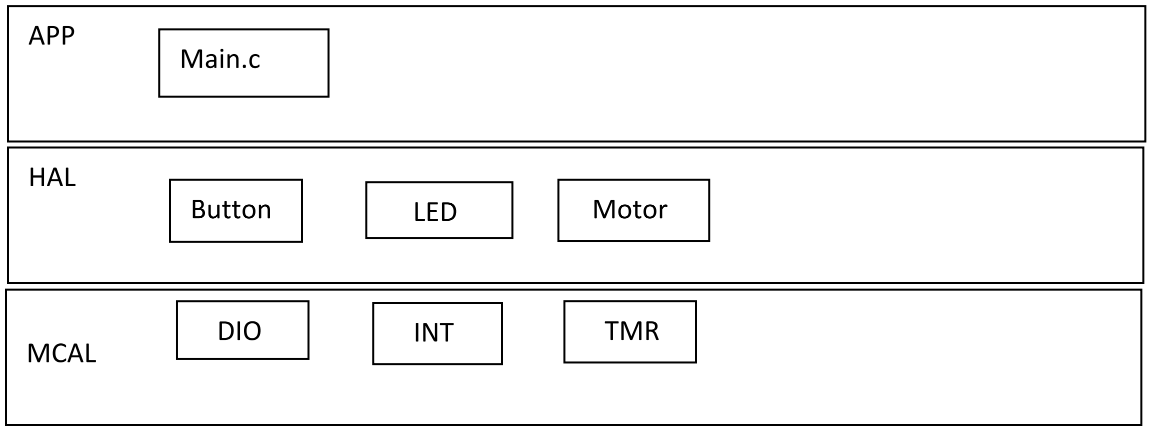
Author: Mustafa Mohammed Abdou Soliman

Project: moving car design

1. *Description*
    1. **Hardware components**:
        1. **Four** motors (**M1**, **M2**, **M3**, **M4**)
        2. **One** button to start (**PB1**)
        3. **One** button for stop (**PB2**)
        4. **Four** LEDs (**LED1**, **LED2**, **LED3**, **LED4**)

    2. **software Requirements**:
        1. The car **starts initially** from **0 speed**
        2. When **PB1** is **pressed**, the car will **move forward after 1 second**
        3. The car will move forward to **create the longest side of the rectangle for 3 seconds with 50% of its maximum speed**
        4. After finishing the first longest side the car will **stop for 0.5 seconds**, **rotate 90 degrees to the right**, and **stop for 0.5 second**
        5. The car will move to **create the short side** of the rectangle at **30% of its speed for 2 seconds**
        6. After finishing the shortest side the car will stop for **0.5 seconds**, **rotate 90 degrees to the right**, and **stop for 0.5 second**
        7. Steps **3 to 6** will be **repeated infinitely** until you press the **stop button (PB2)**
        8. **PB2** acts as a **sudden break**, and it has the highest priority

## Layered architecture:



1- Libraries: standard_types , common macros
2- MCAL: DIO , INT , TMR
3- HAL: LED , button , Motor
4- APP: main.c

1- DIO module contains (DIO_interface.h , DIO_private.h , DIO_config.h , DIO_program.c)

## DIO_interface.h

```c
/** TYPE DEFINITION FOR PIN DIRECTION TO RETURN ITS STATE **/
typedef enum { VALID_DIRECTION , NOT_VALID_DIRECTION } PinDirection_t ;

/** TYPE DEFINITION FOR PIN VALUE TO RETURN ITS STATE  **/
typedef enum { VALID_VALUE , NOT_VALID_VALUE } PinValue_t ;

/** TYPE DEFINITION FOR PIN READ STATUS  **/
typedef enum { VALID_READ , NOT_VALID_READ } PinRead_t ;

/*****************************************************************/
/* DESCRIBTION  : FUNCTION TO SET THE DIRECTION OF SPECIFIC PIN   */
/* INPUT   : PORT , PINID , DIRECTION                             */
/* RETURNS : PinDirection_t                                       */
/*****************************************************************/
PinDirection_t DIO_SETPINDIR(uint8_t portID , uint8_t pinID , uint8_t dir);


/*****************************************************************/
/* DESCRIBTION  : FUNCTION TO SET THE DIRECTION OF SPECIFIC PORT  */
/* INPUT   : PORTID , DIRECTION                                   */
/* RETURNS : VOID                                                 */
/*****************************************************************/
//void DIO_SETPORTDIR(uint8_t portID , uint8_t dir);


/*****************************************************************/
/* DESCRIBTION  : FUNCTION TO SET THE VALUE OF SPECIFIC PIN       */
/* INPUT   : PORT , PINID , DIRECTION                             */
/* RETURNS : PinValue_t                                           */
/*****************************************************************/
PinValue_t DIO_SETPINVAL(uint8_t portID , uint8_t pinID , uint8_t val);



/*****************************************************************/
/* DESCRIBTION  : FUNCTION TO GET THE VALUE OF SPECIFIC PIN       */
/* INPUT   : PORTID , PINID , POINTER TO SET THE VALUE IN IT      */
/* RETURNS : PinRead_t                                            */
/*****************************************************************/
PinRead_t DIO_READPIN(uint8_t portID , uint8_t pinID , uint8_t* val);
```

# INT_interface.h

```c
#define GLOBAL_INTERRUPT_STATE_ENABLE    1
#define GLOBAL_INTERRUPT_STATE_DISABLE   0

#define INT_TRIGGER_RISING_EDGE          0
#define INT_TRIGGER_FALLING_EDGE         1
#define INT_TRIGGER_ANY_CHANGE           2
#define INT_TRIGGER_LOW_LEVEL            3

/*******************************************************/
/** FUNCTION TO SET THE GLOBAL INTERRUPT ENABLE FLAG    */
/** ARGUMENTS  : VOID                                   */
/** RETURNS    : VOID                                   */
/*******************************************************/
void SET_GLOBALINTERRUPT(void);

/*******************************************************/
/** FUNCTION TO INITIALIZE INT0                         */
/** ARGUMENTS  : VOID                                   */
/** RETURNS    : VOID                                   */
/*******************************************************/
void INT0_INIT(void);
```

# TMR0_interface.h

```c
/** MACROS FOR THE CLOCK SOURCE  **/
#define INTERNAL_CLK_SRC                        0
#define EXTERNAL_CLK_SRC                        1


/** MACROS FOR EXTERNAL SOURCE OUNTING ACTION **/
#define FALLING_EDGE_CNT                        0
#define RISING_EDGE_CNT                         1

/** MACROS FOR DIFFERENT OPERATING MODES FOR TMR0 **/
#define TMR0_NORMAL_MODE                        0
#define TMR0_FASTPWM_NON_INVERTED_MODE          1
#define TMR0_FASTPWM_INVERTED_MODE              2
#define TMR_PHASE_CORRECT_NON_INVERTED_MODE     3
#define TMR_PHASE_CORRECT_INVERTED_MODE         4
#define TMR0_CTC_MODE                           5

/** MACROS FOR ACTION ON COMPARE  **/
#define SET_ON_COMPARE                          0
#define CLEAR_ON_COMPARE                        1
#define TOOGLE_ON_COMPARE                       2


/** MACROS TO CONFIGURE TMR0 PRESCALLER **/
#define NO_PRESCALER                            0
#define PRESCALER_8                             1
#define PRESCALER_64                            2
#define PRESCALER_256                           3
#define PRESCALER_1024                          4


/*****************************************************************/
/** FUNCTION TO INITIALIZE TMR0 WITH SOME CONFIGURATIONS        */
/** ARGUMENTS  : VOID                                           */
/** RETURNS    : VOID                                           */
/*****************************************************************/
void TMR0_INIT(void);


/********************************************************************************/
/** FUNCTION TO LET TIMER 0 START WORK BY ASSIGN PRESCALLER OR CLOCK SOURCE    */
/** ARGUMENTS  : VOID                                                          */
/** RETURNS    : VOID                                                          */
/********************************************************************************/
void TMR0_START(void);


/********************************************************************************/
/** FUNCTION TO STOP TIMER 0                                                   */
/** ARGUMENTS  : VOID                                                          */
/** RETURNS    : VOID                                                          */
/********************************************************************************/
void TMR0_STOP(void);
```

```c
/*******************************************************************************/
/** FUNCTION TO SET DELAY USING TIMER 0                                        */
/** ARGUMENTS  : TAKES DELAY IN ms                                             */
/** RETURNS    : VOID                                                          */
/*******************************************************************************/
void TMR0_DELAY_MS(uint32 DELAY_MS);
```

## 2- HAL: button (button_interface.h , button_config.h , button_private.h , button_program.c)

### Button_interface.h

```
/** TYPE DEFINITION FOR BUTTON RETURN ERROR */
typedef enum { Button_Notpressed  , Button_pressed} button_t;


/*******************************************************************/
/* FUNCTION TO INITIALIZE THE BUTTON                             */
/* ARGUMENTS : TAKES THE BUTTON PIN                              */
/* RETURN    : void                                              */
/*******************************************************************/
void Button_init(uint8_t Button_port , uint8_t Button_pin);


/*******************************************************************/
/* FUNCTION TO CHECK THE BUTTON STATUS PRESSED OR NOT            */
/* ARGUMENTS : TAKES THE BUTTON PIN                              */
/* RETURN    : RETURNS BUTTON_t type                             */
/*******************************************************************/
button_t Is_pressed(uint8_t Button_port , uint8_t Button_pin , uint8_t * value);
```

## LED ( LED_interface.h , LED_config.h , LED_program.c , LED_private.h)

### LED_interface.h

```
/****************************************************/
/** FUNCTION TO INITIALIZE A PIN                  **/
/** INPUT : LED PORT , LED PIN                    **/
/** RETURNS : VOID                                **/
/****************************************************/
void LED_INIT(uint8_t led_port , uint8_t ledpin);


/****************************************************/
/** FUNCTION TO SET A LED AS ON                   **/
/** INPUT : LED PORT , LED PIN                    **/
/** RETURNS : VOID                                **/
/****************************************************/
void LED_ON(uint8_t led_port , uint8_t ledpin);

/****************************************************/
/** FUNCTION TO SET A LED AS OFF                  **/
/** INPUT : LED PORT , LED PIN                    **/
/** RETURNS : VOID                                **/
/****************************************************/
void LED_OFF(uint8_t led_port , uint8_t ledpin);
```

## Motor_interface.h

```
void MOTOR_INIT(void);

void TURN_CLOCK_DIR(void);

void TURN_ANTI_CLOCK_DIR(void);

void TURN_OFF_MOTOR(void);

void APPLY_LOW_SPEED(void);

void APPLY_MIDIUM_SPEED(void);

void APPLY_HIGH_SPEED(void);
```