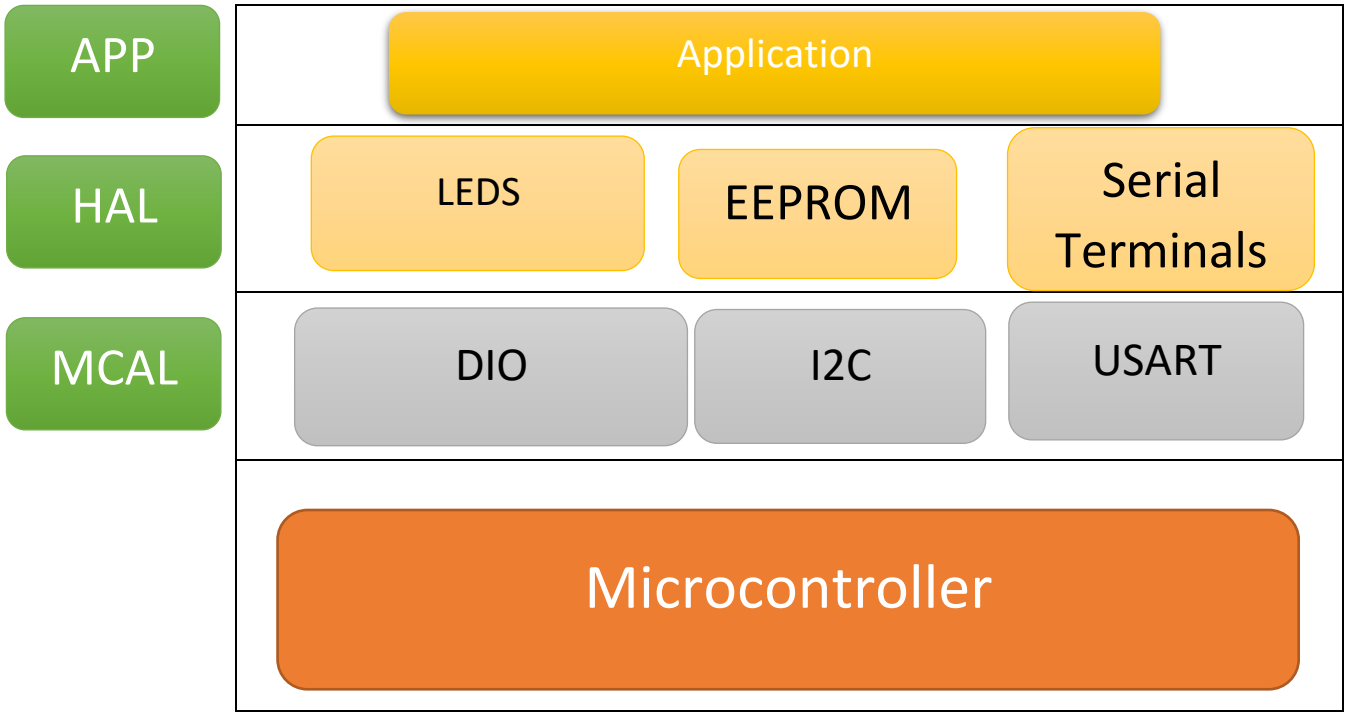

1. MCAL

1.1. DIO APIs

```
/* This Function determine the Pin direction Whether input or output */
E_status  DIO_SetPinDirection(ST_DIO_config_t * Configurations);
typedef ST_DIO_config_t{
                        EN_portname_t port_name;
                        EN_portno_t pin_no;
                        EN_pinstate_t state;            // input or output
}ST_DIO_config_t;
E_status  DIO_SetPinValue(uint8_t au8_port_no, uint8_t au8_pin_no, uint8_t au8_value);
E_status  DIO_GetPinValue(uint8_t au8_port_no, uint8_t au8_pin_no, uint8_t * data);
E_status  DIO_TogglePin(uint8_t au8_port_no, uint8_t au8_pin_no);
```

1.2. Usart APIs

```
error_state usart_Init(ST_UART_config_t * USART_InitStruct);

error_state usart_SendData(uint8_t data_transmitted);

uint8_t usart_ReceiveData(void);

error_state usart_SendString(uint8_t *str);

uint8_t * usart_ReceiveString(uint8_t * au8data ,uint8_t terminating_character);
```

```
typedef struct ST_UART_config_t{
    uint16_t USART_BaudRate;
    uint8_t USART_WordLength;
    uint8_t USART_StopBits;
    uint8_t USART_Parity;
     uint8_t USART_Mode;
}ST_UART_config_t;
```

1.3. I2C APIs

```
I2C_error_states  TWI_MasterInit(ST_I2C_config_t * configuration);
void TWI_SlaveInit(uint8_t Copy_u8Address);

void  TWI_VidInit(void);
void  TWI_VoidStartCondition(void);
void  TWI_VoidRepeatedStartCondition(void);
void TWI_VoidMaster_Send_Slave_Address_With_Write(uint8_t Address, uint8_t Copy_Rw);
uint8_t    TWI_U8ReadACK(void);
uint8_t    TWI_U8ReadNACK(void);
uint8_t    TWI_uint8_tGetStatus(void);
void  TWI_VoidMaster_Write_Byte_To_Slave(uint8_t data);
uint8_t    TWI_VoidMaster_Reading_Byte_From_Slave(uint8_t No_ofBytes);
void  TWI_VoidStopCondition(void);
I2C_error_states  TWI_VoidMaster1_Write_Byte_To_Slave(uint8_t SlaveAddress ,
 uint8_t InternalReg, uint8_t Data);
uint8_t  TWI_VoidMaster1_Reading_Byte_From_Slave(uint8_t SlaveAddress ,
 uint8_t InternalReg);
```

```
typedef struct
{
  uint32_t I2C_DivisionFactor;          /*!< Specifies the clock frequency.
This parameter must be set to a value lower than 400kHz */
  uint8_t I2C_Mode;                /*!< Specifies the I2C mode With or without Interrupt. */
  uint8_t I2C_Ack;                 /*!< Enables or disables the acknowledgement.  */
  uint8_t Prescaller;               /*  This Specifies the Prescaller */
}ST_I2C_config_t;
```

```
typedef enum
{
    PRESCALLER_SEL_ERROR,
    I2C_MODE_ERROR,
    ACK_ERROR,
    I2C_START_ERROR,
    I2C_MT_SLA_ACK_ERROR,
    I2C_MT_DATA_ACK_ERROR,
    I2C_REP_START_ERROR,
    I2C_MR_SLA_ACK_ERROR,
    I2C_MR_DATA_NOT_ACK_ERROR,
    I2C_NO_ERROR
}I2C_error_states;
```

2. HAL

a. EEPROM

```
void EEPROM_VoidInit(void);
void EEPROM_VoidWriteDataByte(uint8_t SlaveAddress , uint8_t InternalReg, uint8_t Data);
uint8_t EEPROM_u8ReadDataByte(uint8_t SlaveAddress , uint8_t InternalReg);
```