สถาบันเทคโนโลยีไทย-ญี่ปุ่น
Thai-Nichi Institute of Technology
泰日工業大学
สร้างนักคิด ผลิตนักปฏิบัติ สร้างนักประดิษฐ์ ผลิตนักบริหาร

10th Anniversary

transtron

STMicrocontroller Training

Basic GPIO

Titichaya Thanamitsombon

Goragod Pongthanisorn

Computer Engineering Department, Faculty of Engineer

Thai-Nichi Institute of Technology

Date 11 Oct 2017

Time xx:xx – xx:xx

Brief Intro to HAL Driver

Hal drivers are library designed by ST that provide API for peripheral management. HAL has provided a several features for STMicrocontroller programming

- Cross-family portability
- Three API programming model (Polling, Interrupt and DMA)
- RTOS Compliant
- Data structure Peripheral Configuration
- Multi-Instance that allow concurrent API call
- User-callback mechanism is provided
- Timeout event
- Instance lock mechanism, prevent multiple spurious hardware access

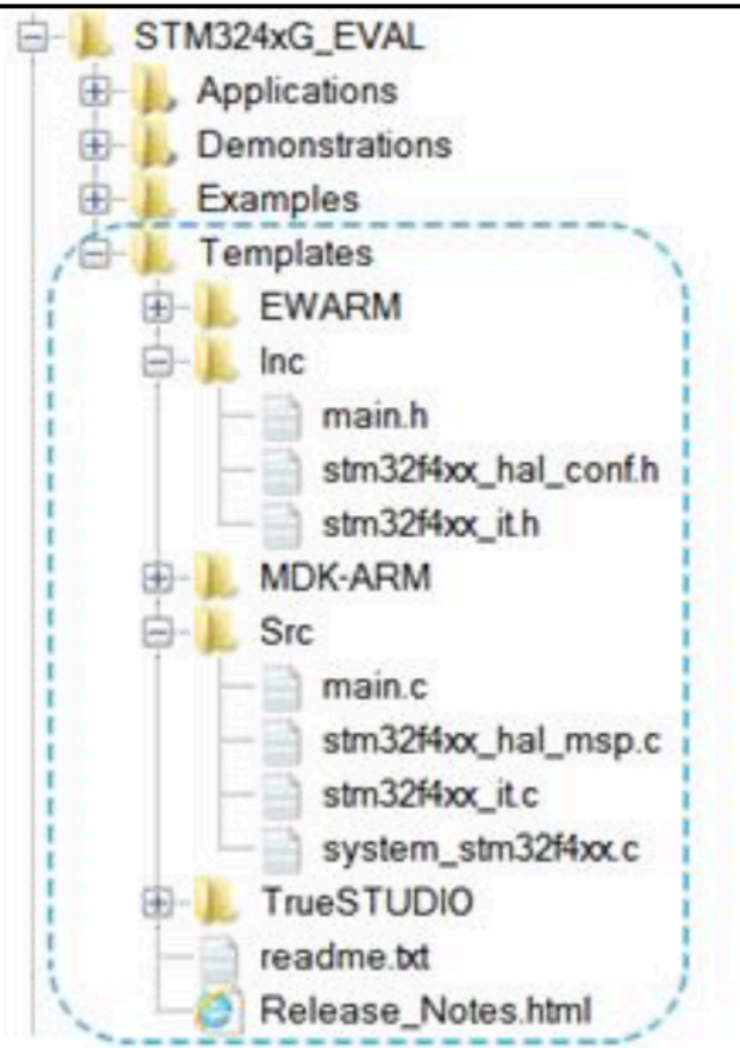
HAL Driver Files

stm32f4xx_hal_ppp.c	Main peripheral/module driver file. It includes the APIs that are common to all STM32 devices. *Example: stm32f4xx_hal_adc.c, stm32f4xx_hal_irda.c, …*
stm32f4xx_hal_ppp.h	Header file of the main driver C file It includes common data, handle and enumeration structures, define statements and macros, as well as the exported generic APIs. *Example: stm32f4xx_hal_adc.h, stm32f4xx_hal_irda.h, …*
stm32f4xx_hal_ppp_ex.c	Extension file of a peripheral/module driver. It includes the specific APIs for a given part number or family, as well as the newly defined APIs that overwrite the default generic APIs if the internal process is implemented in different way. *Example: stm32f4xx_hal_adc_ex.c,stm32f4xx_hal_dma_ex.c, …*
stm32f4xx_hal_ppp_ex.h	Header file of the extension C file. It includes the specific data and enumeration structures, define statements and macros, as well as the exported device part number specific APIs *Example: stm32f4xx_hal_adc_ex.h, stm32f4xx_hal_dma_ex.h, …*

HAL

system_stm32f4xx.c	This file contains SystemInit() which is called at startup just after reset and before branching to the main program. It does not configure the system clock at startup (contrary to the standard library). This must be performed using the HAL APIs in the user files. It allows to: • relocate the vector table in internal SRAM. • configure FSMC/FMC peripheral (when available) to use as data memory the external SRAM or SDRAM mounted on the evaluation board.
startup_stm32f4xx.s	Toolchain specific file that contains reset handler and exception vectors. For some toolchains, it allows adapting the stack/heap size to fit the application requirements.
stm32f4xx_flash.icf (optional)	Linker file for EWARM toolchain allowing mainly to adapt the stack/heap size to fit the application requirements.
stm32f4xx_hal_msp.c	This file contains the MSP initialization and de-initialization (main routine and callbacks) of the peripheral used in the user application.
stm32f4xx_hal_conf.h	This file allows the user to customize the HAL drivers for a specific application. It is not mandatory to modify this configuration. The application can use the default configuration without any modification.
stm32f4xx_it.c/.h	This file contains the exceptions handler and peripherals interrupt service routine, and calls HAL_IncTick() at regular time intervals to increment a local variable (declared in*stm32f4x_hal.c*) used as HAL timebase. By default, this function is called each 1ms in Systick ISR. . The PPP_IRQHandler() routine must call HAL_PPP_IRQHandler() if an interrupt based process is used within the application.
main.c/.h	This file contains the main program routine, mainly: • the call to HAL_Init() • assert_failed() implementation • system clock configuration • peripheral HAL initialization and user application code.

HAL Driver Files on IDE



STM32L152 Discovery Overview

Figure 2. Hardware block diagram

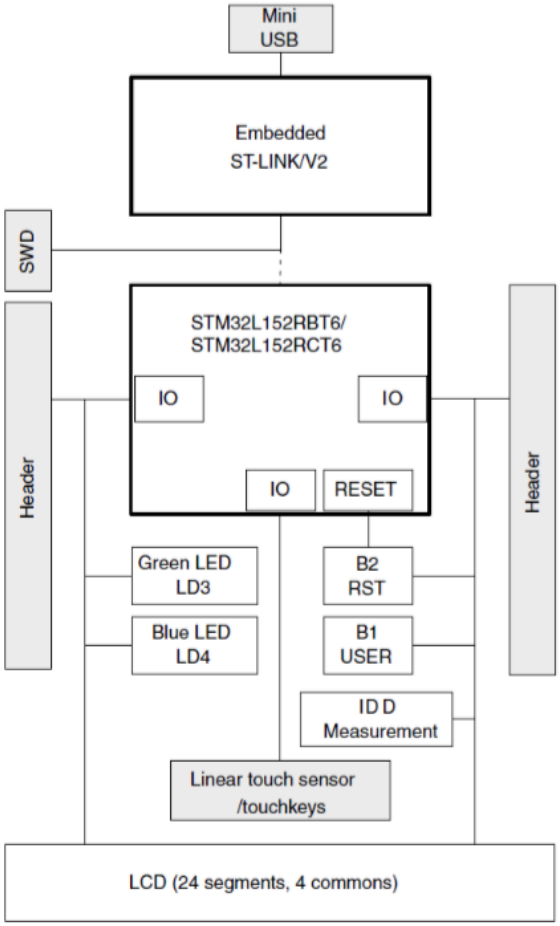
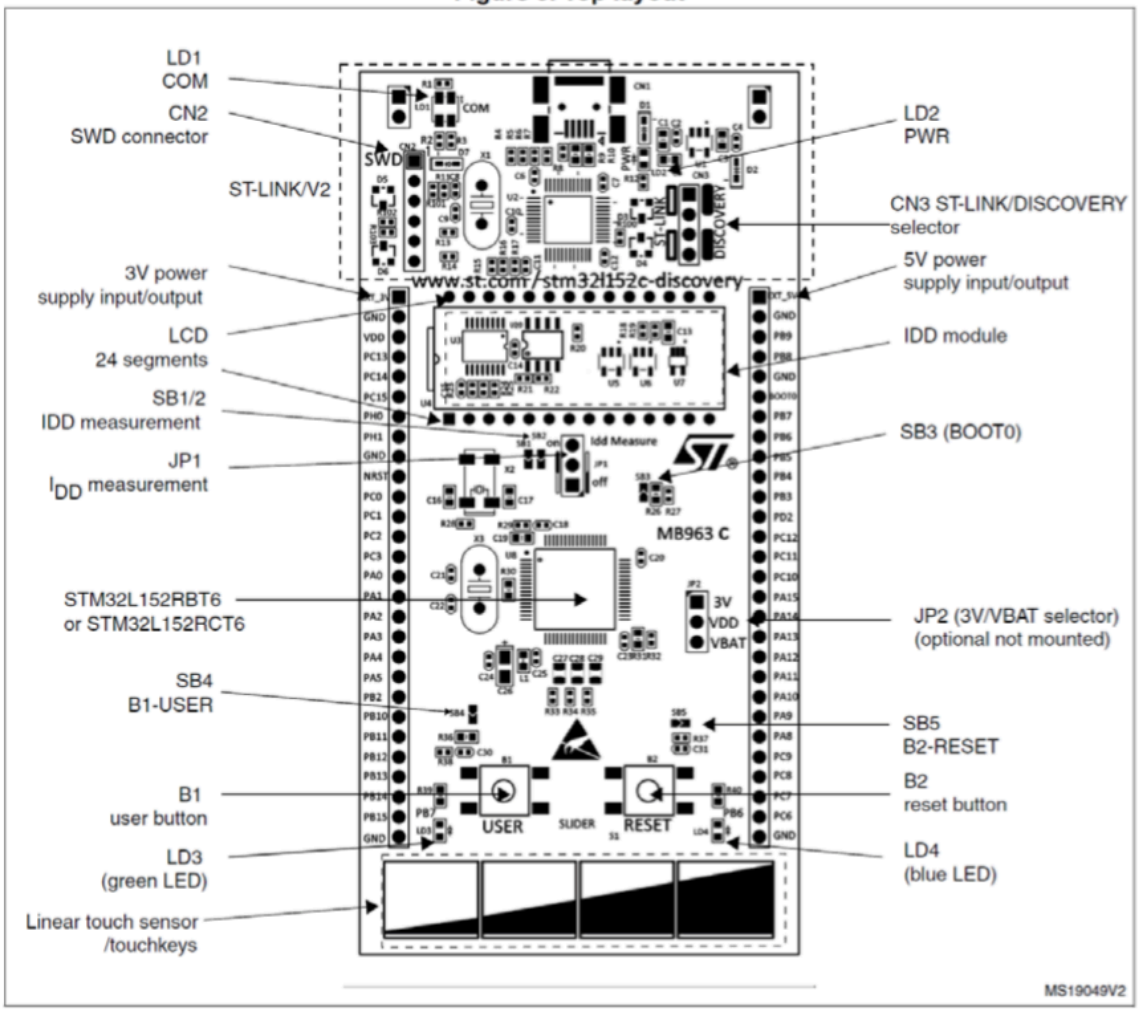


Figure 3. Top layout

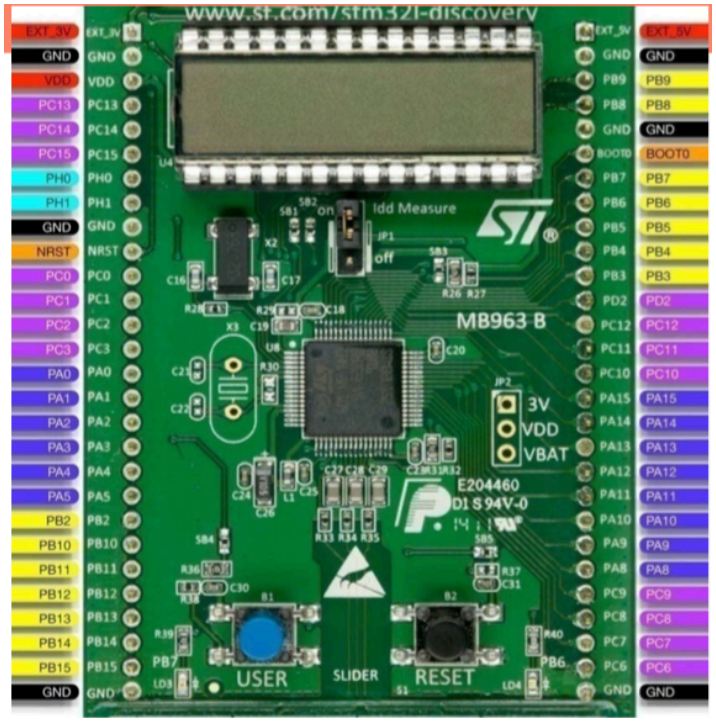


Pin and Port

STM32 catagorized each pin by

- PORT eg. A,B,C...H
- PIN eg. 1,2,3 16

Some of its pin can be used as special function (USART, SPI, I2C etc)



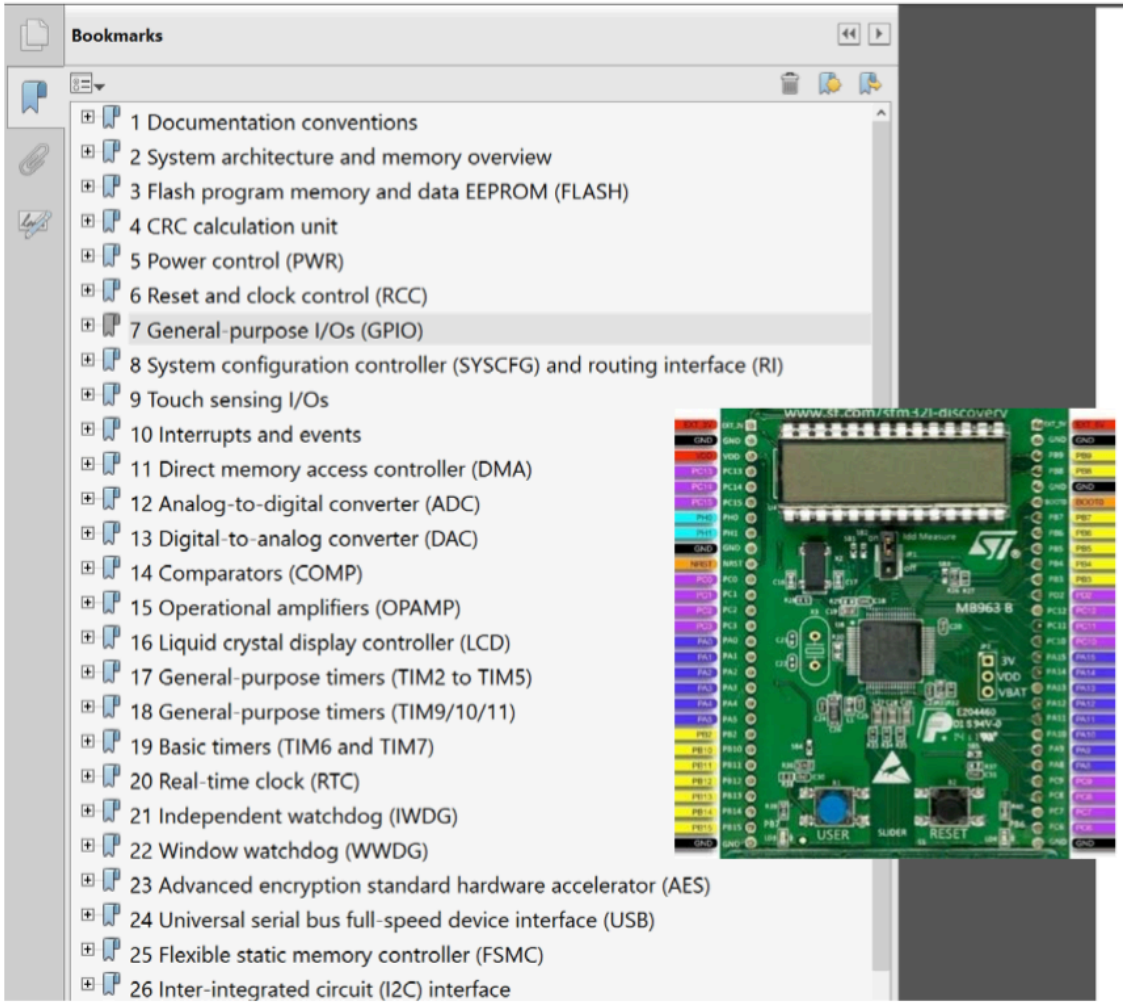
Available pins

Special pins

No pin out

General-purpose I/Os (GPIO)

General Purpose Input/Output



7 General-purpose I/Os (GPIO)

This section applies to the whole STM32L1xxxx family, unless otherwise specified.

7.1 GPIO introduction

Each general-purpose I/O port has four 32-bit configuration registers (GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR and GPIOx_PUPDR), two 32-bit data registers (GPIOx_IDR and GPIOx_ODR), a 32-bit set/reset register (GPIOx_BSRR), a 32-bit locking register (GPIOx_LCKR) and two 32-bit alternate function selection register (GPIOx_AFRH and GPIOx_AFRL).

7.2 GPIO main features

- Up to 16 I/Os under control
- Output states: push-pull or open drain + pull-up/down
- Output data from output data register (GPIOx_ODR) or peripheral (alternate function output)
- Speed selection for each I/O
- Input states: floating, pull-up/down, analog
- Input data to input data register (GPIOx_IDR) or peripheral (alternate function input)
- Bit set and reset register (GPIOx_BSRR) for bitwise write access to GPIOx_ODR
- Locking mechanism (GPIOx_LCKR) provided to freeze the I/O configuration
- Analog function
- Alternate function input/output selection registers (at most 16 AFs per I/O)
- Fast toggle capable of changing every two clock cycles
- Highly flexible pin multiplexing allows the use of I/O pins as GPIOs or as one of several peripheral functions

7.3 GPIO functional description

Subject to the specific hardware characteristics of each I/O port listed in the datasheet, each port bit of the general-purpose I/O (GPIO) ports can be individually configured by software in several modes:

- Input floating
- Input pull-up
- Input-pull-down
- Analog
- Output open-drain with pull-up or pull-down capability
- Output push-pull with pull-up or pull-down capability
- Alternate function push-pull with pull-up or pull-down capability
- Alternate function open-drain with pull-up or pull-down capability

Basic GPIO Control

Utilization of GPIO on STM32 can be done by these following steps

1. Enable Clock Source for bus
2. Configure desire pin (Mode, Speed, Port etc.)
3. Call function to manipulate input and output

These files are required

1. stm32l1xx_hal_rcc
2. stm32l1xx_hal_gpio

1. Enable Clock Source for bus

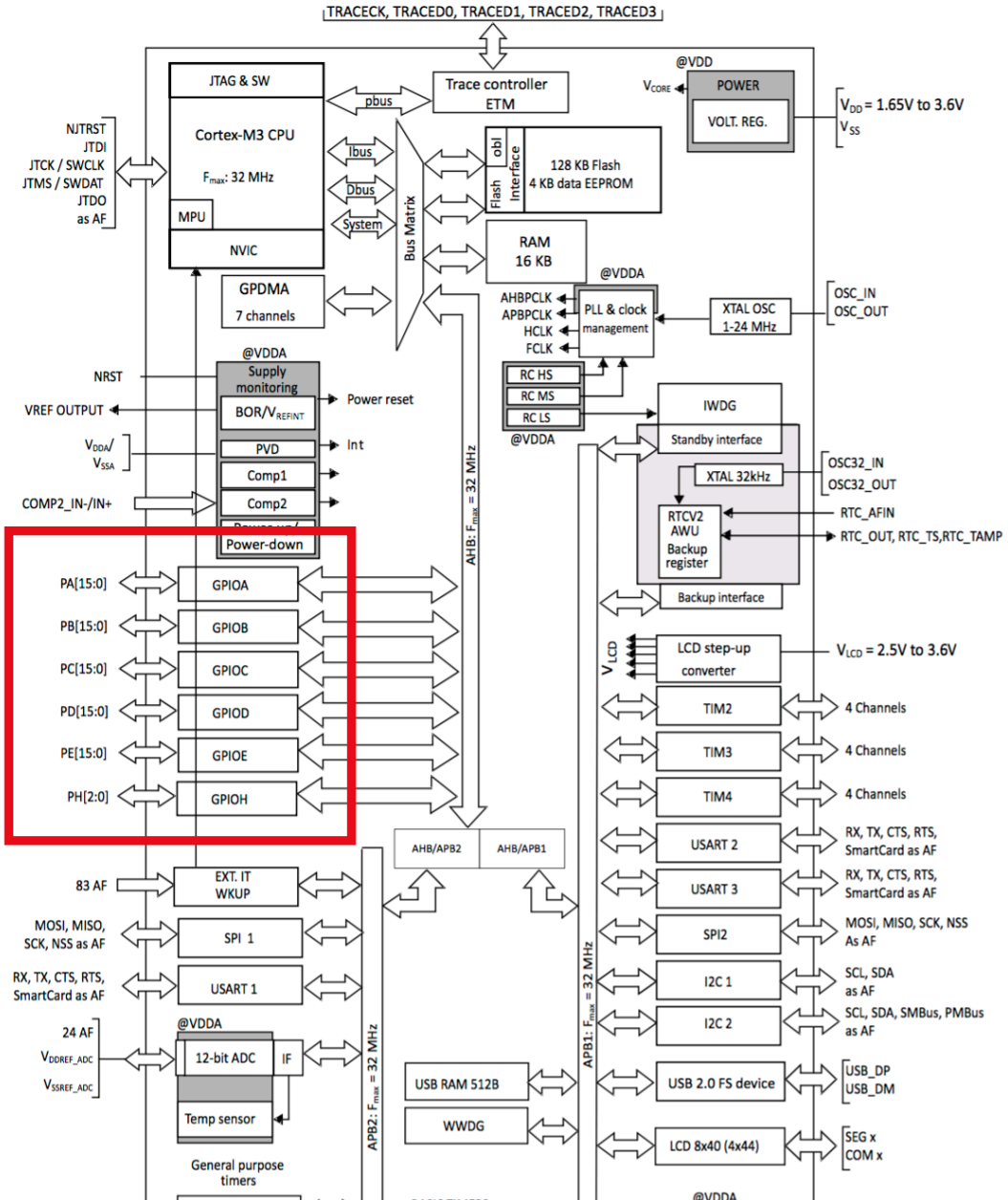
STM32 contain 2 type of bus

- Advance High-Speed Bus (AHB)
- Advance Peripheral Bus (APB)

All of buses are connected to Bus Matrix then connected to processor

To enable target peripheral, you need to specify what bus are that peripheral is connected

Enable bus clock source to enable target peripheral



1. Enable Clock Source for bus

RCC Register controller is embedded into STM32 processor to manage system and peripheral clock

- RCC_CR (Clock Control Reg)
- RCC_ICSCR (Internal Clock Source Calibration Reg)
- RCC_CFGR (Clock Configuration Reg)
- RCC_ICR (Clock Interrupt Reg)
- RCC_AHBRSTR (AHB Peripheral Reset Reg)
- RCC_APB2RSTR (APB2 Peripheral Reset Reg)
- RCC_APB1RSTR (APB1 Peripheral Reset Reg)
- **RCC_AHBENR (AHB Peripheral Enable Reg)**
- **RCC_APB2ENR (APB2 Peripheral Enable Reg)**
- **RCC_APB1ENR (APB1 Peripheral Enable Reg)**
- RCC_AHBLPENR (AHB Peripheral Low-Power Enable Reg)
- RCC_APB2LPENR (APB2 Peripheral Low-Power Enable Reg)
- RCC_APB1LPENR (APB1 Peripheral Low-Power Enable Reg)

1. Enable Clock Source for bus

6.3.8 AHB peripheral clock enable register (RCC_AHBENR)

Address offset: 0x1C

Reset value: 0x0000 8000

Access: no wait state, word, half-word and byte access

Note: When the peripheral clock is not active, the peripheral register values may not be readable by software and the returned value is always 0x0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	FSMC EN	Reserved		AES EN	Res.	DMA2EN	DMA1EN	Reserved							
	rw			rw		rw	rw								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLITF EN	Reserved		CRCEN	Reserved				GPIOG EN	GPIOF EN	GPIOH EN	GPIOE EN	GPIOD EN	GPIOC EN	GPIOB EN	GPIOA EN
rw			rw					rw	rw	rw	rw	rw	rw	rw	rw

1. Enable Clock Source for bus

To enable AHB Bus Clock, these function on stm32l1xx_hal_rcc.h can be used

```
#define __HAL_RCC_GPIOA_CLK_ENABLE()   do { \
                                         __IO uint32_t tmpreg; \
                                         SET_BIT(RCC->AHBENR, RCC_AHBENR_GPIOAEN);\
                                         /* Delay after an RCC peripheral clock enabling */\
                                         tmpreg = READ_BIT(RCC->AHBENR, RCC_AHBENR_GPIOAEN);\
                                         UNUSED(tmpreg); \
                                       } while(0)
#define __HAL_RCC_GPIOB_CLK_ENABLE()   do { \
                                         __IO uint32_t tmpreg; \
                                         SET_BIT(RCC->AHBENR, RCC_AHBENR_GPIOBEN);\
                                         /* Delay after an RCC peripheral clock enabling */\
                                         tmpreg = READ_BIT(RCC->AHBENR, RCC_AHBENR_GPIOBEN);\
                                         UNUSED(tmpreg); \
                                       } while(0)
#define __HAL_RCC_GPIOC_CLK_ENABLE()   do { \
```

2. Configure desire pin

A utilization of GPIO need following field to be specific

1. Mode (GPIOx_MODER)
2. Otype (GPIOx_OTYPER)
3. Speed (GPIOx_OSPEEDR)
4. Pin
5. Port

HAL drivers provide a data structure to configure these GPIO on stm32l1xx_hal_gpio.h

2. Configure desire pin

GPIO configure

```
typedef struct
{
  uint32_t Pin;

  uint32_t Mode;

  uint32_t Pull;

  uint32_t Speed;

  uint32_t Alternate;

}GPIO_InitTypeDef;
```

```
#define GPIO_PIN_0                 ((uint16_t)0x0001)  /* Pin 0 selected    */
#define GPIO_PIN_1                 ((uint16_t)0x0002)  /* Pin 1 selected    */
#define GPIO_PIN_2                 ((uint16_t)0x0004)  /* Pin 2 selected    */
#define GPIO_PIN_3                 ((uint16_t)0x0008)  /* Pin 3 selected    */
#define GPIO_PIN_4                 ((uint16_t)0x0010)  /* Pin 4 selected    */
#define GPIO_PIN_5                 ((uint16_t)0x0020)  /* Pin 5 selected    */
#define GPIO_PIN_6                 ((uint16_t)0x0040)  /* Pin 6 selected    */
#define GPIO_PIN_7                 ((uint16_t)0x0080)  /* Pin 7 selected    */
#define GPIO_PIN_8                 ((uint16_t)0x0100)  /* Pin 8 selected    */
#define GPIO_PIN_9                 ((uint16_t)0x0200)  /* Pin 9 selected    */
#define GPIO_PIN_10                ((uint16_t)0x0400)  /* Pin 10 selected   */
#define GPIO_PIN_11                ((uint16_t)0x0800)  /* Pin 11 selected   */
#define GPIO_PIN_12                ((uint16_t)0x1000)  /* Pin 12 selected   */
#define GPIO_PIN_13                ((uint16_t)0x2000)  /* Pin 13 selected   */
#define GPIO_PIN_14                ((uint16_t)0x4000)  /* Pin 14 selected   */
#define GPIO_PIN_15                ((uint16_t)0x8000)  /* Pin 15 selected   */
#define GPIO_PIN_All               ((uint16_t)0xFFFF)  /* All pins selected */
```

```
#define  GPIO_MODE_INPUT                        ((uint32_t)0x00000000)
#define  GPIO_MODE_OUTPUT_PP                    ((uint32_t)0x00000001)
#define  GPIO_MODE_OUTPUT_OD                    ((uint32_t)0x00000011)
#define  GPIO_MODE_AF_PP                        ((uint32_t)0x00000002)
#define  GPIO_MODE_AF_OD                        ((uint32_t)0x00000012)

#define  GPIO_MODE_ANALOG                       ((uint32_t)0x00000003)

#define  GPIO_MODE_IT_RISING                    ((uint32_t)0x10110000)
#define  GPIO_MODE_IT_FALLING                   ((uint32_t)0x10210000)
#define  GPIO_MODE_IT_RISING_FALLING            ((uint32_t)0x10310000)

#define  GPIO_MODE_EVT_RISING                   ((uint32_t)0x10120000)
#define  GPIO_MODE_EVT_FALLING                  ((uint32_t)0x10220000)
#define  GPIO_MODE_EVT_RISING_FALLING           ((uint32_t)0x10320000)
```

```
#define  GPIO_SPEED_FREQ_LOW         ((uint32_t)0x00000000)
#define  GPIO_SPEED_FREQ_MEDIUM      ((uint32_t)0x00000001)
#define  GPIO_SPEED_FREQ_HIGH        ((uint32_t)0x00000002)
#define  GPIO_SPEED_FREQ_VERY_HIGH   ((uint32_t)0x00000003)
```

Example for GPIOB6 Configuration

```
void GPIO_Config(void)
{
  GPIO_InitTypeDef GPIO_InitStructure;
  __HAL_RCC_GPIOB_CLK_ENABLE();

  GPIO_InitStructure.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStructure.Pin = GPIO_PIN_6;
  GPIO_InitStructure.Pull = GPIO_NOPULL;
  GPIO_InitStructure.Speed = GPIO_SPEED_FREQ_HIGH;
  HAL_GPIO_Init(GPIOB, &GPIO_InitStructure);

}
```

3. Call function to manipulate input and output

```
/* IO operation functions *****************************************************/
GPIO_PinState     HAL_GPIO_ReadPin(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin);
void              HAL_GPIO_WritePin(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin, GPIO_PinState PinState);
void              HAL_GPIO_TogglePin(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin);
HAL_StatusTypeDef HAL_GPIO_LockPin(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin);
void              HAL_GPIO_EXTI_IRQHandler(uint16_t GPIO_Pin);
void              HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin);
```

GPIO_TypeDef can be

```
#define GPIOA                ((GPIO_TypeDef *) GPIOA_BASE)
#define GPIOB                ((GPIO_TypeDef *) GPIOB_BASE)
#define GPIOC                ((GPIO_TypeDef *) GPIOC_BASE)
#define GPIOD                ((GPIO_TypeDef *) GPIOD_BASE)
#define GPIOE                ((GPIO_TypeDef *) GPIOE_BASE)
#define GPIOH                ((GPIO_TypeDef *) GPIOH_BASE)
```

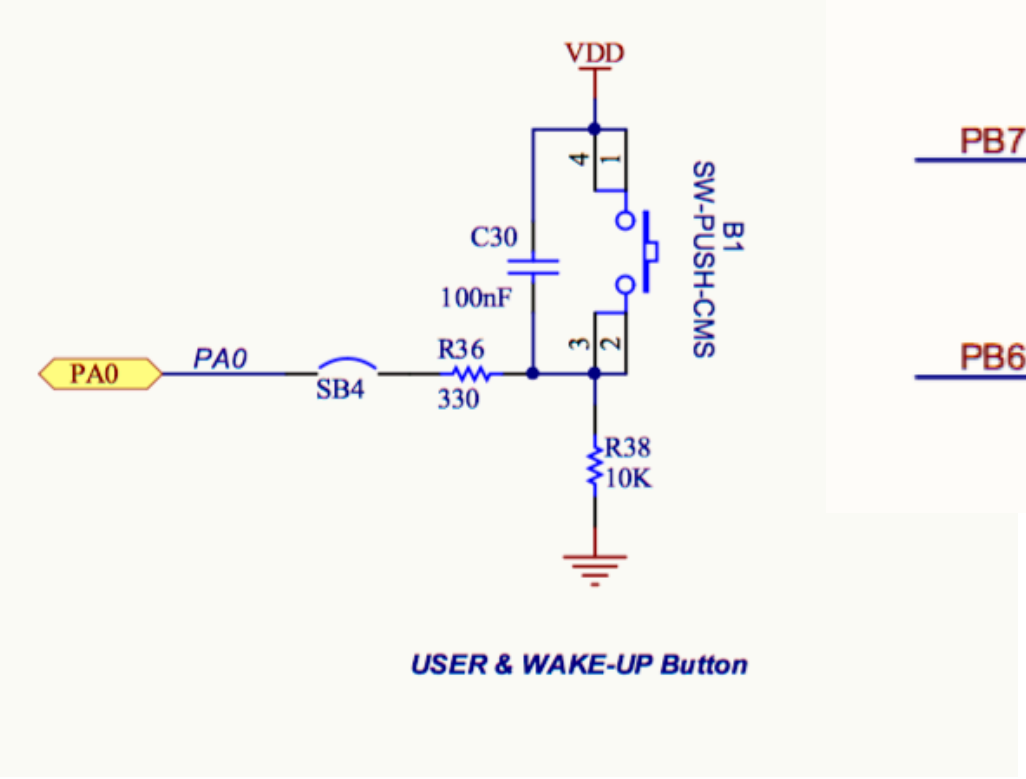
```
typedef enum
{
  GPIO_PIN_RESET = 0,
  GPIO_PIN_SET
}GPIO_PinState;
```

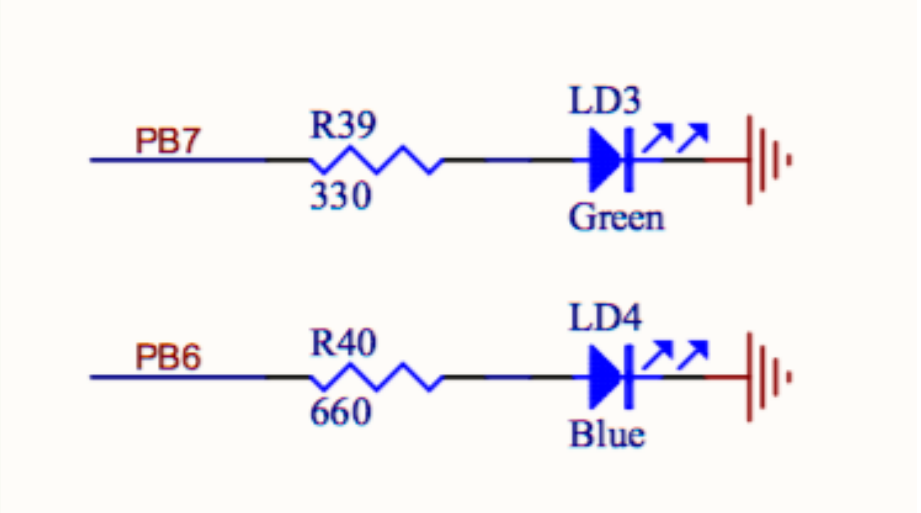
Delay Function

```
/**
  * @brief This function provides accurate delay (in milliseconds) based
  *        on variable incremented.
  * @note In the default implementation , SysTick timer is the source of time base.
  *       It is used to generate interrupts at regular time intervals where uwTick
  *       is incremented.
  * @note ThiS function is declared as __weak to be overwritten in case of other
  *       implementations in user file.
  * @param Delay: specifies the delay time length, in milliseconds.
  * @retval None
  */
__weak void HAL_Delay(__IO uint32_t Delay)
{
  uint32_t tickstart = 0;
  tickstart = HAL_GetTick();
  while((HAL_GetTick() - tickstart) < Delay)
  {
  }
}
```

QUIZ1

Press a button and both turn LED on otherwise turn them off





QUIZ2

Push button to toggle LED state