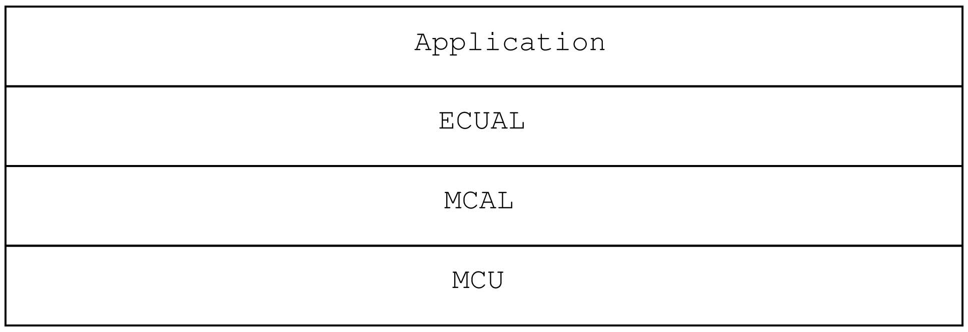
Table of Contents

Layered architecture



Systemx modules/drivers

1. Motor
2. Push button
3. LED
4. Timer
5. DIO
6. Utilities

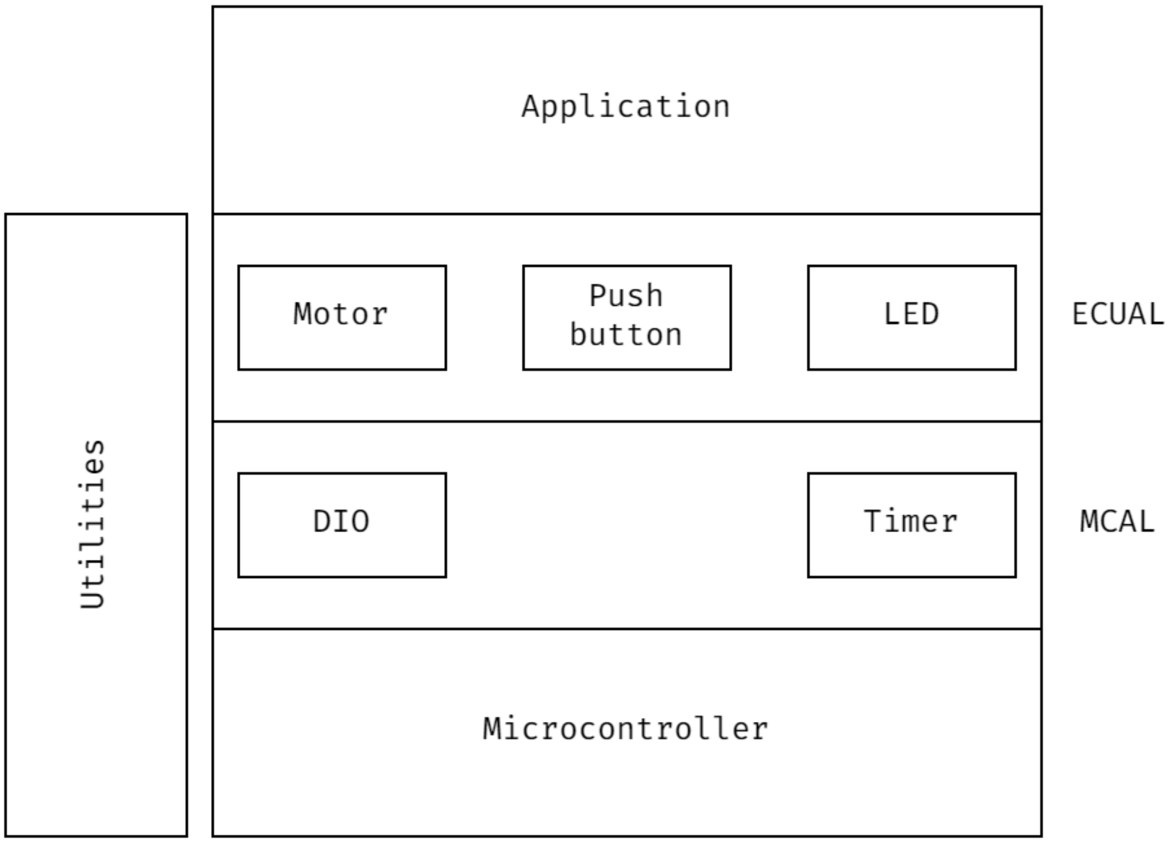

Application
Utilities
Motor
Push button
LED
ECUAL
DIO
Timer
MCAL
Microcontroller

Figure 1: img

System modules

APIs

DIO

```c
#ifndef MOVING_CAR_SYSTEM_DIO_H
#define MOVING_CAR_SYSTEM_DIO_H

#include "../../Utilities/registers.h"
#include "../../Utilities/std_types.h"
#include "../../Utilities/common_macros.h"

/**
 * @enum EN_DIO_ERROR_STATE
 * @brief Defines the state of DIO functions.
 */
typedef enum EN_DIO_ERROR_STATE {
    DIO_SUCCESS = 0, DIO_PORT_INVALID, DIO_DIRECTION_INVALID,
        DIO_PIN_INVALID
}EN_DIO_ERROR_STATE;

/**
 * @enum EN_DIO_DIRECTION
 * @brief Specifies the state of the pin.
 */
typedef enum EN_DIO_DIRECTION {
    DIO_INPUT = 0, DIO_OUTPUT
}EN_DIO_DIRECTION;

/**
 * @enum EN_DIO_PIN
 * @brief Specifies the number of pin.
 */
typedef enum EN_DIO_PIN {
   PIN0 = 0, PIN1, PIN2, PIN3, PIN4, PIN5, PIN6, PIN7, PIN8
}EN_DIO_PIN;

/**
 * @enum EN_DIO_PORT
 * @brief Specifies the port number.
 * the port number and returns the address of the corresponding port.
 */
typedef enum EN_DIO_PORT {
    PORT_A = 0, PORT_B, PORT_C, PORT_D
}EN_DIO_PORT;

/**
 * @enum EN_DIO_LEVEL
 * @brief Specifies the level of the pin.
 */
typedef enum EN_DIO_LEVEL {
    DIO_LOW = 0, DIO_HIGH
}EN_DIO_LEVEL;

/**
 * @struct DIO_Init_t
 * @brief Holds the configuration of a specific pin of a port.
 * @var DIO_Init_t::port
 * Member 'port' sets the port to be configured.
 * @var DIO_Init_t::pin
 * Member 'pin' sets the pin to be configured.
 * @var DIO_Init_t::direction
 * Member 'direction' sets the direction of the pin.
 * @var DIO_Init_t::pin_value
 * Member 'pin_value; contains the value of the pin when it's
   configured as input mode.
 * @var DIO_Init_t::port_value
 * Member 'port_value' contains the value to be written to the port
   register if the pin is configured as output.
 */
typedef struct DIO_Init_t {
    EN_DIO_PORT port;
    EN_DIO_PIN pin;
    EN_DIO_DIRECTION direction;
    union {
    uint8 pin_value;
    uint8 port_value;
    };
}DIO_Init_t;

/**
 * @brief Initializes the direction of the specified pin.
 * @param[in] p_config_struct Address of the configuration structure.
 * @return DIO_PORT_INVALID Port in invalid.
 * @return DIO_SUCCESS The pin initialization is a success.
 */
EN_DIO_ERROR_STATE DIO_Init(DIO_Init_t *p_config_struct);

/**
 * @brief Reads the state of a specific pin.
 * @param[in] p_config_struct Address of the configuration structure.
 * @return DIO_PORT_INVALID Port is invalid.
 * @return DIO_DIRECTION_INVALID Reading from a pin that is configured
   as output.
 * @return DIO_SUCCESS The read operation is a success.
 */
EN_DIO_ERROR_STATE DIO_ReadPin(DIO_Init_t *p_config_struct);

/**
```

```
 * @brief Write a specific level to a specified pin.
 * @param[in] p_config_struct Address of the configuration structure.
 * @return DIO_PORT_INVALID Port is invalid.
 * @return DIO_DIRECTION_INVALID Writing to a pin that is configured as
     input.
 * @return DIO_SUCCESS The write operation is a success.
 */
EN_DIO_ERROR_STATE DIO_WritePin(DIO_Init_t *p_config_struct);

/**
 * @brief Toggles the current level of a pin.
 * @param[in] p_config_struct Address of the configuration structure.
 *  @return DIO_PORT_INVALID Port is invalid.
 * @return DIO_DIRECTION_INVALID Toggle a pin that is configured as
    input.
 * @return DIO_SUCCESS The toggle operation is a success.
 */
EN_DIO_ERROR_STATE DIO_TogglePin(DIO_Init_t *p_config_struct);

#endif //MOVING_CAR_SYSTEM_DIO_H
```

Timer

```
//
// Created by khale on 2023-04-05.
//

#ifndef MOVING_CAR_SYSTEM_TIMER_H
#define MOVING_CAR_SYSTEM_TIMER_H

#include "../../Utilities/registers.h"
#include "../../Utilities/std_types.h"
#include "../../Utilities/common_macros.h"

typedef enum
{
    NORMAL_WG, PWM_WG, CTC_WG, FAST_PWM_WG
}TIMER0_WaveFormGeneration;

/* Clock Selection. */
typedef enum
{
    NO_CLOCK, F_CPU_CLOCK, F_CPU_8, F_CPU_64, F_CPU_256, F_CPU_1024
}TIMER0_ClockSelect;

typedef enum
{
    INTERRUPT_DISABLED, INTERRUPT_ENABLED
}TIMER0_InterruptMode;
```

```

typedef enum
{
    TIMER0_OVERFLOW, TIMER0_COMPARE
}TIMER0_MODE;

/* Timer configuration structure. */
typedef struct
{
    TIMER0_MODE timerMode;
    TIMER0_ClockSelect timerClock;
    TIMER0_WaveFormGeneration timerWaveGeneration;
    uint8 TIMER0_Reg;
    uint8 TIMER0_CompareValue;
    TIMER0_InterruptMode InterruptMode;
}TIMER0_Config;

/* Initialize and start Timer0.
 * Global interrupt is enabled when interrupt mode is chosen.
 */
void Timer0_Init(TIMER0_Config *UserConfig);

/* De-initialize Timer0 registers and turn off the timer. */
void Timer0_DeInit(void);


void Timer0_setCallBack(void (*a_ptr) (void));

#endif //MOVING_CAR_SYSTEM_TIMER_H
```

LED

```
#ifndef MOVING_CAR_SYSTEM_LED_H
#define MOVING_CAR_SYSTEM_LED_H

#include "../../MCAL/DIO/dio.h"

/**
 * @struct LED_Init_t
 * @brief Holds the port number and the pin number of the LED.
 * @var LED_Init_t::port
 * Member 'port' specifies the port number.
 * @var LED_Init_t::pin
 * Member 'pin' specifies the pin number.
 */
typedef struct LED_Init_Typedef {
    EN_DIO_PORT port;
    EN_DIO_PIN pin;
}LED_Init_t;
```

```

/**
 * @brief Initializes the pin attached to the LED.
 * @param p_config_struct Address of the configuration structure.
 */
void LED_Init(LED_Init_t *p_config_struct);

/**
 * @brief Turns the LED on.
 * @param p_config_struct Address of the configuration structure.
 */
void LED_On(LED_Init_t *p_config_struct);

/**
 * @brief Turns the LED off.
 * @param p_config_struct Address of the configuration structure.
 */
void LED_Off(LED_Init_t *p_config_struct);

#endif //MOVING_CAR_SYSTEM_LED_H
```

Motor

```
#ifndef MOVING_CAR_SYSTEM_MOTOR_H
#define MOVING_CAR_SYSTEM_MOTOR_H

#include "../../MCAL/DIO/dio.h"
#include "../../MCAL/Timer/timer.h"

/**
 * @struct MOTOR_Init_t
 * @var MOTOR_Init_t::port
 * Member 'port' specifies the port number that the motor is attached
    to.
 * @var MOTOR_Init_t::pin
 * Member 'pin' specifies the pin number that the motor is attached to.
 */
typedef struct MOTOR_Init_Typedef {
    EN_DIO_PORT port;
    EN_DIO_PIN pin;
}MOTOR_Init_t;

/**
 * @enum MOTOR_Direction_t
 * @brief Specifies the direction of rotation of the motor.
 */
typedef enum MOTOR_Direction_Typedef {
    CW = 0, CCW
}MOTOR_Direction_t;
```

```c

/**
 * @brief Initializes the state of the pin the motor is attached to.
 * @param p_config_struct[in] Address of the configuration structure.
 */
void MOTOR_Init(MOTOR_Init_t *p_config_struct);

/**
 * @brief Moves the motor in the forward direction.
 * @param p_config_struct[in] Address of the configuration structure.
 */
void MOTOR_Forward(MOTOR_Init_t *p_config_struct);

/**
 * @brief Stops the movement of the motors.
 * @param p_config_struct[in] Address of the configuration structure.
 */
void MOTOR_Stop(MOTOR_Init_t *p_config_struct);

/**
 * @brief Rotates the motor is the specified direction.
 * @param motor_direction Direction of the rotation.
 */
void MOTOR_Rotation(MOTOR_Direction_t motor_direction);

/**
 * @brief Specifies the speed of the motor.
 * @param speed[in] The speed of the motor.
 */
void MOTOR_Speed(uint8_t speed);

#endif //MOVING_CAR_SYSTEM_MOTOR_H
```

Push button

```c
#ifndef MOVING_CAR_SYSTEM_PUSH_BUTTON_H
#define MOVING_CAR_SYSTEM_PUSH_BUTTON_H

#include "../../MCAL/DIO/dio.h"

/**
 * @struct PB_Init_t
 * @var PB_Init_t::port
 * Member 'port' specifies the port which the push button is connected
   to.
 * @var PB_Init::pin
 * Member 'pin' specifies the pin number which the push button is
   connected to.
 */
```

```c
typedef struct PB_Init_Typedef {
    EN_DIO_PORT port;
    EN_DIO_PIN pin;
}PB_Init_t;

/**
 * @enum EN_PB_LEVEL
 * @brief Specifies the state of push button.
 */
typedef enum EN_PB_LEVEL {
    PB_LOW = 0, PB_HIGH
}EN_PB_LEVEL;

/**
 * @brief Initializes the state of the pin connected to the push button
     .
 * @param p_config_struct Address of the configuration structure.
 */
void PB_Init(PB_Init_t *p_config_struct);

/**
 * @brief Reads the current state of the push button.
 * @param p_config_struct Address of the configuration structure.
 * @return The current state of the push button.
 */
EN_PB_LEVEL PB_ReadState(PB_Init_t *p_config_struct);

#endif //MOVING_CAR_SYSTEM_PUSH_BUTTON_H
```