toggle Mechanism)

3- If Ignition ON--> Ignition Status ON: LCD to display "CARON"

i) If Right Indicator ON (Switch1 1st press-> Right Indicator Status ON -> next press-> Right Indicator Status Off)

- LCD to display "Right"

- Buzzer will indicate with a beep frequency of 0.5Hz 50% Duty

\*Above status won't work if both Left & Right Indicator Switches are ON

ii) If Left Indicator ON (Switch3 1st press-> Left Indicator Status ON -> next press-> Left Indicator Status Off)

- LCD to display "Left"

- Buzzer will indicate with a with a frequency of 0.5Hz 50% Duty (Same as Right Indicator tone)

\*Above status won't work if both Left & Right Indicator Switches are ON

iii) Simulated Fuel Indicator status to be continuously displayed on LCD as: "Fuel:value". Simulate the POT1 as Fuel Level Indicator & design respective ADC driver to display the Fuel Level.

iv) If Head Light switch is pressed (Switch4 1st press->Light Low Beam ->next press-> Light High Beam-> next press-> Parking Light(both LED4 & LED3) blink-> next press OFF

- For Low Beam: LED1 with 10% PWM Duty Cycle: LCD to display "LoBeam"

- For High Beam: LED1 with 90% PWM Duty Cycle: LCD to display "HiBeam"

- For Parking Light: LCD to display "park" & Buzzer to indicate with different tone(with a frequency of 2Hz 50% Duty) than indicator

\*Parking mode won't operate, if either Left or Right Indicator is ON

v) UART is continuously passing a status message 'j' from Rx to Tx & the status has to be shown on LCD at regular interval.

Loop Back UART3 RX & TX pins to verify the data sent & received(Find the Rx & Tx pins of UART3 with the help of datasheet). Every 500msec the UART status has to be displayed on LCD as below:

-If Message passed: LCD to display"U:OK"

-If message not passed: LCD to display"U:NO"

**Code:**

/\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @file : main.c

\* @author : Ranjith

\* @brief : Mini project for car indicator, Head light and UART

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @attention

\*

\* Copyright (c) 2023 STMicroelectronics.

\* All rights reserved.

\*

\* This software is licensed under terms that can be found in the LICENSE file

\* in the root directory of this software component.

\* If no LICENSE file comes with this software, it is provided AS-IS.

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

**#include** <stdint.h>

**#include** <stdio.h>

**#include** "stm32f405xx.h"

//global variables

uint32\_t Value;

**char** key;

// Function prototypes

**static** **void** **tim3\_pc6\_set\_dutycycle**(uint32\_t freq, uint32\_t duty\_cycle, uint8\_t channel);

**static** **void** **pc6\_tim3\_pwm\_config**(**void**);

**static** **void** **pwm\_moder**(**void**);

**static** **void** **timer\_delay**(**int** sec);

**static** **void** **ADC\_init**(**void**);

**static** **void** **start\_conversion**(**void**);

**static** uint32\_t **ADC\_result**(**void**);

**int** **main**()

{

pc6\_tim3\_pwm\_config();

ADC\_init();

start\_conversion();

UART2\_init();

// Enable clock for GPIOA, GPIOB, and GPIOC

RCC->AHB1ENR |= (1 << 0) | (1 << 1) | (1 << 2);

// Configure GPIOB pins

GPIOB->MODER |= (1 << 26); // Set PB13 as general-purpose output mode

GPIOB->MODER &= ~(1 << 27); // Clear PB13 alternate function mode

GPIOB->MODER |= (1 << 28); // Set PB14 as general-purpose output mode

GPIOB->MODER &= ~(1 << 29); // Clear PB14 alternate function mode

GPIOB->MODER |= (1 << 30); // Set PB15 as general-purpose output mode

GPIOB->MODER &= ~(1 << 31); // Clear PB15 alternate function mode

// Configure GPIOC pins

GPIOC->MODER |= (1 << 18); // Set PC9 as general-purpose output mode

GPIOC->MODER &= ~(1 << 19); // Clear PC9 alternate function mode

// Initialize indicator counters and headlight mode

**int** ign = 0, left = 0, right = 0, head = 0;

// Turn on initial LEDs

GPIOB->ODR |= (1 << 13); // Turn on the LED connected to PB13

GPIOB->ODR |= (1 << 14); // Turn on the LED connected to PB14

GPIOB->ODR |= (1 << 15); // Turn on the LED connected to PB15

**while**(1)

{

**while**(!(ADC1->SR) & (1U<<1));

Value=ADC\_result()/41;

**int** pinstatus1= GPIOA->IDR & (1<<15);

**int** pinstatus2= GPIOB->IDR & (1<<4);

**int** pinstatus3= GPIOB->IDR & (1<<3);

**int** pinstatus4= GPIOB->IDR & (1<<7);

**if**(pinstatus1==0)

{

**while**(!(GPIOA->IDR & (1<<15)));

ign++;

}

**if**(ign==1)

{

GPIOB->ODR &= ~(1<<13);

UART2\_write('j');

key=UART2\_READ();

**if**(pinstatus2==0)

{

**while**(!(GPIOB->IDR & (1<<4)));

**if**(right!=1)

{

left++;

}

}

**if**(left==1 && head!=3)

{

GPIOB->ODR ^=(1<<14); //led toggle

GPIOC->ODR ^= (1<<9); //buzzer toggle

timer\_delay(1); //for creating one second delay

}

**else** **if**(left==2)

{

GPIOC->ODR &=~(1<<9); //turning buzzer off

GPIOB->ODR |=(1<<14); //turning led off

left=0;

}

**if**(pinstatus3==0)

{

**while**(!(GPIOB->IDR & (1<<3)));

**if**(left!=1)

{

right++;

}

}

**if**(right==1 && head!=3)

{

GPIOB->ODR ^=(1<<15); //led toggle

GPIOC->ODR ^=(1<<9); //buzzer toggle

timer\_delay(1); //for creating one second delay

}

**else** **if**(right==2)

{

GPIOC->ODR &=~(1<<9); //turning buzzer off

GPIOB->ODR |=(1<<15); //turning led off

right=0;

}

//PWM parts comes in

**if**(pinstatus4==0 && left!=1 && right!=1)

{

**while**(!(GPIOB->IDR & (1<<7)));

head++;

}

**if**(head==1)

{

pwm\_moder();

tim3\_pc6\_set\_dutycycle(5000,30,1); //setting brightness 10 percentage

}

**else** **if**(head==2)

{

pwm\_moder();

tim3\_pc6\_set\_dutycycle(5000,90,1); //setting brightness 90 percentage

}

**else** **if**(head==3)

{

GPIOC->MODER &= ~(3 << 12); // Clear the bits responsible for the pin mode configuration for PC6

GPIOC->MODER |= (1 << 12); // Set PC6 as general-purpose output mode

GPIOC->ODR |= (1 << 6); // Turn on the LED connected to PC6 (led toggle)

GPIOB->ODR ^= (1 << 15); // Toggle the LED connected to PB15 (led toggle)

GPIOB->ODR ^= (1 << 14); // Toggle the LED connected to PB14 (led toggle)

GPIOC->ODR ^= (1 << 9); // Toggle the buzzer connected to PC9 (buzzer toggle)

**for** (**int** i = 0; i < 150000; i++); // Introduce a software delay using a loop

}

**else** **if**(head==4)

{

RCC->AHB1ENR |= (1 << 2); // Enable clock for GPIOC

GPIOC->MODER &= ~(3 << 12);// Reset the bits responsible for the pin mode configuration for PC6

GPIOC->MODER |= (1 << 12); // Set PC6 as general-purpose output mode

GPIOC->ODR |= (1 << 6);// Turn off the LED connected to PC6

GPIOB->ODR |= (1 << 15); // Turn off the LED connected to PB15

GPIOB->ODR |= (1 << 14); // Turn off the LED connected to PB14

GPIOC->ODR &= ~(1 << 9); // Turn off the buzzer connected to PC9

head=0;

}

}

**if**(ign==2)

{

GPIOC->ODR &= ~(1 << 9); // Turn off the buzzer connected to PC9 (BUZZOFF)

GPIOB->ODR |= (1 << 13); // Turn on the LED connected to PB13

GPIOB->ODR |= (1 << 15); // Turn on the LED connected to PB15

ign = 0; // Reset the ignition counter

GPIOB->ODR |= (1 << 14); // Turn on the LED connected to PB14 (led)

left = 0; // Reset the left indicator counter

right = 0; // Reset the right indicator counter

GPIOC->MODER&= ~(3<<12);// Clear the bits resp for the pin mode config for PC6

GPIOC->MODER |= (1 << 12); // Set PC6 as general-purpose output mode

GPIOC->ODR |= (1 << 6); // Turn on the LED connected to PC6 (led)

head = 0; // Reset the headlight mode

}

}

}

**static** **void** **tim3\_pc6\_set\_dutycycle**(uint32\_t freq, uint32\_t duty\_cycle, uint8\_t channel)

{

TIM8->ARR=((16000000/freq)-1);

**switch**(channel)

{

**case** 1:

TIM3->CCR1=(duty\_cycle\*((TIM3->ARR)+1)-1)/100;

**break**;

**case** 2:

TIM3->CCR4=(duty\_cycle\*((TIM3->ARR)+1)-1)/100;

**break**;

**default**:

**break**;

}

}

**static** **void** **pc6\_tim3\_pwm\_config**(**void** )

{

// Configure GPIO for PWM

RCC->AHB1ENR |= (1 << 2); // Enable GPIOC Clock

GPIOC->MODER |= (2 << 12); // Set PC6 to alternate function mode

GPIOC->AFR[0] = (2 << 24); // Set alternate function for PC6

RCC->APB1ENR |= (1 << 1); // Enable TIM3 Clock

//Configure TIM3 for PWM

TIM3->PSC=0;

TIM3->ARR=1000-1;

TIM3->CNT=0;

TIM3->CCMR1|=(1<<3);

TIM3->CCMR1|=(6<<4);

TIM3->CCER|=(3<<0);

TIM3->CR1|=(1<<0);

RCC->AHB1ENR |= (1<<0);

RCC->AHB1ENR |= (1<<1);

}

**static** **void** **pwm\_moder**(**void**)

{

// Configure GPIO for PWM mode

RCC->AHB1ENR |= (1 << 2); // Enable GPIOC Clock

GPIOC->MODER &= ~(3 << 12); // Clear bits 13 and 12 for PC6

GPIOC->MODER |= (2 << 12); // Set PC6 to alternate function mode

}

**static** **void** **timer\_delay**(**int** sec)

{

RCC->APB1ENR|=RCC\_APB1ENR\_TIM2EN; //Timer2 RCC Enable

TIM2->CR1=0; //Counter disabled

TIM2->PSC=16000-1; //Configuring 1kHZ clock

TIM2->ARR=500-1; //Producing 0.5s delay

TIM2->CR1|=(1<<0); //Counter enabled

**int** loop;

loop=sec\*2;

**for**(**int** k=1;k<loop;k++)

{

TIM2->CR1|=(1<<0);

**while**(!(TIM2->SR)&1)

{

}

TIM2->SR=0;

}

}

**static** **void** **ADC\_init**(**void**)

{

/\*

\* Configure ADC GPIO PIN PC2

\*/

RCC->AHB1ENR |=(1<<2); //Enable GPIO Clock

GPIOC->MODER |=(1<<4);

GPIOC->MODER |=(1<<5); //GPIOC->MODER |=(1<<4) |(1<<5);

/\*

\* Enable ADC module

\*/

RCC->APB2ENR |=(1<<8); //Clock Register for ADC1

ADC1->SQR1=0; //1 conversion

ADC1->SQR3 |=(0xC<<0); // Selecting Channel 12

ADC1->CR2 |=(1<<0);

}

**static** **void** **start\_conversion**(**void**)

{

//Enable Continuous Mode

ADC1->CR2|=(1<<1);

ADC1->CR2 |=(1U<<30); //START conversion

}

**static** uint32\_t **ADC\_result**(**void**)

{

**while**(!(ADC1->SR) & (1U<<1)); // Waiting for the conversion to complete

**return** (ADC1->DR);

}

**Test cases:**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| SI NO | SRS ID | Test ID | Test Scenario | Pre condition | Expected Result | Actual Result | Test Status | Remark |
| 1 |  | ST01 | Pres the Ignition Key "1"time" | 1. Board Power Up  2. Program Running  3. Ignition status ON | Ignition Status to High & Ignition LED to be on. ADC Value & UART status shown on Debug Live Expression window |  |  |  |
| 2 |  | ST02 | Pres the Ignition Key "2"time" | 1. Board Power Up  2. Program Running  3. Ignition LED ON  4. ADC value on the Live expression window  5. UART value on Live expression window | Ignition Status Off All other operations off |  |  |  |
| 3 |  | ST03 | Press the Right Indicator key 1st time | 1. Board Power Up  2. Program Running  3. Ignition LED ON  4.ADC Value & UART shown on Debug Live Expression window | Right indicator LED & Buzzer to blink with 0.5Hz frequency |  |  |  |
| 4 |  | ST04 | Press the Right Indicator key 2nd time | 1. Board Power Up  2. Program Running  3. ADC Value & UART shown on Debug Live Expression window  4. Right indicator LED & Buzzer to blinking with 0.5Hz frequency | Right indicator LED & Buzzer to be off |  |  |  |
| 5 |  | ST05 | Press the Left Indicator key “1” time | 1. Board Power Up  2. Program  Running  3. Ignition LED ON  4.ADC Value & UART shown on Debug Live Expression window | Left indicator LED & Buzzer to blink with 0.5Hz frequency |  |  |  |
| 6 |  | ST06 | Press the Left Indicator key “2” time | 1. Board Power Up  2. Program Running  3. ADC Value & UART shown on Debug Live Expression window  4.Left indicator LED & Buzzer to blinking with 0.5Hz frequency | Left indicator LED & Buzzer to be off |  |  |  |
| 7 |  | ST07 | Press the Light Switch "1" time | 1. Board Power Up  2. Program Running  3. Ignition LED ON  4.ADC Value & UART shown on Debug Live Expression window | The Low Beam Light to be activated in LED with 10% Duty Cycle |  |  |  |
| 8 |  | ST08 | Press the Light Switch "2" time | 1. Board Power Up  2. Program Running  3. Ignition LED ON  4.ADC Value & UART shown on Debug Live Expression window  5. Low Beam light ON | The High Beam Light to be activated in LED with 90% Duty Cycle |  |  |  |
| 9 |  | ST09 | Press the Light Switch "3" time | 1. Board Power Up  2. Program Running  3. Ignition LED ON  4.ADC Value & UART shown on Debug Live Expression window  5. High Beam light ON | The Parking Light to be activated. Left & right Indicator LED & Buzzer to blink with 2Hz frequency |  |  |  |
| 10 |  | ST010 | Press the Light Switch "4" time | 1. Board Power Up  2. Program Running  3. Ignition LED ON  4.ADC Value & UART shown on Debug Live Expression window  5. Parking LEDs and Buzzer ON | The Parking Light to be deactivated. Left & right Indicator LED & Buzzer to be OFF |  |  |  |
| 11 |  | ST011 | If the Left/Right Indicator is ON: Light Switch is pressed 3rd  Time | 1. Board Power Up  2. Program Running  3.ADC Value & UART shown on Debug Live Expression window  4. Left & right Indicator LED & Buzzer to blinking with 0.5HZ | The Indicator status LED & Buzzer continues to work & Parking doesn't get activated. |  |  |  |