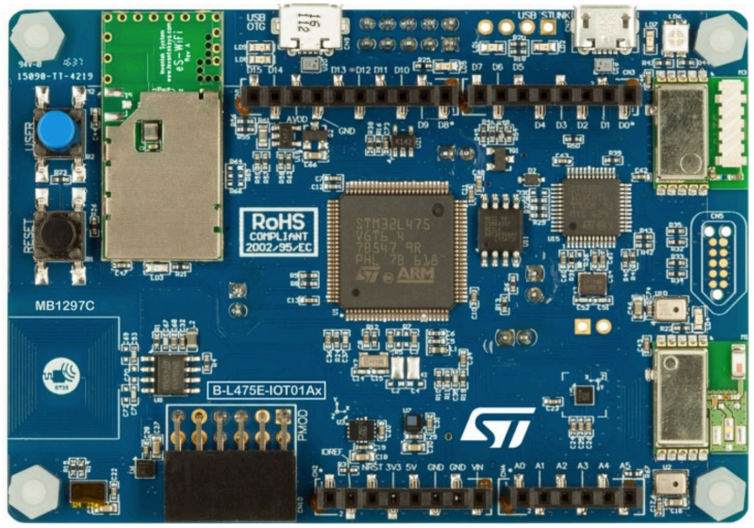
Embedded Systems Hardware Interfacing

GPIO

Norman McEntire



Contents

- Concepts
- Data Sheet - STM32L475
- User Manual - UM2153 - STM32L Discovery Kit for IoT
- Schematics - STM32L Discovery Kit for IoT
- API - STM32L HAL
    - Data Structures
    - Functions
- Hands-On Project

References

- https://en.wikipedia.org/wiki/General-purpose_input%2Foutput

- https://www.st.com/resource/en/schematic_pack/b-l475e-iot01ax_sch.zip

- https://www.st.com/resource/en/user_manual/dm00347848-discovery-kit-for-iot-node-multichannel-communication-with-stm32l4-stmicroelectronics.pdf

- https://www.st.com/resource/en/datasheet/stm32l475vg.pdf

GPIO Concepts

GPIO Concepts
General Purpose Pins

- GPIO - General Purpose Input/Output
- General purpose pins that are programmable at run-time
  - Option 1: On a microcontroller (e.g. STM32L486)
  - Option 2: On a port expander Integrated Circuit

GPIO Concepts
Programmable Options

- Many programmable options!
  - Unused - often the default
  - Input
    - With no pull-up or pull-pull-down resistor
    - With Pull-up Resistor
    - With Pull-down Resistor
- Output
  - Open Drain
  - Push/Pull

GPIO Concepts
Alternate Functions

- Some GPIO pins have “alternate functions”
    - Example: On STM32L4 a pin may be:
        - Option 1: GPIO
        - Option 2: Alternate Functions
            - Serial, Timer, Analog, etc
- NOTE: Alternate functions often help with PCB layout

GPIO Concepts
Additional Features

- Some GPIO pins have additional features
    - Inputs
        - Debouncing
        - Edge Detection (including interrupts)
        - Wakeup
    - Outputs
        - PWM - Pulse Width Modulation
        - High Current Outputs
    - Input/Output
        - Optical Isolators

GPIO Concepts
Interrupt Inputs

- GPIO Inputs often used as interrupt inputs
  - Level Triggered Interrupt
  - Edge Triggered Interrupt
    - Falling Edge
    - Rising Edge
- Wakeup Interrupt

Data Sheet

STM32l475

STM32L476 Data Sheet

STM32L475xx

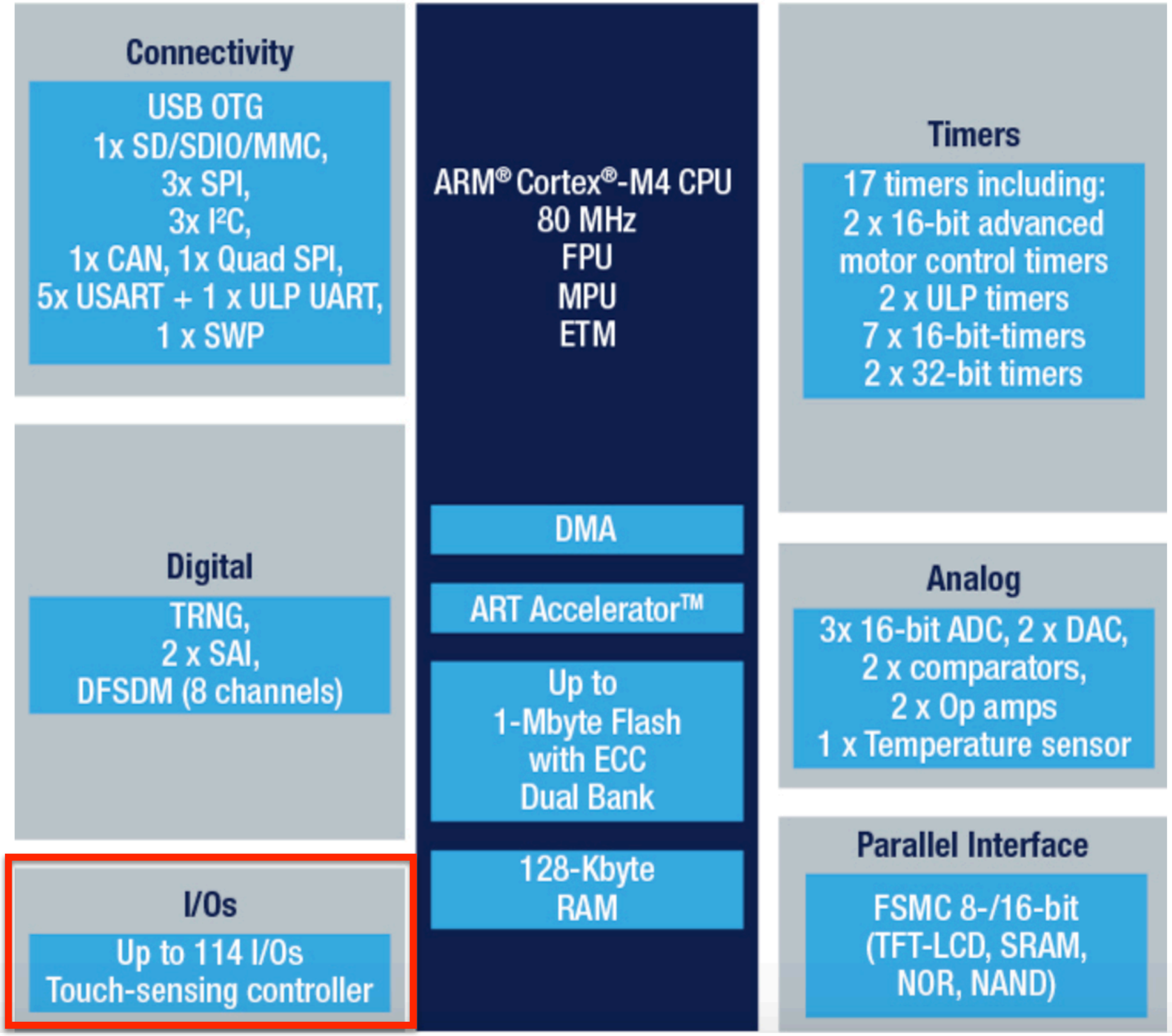
Ultra-low-power Arm® Cortex®-M4 32-bit MCU+FPU, 100DMIPS, up to 1MB Flash, 128 KB SRAM, USB OTG FS, analog, audio

Datasheet - production data

- Up to 114 fast I/Os, most 5 V-tolerant

NOTE: STM32 is a 3.3V device

STM32L475

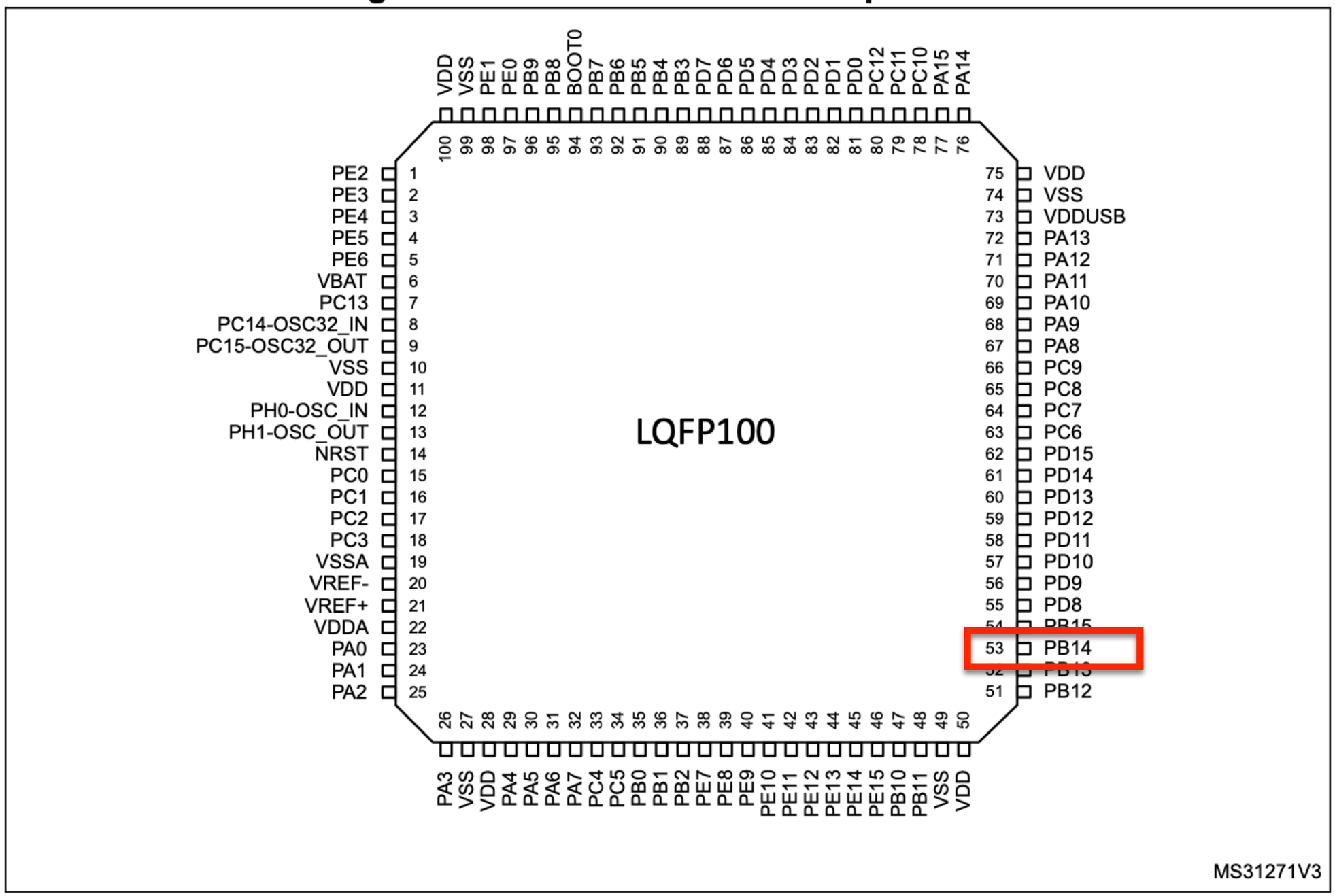


3.12 General-purpose inputs/outputs (GPIOs)

Each of the GPIO pins can be configured by software as output (push-pull or open-drain), as input (with or without pull-up or pull-down) or as peripheral alternate function. Most of the GPIO pins are shared with digital or analog alternate functions. Fast I/O toggling can be achieved thanks to their mapping on the AHB2 bus.

The I/Os alternate function configuration can be locked if needed following a specific sequence in order to avoid spurious writing to the I/Os registers.

Figure 6. STM32L475Vx LQFP100 pinout(1)



1. The above figure shows the package top view.

Table 16. STM32L475xx pin definitions (continued)

Pin Number		Pin name (function after reset)	Pin type	I/O structure	Notes	Pin functions	
LQFP64	LQFP100					Alternate functions	Additional functions
31	49	VSS	S	-	-	-	-
32	50	VDD	S	-	-	-	-
33	51	PB12	I/O	FT	-	TIM1_BKIN, TIM1_BKIN_COMP2, I2C2_SMBA, SPI2_NSS, DFSDM1_DATIN1, USART3_CK, LPUART1_RTS_DE, TSC_G1_IO1, SWPMI1_IO, SAI2_FS_A, TIM15_BKIN, EVENTOUT	-
34	52	PB13	I/O	FT_f	-	TIM1_CH1N, I2C2_SCL, SPI2_SCK, DFSDM1_CKIN1, USART3_CTS, LPUART1_CTS, TSC_G1_IO2, SWPMI1_TX, SAI2_SCK_A, TIM15_CH1N, EVENTOUT	-
35	53	PB14	I/O	FT_f	-	TIM1_CH2N, TIM8_CH2N, I2C2_SDA, SPI2_MISO, DFSDM1_DATIN2, USART3_RTS_DE, TSC_G1_IO3, SWPMI1_RX, SAI2_MCLK_A, TIM15_CH1, EVENTOUT	-

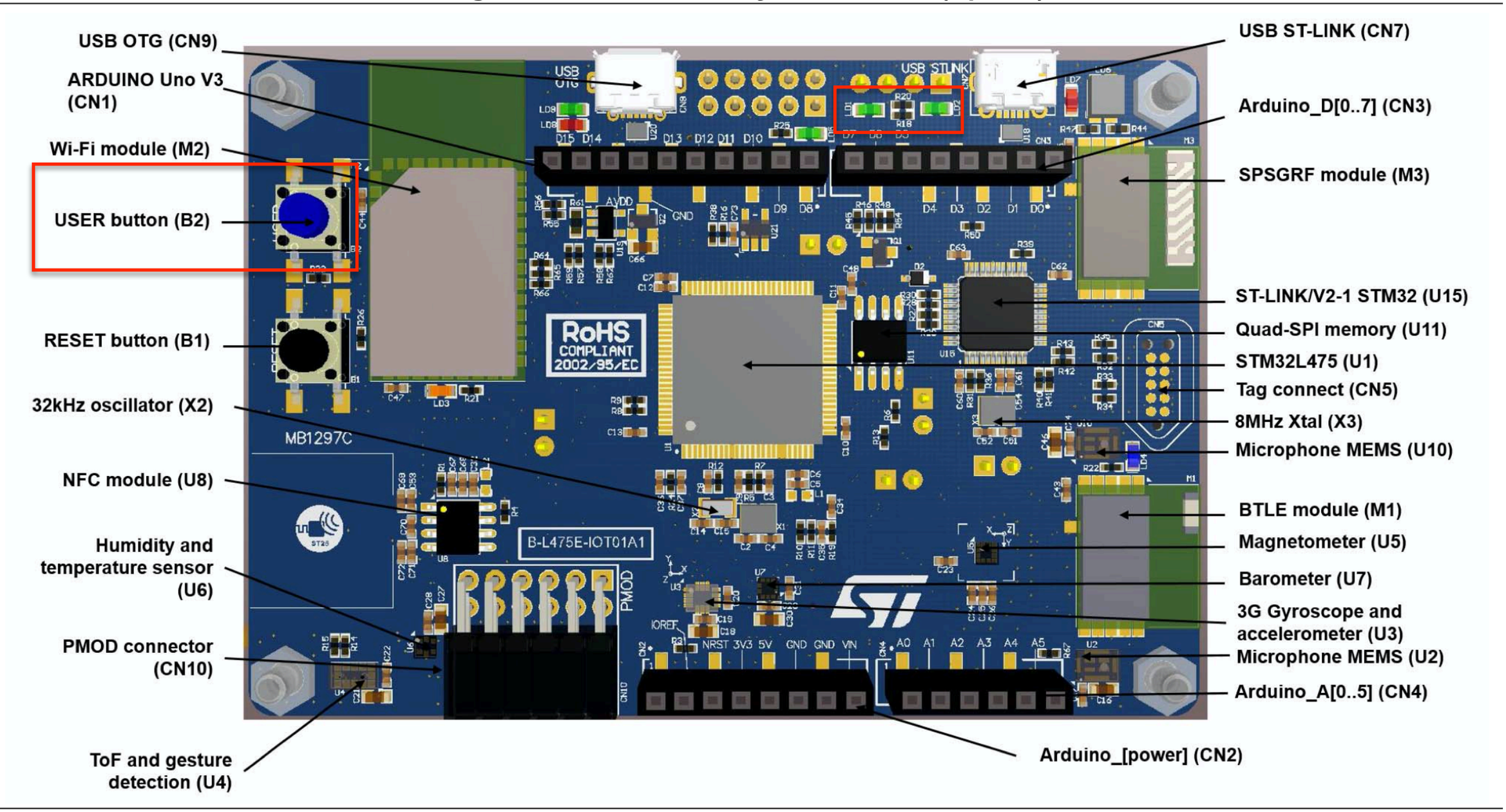
Table 17. Alternate function AF0 to AF7[(1)] (continued)

Port		AF0	AF1	AF2	AF3	AF4	AF5	AF6	AF7
		SYS_AF	TIM1/TIM2/ TIM5/TIM8/ LPTIM1	TIM1/TIM2/ TIM3/TIM4/ TIM5	TIM8	I2C1/I2C2/I2C3	SPI1/SPI2	SPI3/DFSDM	USART1/ USART2/ USART3
Port B	PB0	-	TIM1_CH2N	TIM3_CH3	TIM8_CH2N	-	-	-	USART3_CK
	PB1	-	TIM1_CH3N	TIM3_CH4	TIM8_CH3N	-	-	DFSDM1_ DATIN0	USART3_RTS_ DE
	PB2	RTC_OUT	LPTIM1_OUT	-	-	I2C3_SMBA	-	DFSDM1_CKIN0	-
	PB3	JTDO-TRACESWO	TIM2_CH2	-	-	-	SPI1_SCK	SPI3_SCK	USART1_RTS_ DE
	PB4	NJTRST	-	TIM3_CH1	-	-	SPI1_MISO	SPI3_MISO	USART1_CTS
	PB5	-	LPTIM1_IN1	TIM3_CH2	-	I2C1_SMBA	SPI1_MOSI	SPI3_MOSI	USART1_CK
	PB6	-	LPTIM1_ETR	TIM4_CH1	TIM8_BKIN2	I2C1_SCL	-	DFSDM1_ DATIN5	USART1_TX
	PB7	-	LPTIM1_IN2	TIM4_CH2	TIM8_BKIN	I2C1_SDA	-	DFSDM1_CKIN5	USART1_RX
	PB8	-	-	TIM4_CH3	-	I2C1_SCL	-	DFSDM1_ DATIN6	-
	PB9	-	IR_OUT	TIM4_CH4	-	I2C1_SDA	SPI2_NSS	DFSDM1_CKIN6	-
	PB10	-	TIM2_CH3	-	-	I2C2_SCL	SPI2_SCK	DFSDM1_ DATIN7	USART3_TX
	PB11	-	TIM2_CH4	-	-	I2C2_SDA	-	DFSDM1_CKIN7	USART3_RX
	PB12	-	TIM1_BKIN	-	TIM1_BKIN_ COMP2	I2C2_SMBA	SPI2_NSS	DFSDM1_ DATIN1	USART3_CK
	PB13	-	TIM1_CH1N	-	-	I2C2_SCL	SPI2_SCK	DFSDM1_CKIN1	USART3_CTS
	PB14	-	TIM1_CH2N	-	TIM8_CH2N	I2C2_SDA	SPI2_MISO	DFSDM1_ DATIN2	USART3_RTS_ DE

User Manual

STM32L Discovery Kit IoT Node

Use of GPIO
Board View - Buttons and LEDs



Use of GPIO Functional Description

7.14 Buttons and LEDs

The black button B1 located on top side is the reset of the microcontroller STM32L475VGT6. Refer to the *Figure 3: STM32L4 Discovery kit for IoT node (top view)*.

The blue button B1 located top side is available to be used as a digital input or as alternate wake-up function.

When the button is depressed the logic state is “0”, otherwise the logic state is “1”.

Two green LEDs (LD1 and LD2) located on the top side are available for the user. To light a LED a high logic state “1” should be written in the corresponding GPIO.

Table 2 gives the assignment of the control ports to the LED indicators.

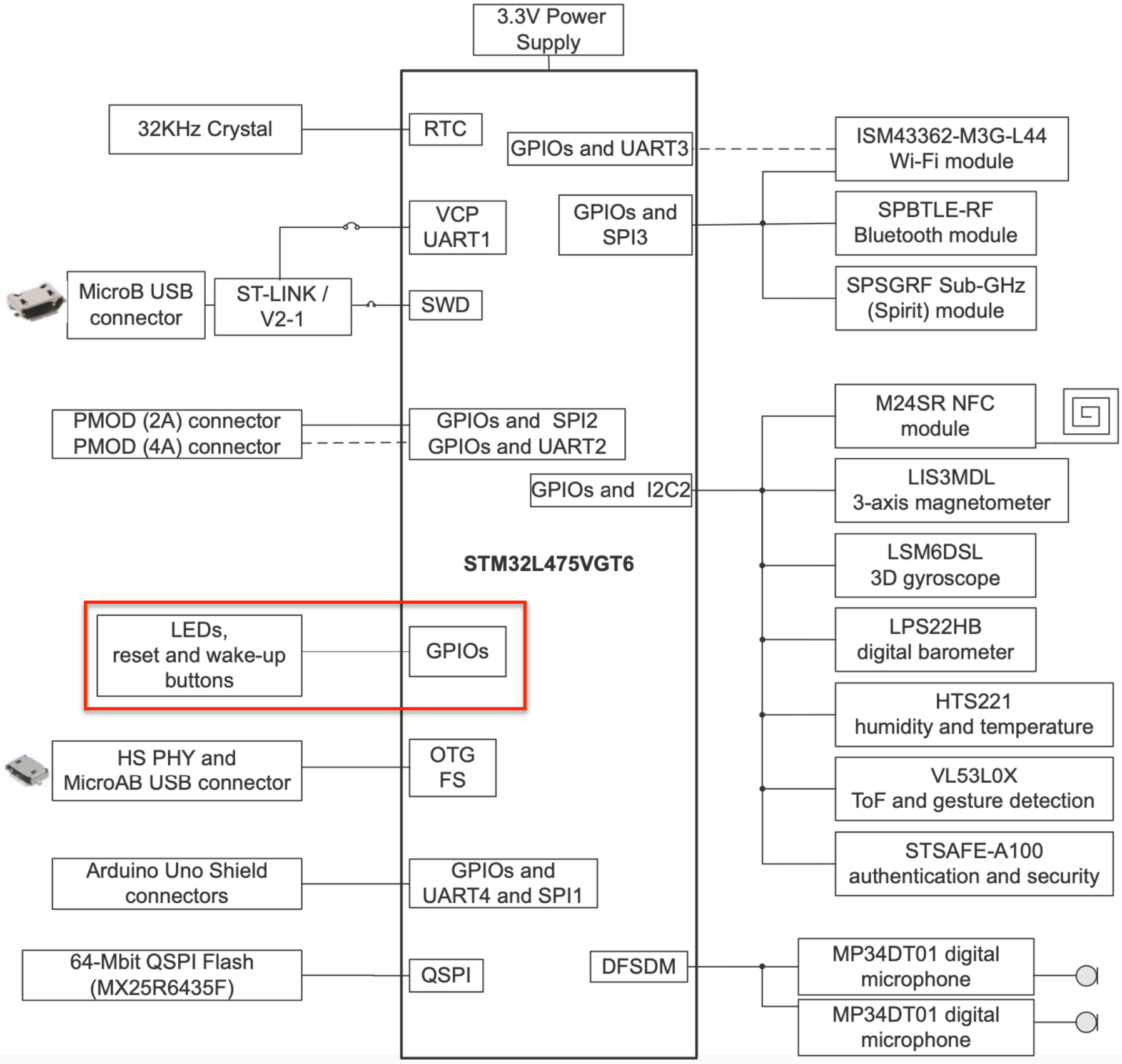
Use of GPIO
Pin Assignment

Table 2. Button and LED control port

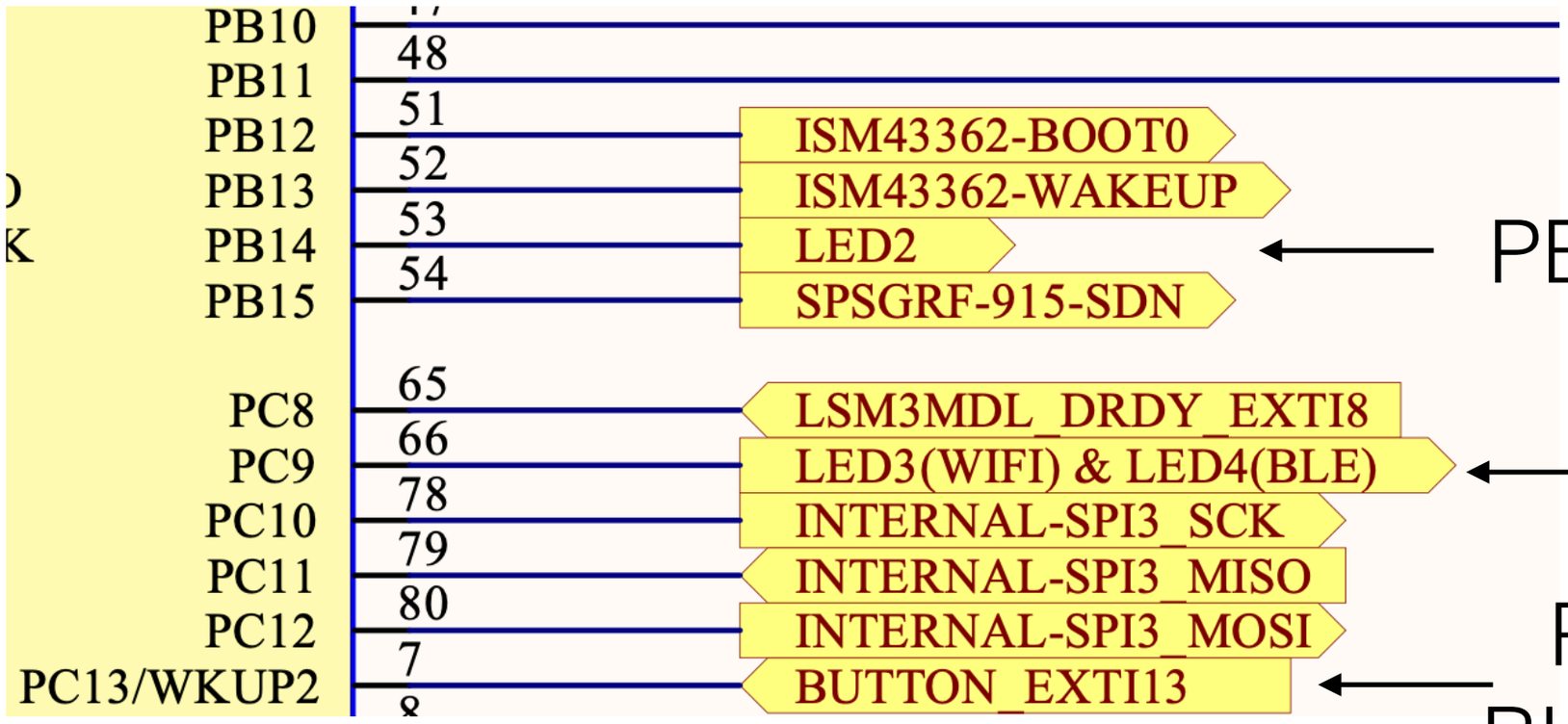
Reference	Color	Name	Comment
B1	black	Reset	-
B2	blue	Wake-up	Alternate function Wake-up
LD1	green	LED1	PA5 (alternate with ARD.D13)
LD2	green	LED2	PB14
LD3	yellow	LED3 (Wi-Fi)	PC9, Wi-Fi activity
LD4	blue	LED4 (BLE)	PC9, Bluetooth activity
LD5	green	5V Power	5 V available
LD6	Bicolor (red and green)	ST-LINK COM	green when communication
LD7	red	Fault Power	Current upper than 750 mA
LD8	red	V_{BUS} OCRCR	PE3
LD9	green	V_{BUS}OK	5 V USB available

Schematics

STM324L Discovery Kit IoT Node



STM32L Discovery Kit IoT Node

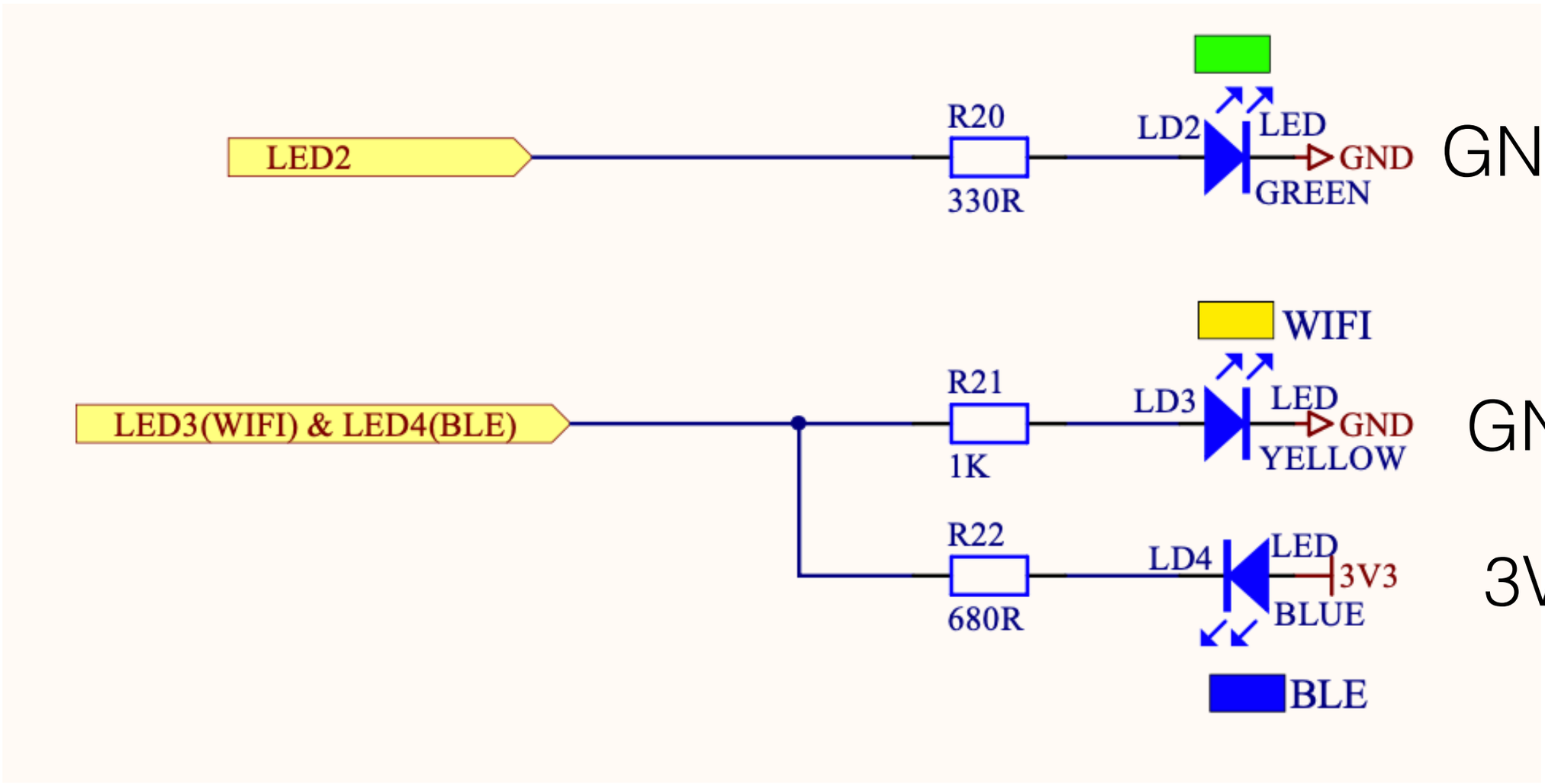


PB14 - LED2

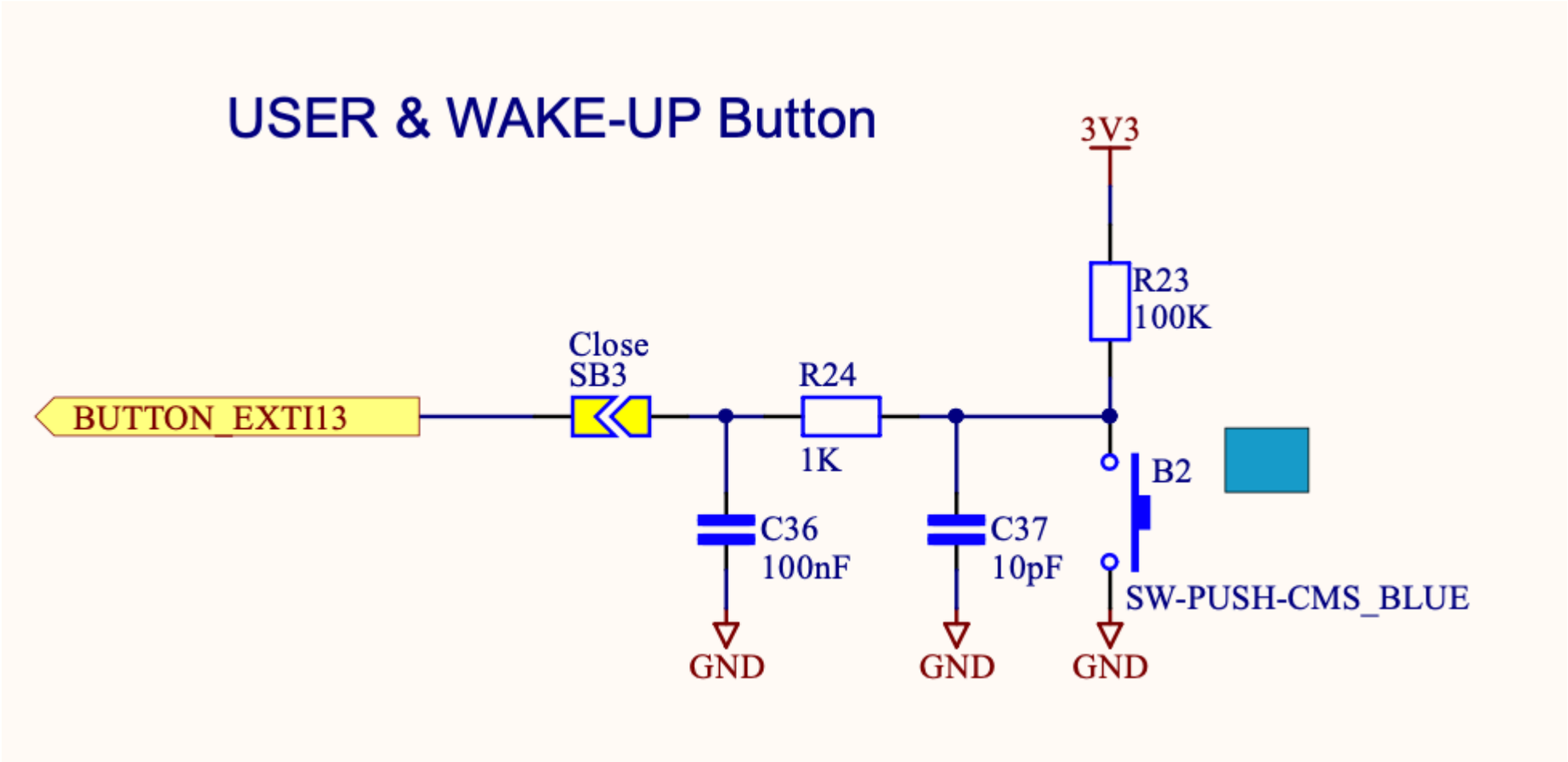
PC9 - LED3

PC13/WKUP2
BUTTON EXT113

STM32L Discovery Kit IoT Node - LED2 and LED3 Outputs



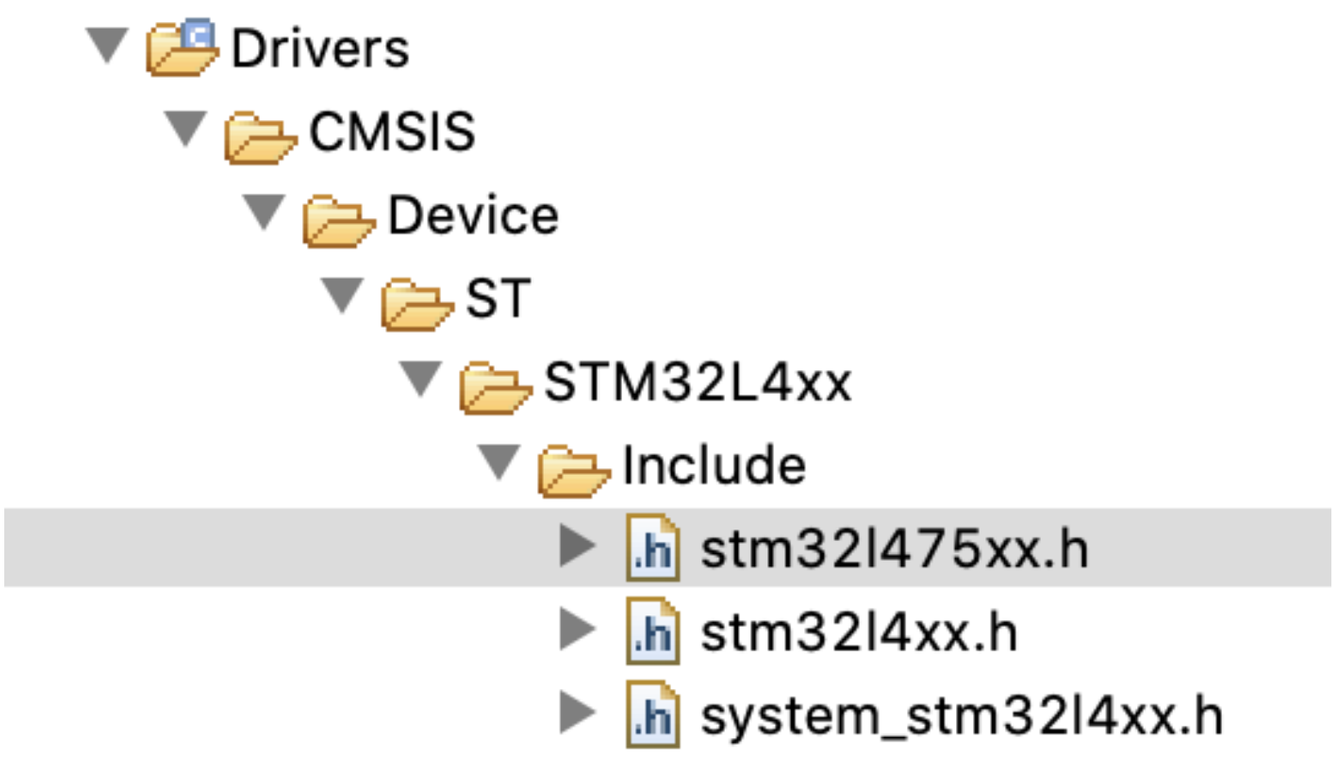
STM32L Discovery Kit IoT Node - Button Input



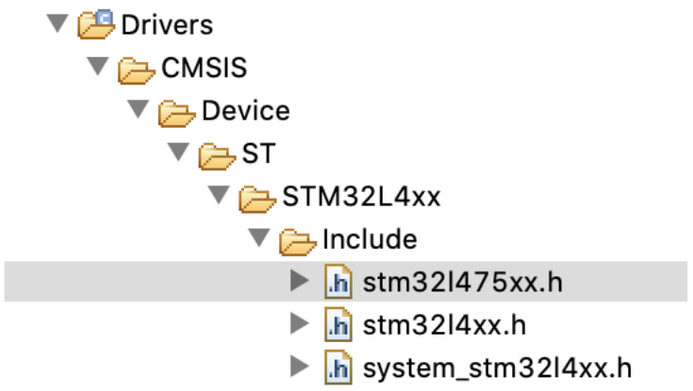
GPIO API

Data Structures

stm32l475xx.h

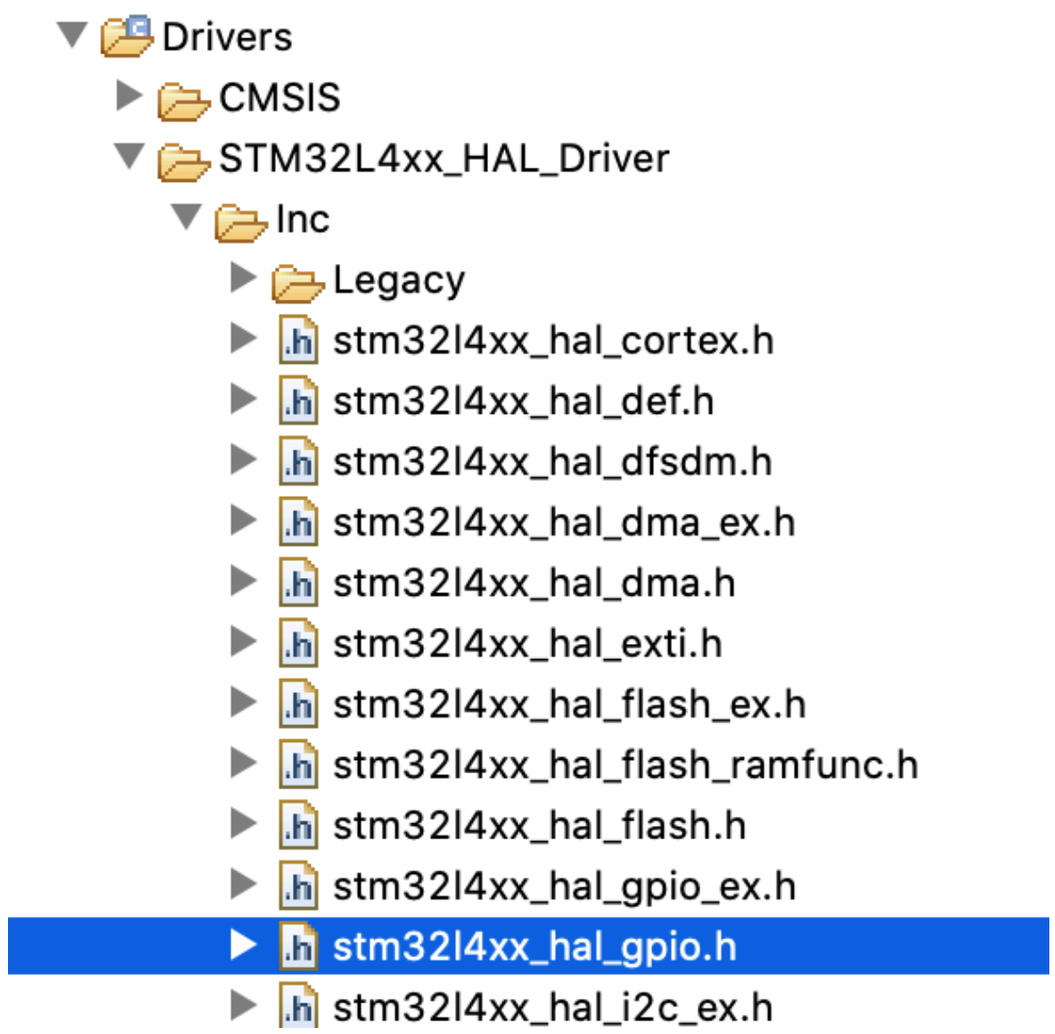


stm32l475xx.h
GPIO_TypeDef




```
526
527 /**
528   * @brief General Purpose I/O
529   */
530
531 typedef struct
532 {
533   __IO uint32_t MODER;       /*!< GPIO port mode register,               Address offset: 0x00      */
534   __IO uint32_t OTYPER;      /*!< GPIO port output type register,        Address offset: 0x04      */
535   __IO uint32_t OSPEEDR;     /*!< GPIO port output speed register,       Address offset: 0x08      */
536   __IO uint32_t PUPDR;       /*!< GPIO port pull-up/pull-down register,  Address offset: 0x0C      */
537   __IO uint32_t IDR;         /*!< GPIO port input data register,         Address offset: 0x10      */
538   __IO uint32_t ODR;         /*!< GPIO port output data register,        Address offset: 0x14      */
539   __IO uint32_t BSRR;        /*!< GPIO port bit set/reset  register,     Address offset: 0x18      */
540   __IO uint32_t LCKR;        /*!< GPIO port configuration lock register, Address offset: 0x1C      */
541   __IO uint32_t AFR[2];      /*!< GPIO alternate function registers,     Address offset: 0x20-0x24 */
542   __IO uint32_t BRR;         /*!< GPIO Bit Reset register,               Address offset: 0x28      */
543   __IO uint32_t ASCR;        /*!< GPIO analog switch control register,   Address offset: 0x2C     */
544
545 } GPIO_TypeDef;
546
```

stm32l4xx_hal_gpio.h



stm32l4xx_hal_gpio.h
GPIO_InitTypeDef

```
40    */
47  typedef struct
48  {
49    uint32_t Pin;        /*!< Specifies the GPIO pins to be configured.
50                              This parameter can be any value of @ref GPIO_pins */
51
52    uint32_t Mode;       /*!< Specifies the operating mode for the selected pins.
53                              This parameter can be a value of @ref GPIO_mode */
54
55    uint32_t Pull;       /*!< Specifies the Pull-up or Pull-Down activation for the selected pins.
56                              This parameter can be a value of @ref GPIO_pull */
57
58    uint32_t Speed;      /*!< Specifies the speed for the selected pins.
59                              This parameter can be a value of @ref GPIO_speed */
60
61    uint32_t Alternate;  /*!< Peripheral to be connected to the selected pins
62                               This parameter can be a value of @ref GPIOEx_Alternate_function_selection */
63  }GPIO_InitTypeDef;
64
```

stm32l4xx_hal_gpio.h GPIO_PinState

```
64
65 /**
66   * @brief  GPIO Bit SET and Bit RESET enumeration
67   */
68 typedef enum
69 {
70   GPIO_PIN_RESET = 0U,
71   GPIO_PIN_SET
72 }GPIO_PinState;
```

stm32l4xx_hal_gpio.h GPIO_PIN_...

```
78 /** @defgroup GPIO_Exported_Constants GPIO Exported Constants
79   * @{
80   */
81 /** @defgroup GPIO_pins GPIO pins
82   * @{
83   */
84 #define GPIO_PIN_0                 ((uint16_t)0x0001)  /* Pin 0 selected    */
85 #define GPIO_PIN_1                 ((uint16_t)0x0002)  /* Pin 1 selected    */
86 #define GPIO_PIN_2                 ((uint16_t)0x0004)  /* Pin 2 selected    */
87 #define GPIO_PIN_3                 ((uint16_t)0x0008)  /* Pin 3 selected    */
88 #define GPIO_PIN_4                 ((uint16_t)0x0010)  /* Pin 4 selected    */
89 #define GPIO_PIN_5                 ((uint16_t)0x0020)  /* Pin 5 selected    */
90 #define GPIO_PIN_6                 ((uint16_t)0x0040)  /* Pin 6 selected    */
91 #define GPIO_PIN_7                 ((uint16_t)0x0080)  /* Pin 7 selected    */
92 #define GPIO_PIN_8                 ((uint16_t)0x0100)  /* Pin 8 selected    */
93 #define GPIO_PIN_9                 ((uint16_t)0x0200)  /* Pin 9 selected    */
94 #define GPIO_PIN_10                ((uint16_t)0x0400)  /* Pin 10 selected   */
95 #define GPIO_PIN_11                ((uint16_t)0x0800)  /* Pin 11 selected   */
96 #define GPIO_PIN_12                ((uint16_t)0x1000)  /* Pin 12 selected   */
97 #define GPIO_PIN_13                ((uint16_t)0x2000)  /* Pin 13 selected   */
98 #define GPIO_PIN_14                ((uint16_t)0x4000)  /* Pin 14 selected   */
99 #define GPIO_PIN_15                ((uint16_t)0x8000)  /* Pin 15 selected   */
100 #define GPIO_PIN_All               ((uint16_t)0xFFFF)  /* All pins selected */
101
102 #define GPIO_PIN_MASK              (0x0000FFFFu) /* PIN mask for assert test */
```

stm32l4xx_hal_gpio.h GPIO_MODE_...

```
106
107⊖/** @defgroup GPIO_mode GPIO mode
108   * @brief GPIO Configuration Mode
109   *        Elements values convention: 0xX0yz00YZ
110   *           - X  : GPIO mode or EXTI Mode
111   *           - y  : External IT or Event trigger detection
112   *           - z  : IO configuration on External IT or Event
113   *           - Y  : Output type (Push Pull or Open Drain)
114   *           - Z  : IO Direction mode (Input, Output, Alternate or Analog)
115   * @{
116   */
117  #define  GPIO_MODE_INPUT                        (0x00000000u)   /*!< Input Floating Mode                   */
118  #define  GPIO_MODE_OUTPUT_PP                    (0x00000001u)   /*!< Output Push Pull Mode                 */
119  #define  GPIO_MODE_OUTPUT_OD                    (0x00000011u)   /*!< Output Open Drain Mode                */
120  #define  GPIO_MODE_AF_PP                        (0x00000002u)   /*!< Alternate Function Push Pull Mode     */
121  #define  GPIO_MODE_AF_OD                        (0x00000012u)   /*!< Alternate Function Open Drain Mode    */
122  #define  GPIO_MODE_ANALOG                       (0x00000003u)   /*!< Analog Mode  */
123  #define  GPIO_MODE_ANALOG_ADC_CONTROL           (0x0000000Bu)   /*!< Analog Mode for ADC conversion */
124  #define  GPIO_MODE_IT_RISING                    (0x10110000u)   /*!< External Interrupt Mode with Rising edge trigger detection
125  #define  GPIO_MODE_IT_FALLING                   (0x10210000u)   /*!< External Interrupt Mode with Falling edge trigger detection
126  #define  GPIO_MODE_IT_RISING_FALLING            (0x10310000u)   /*!< External Interrupt Mode with Rising/Falling edge trigger dete
127  #define  GPIO_MODE_EVT_RISING                   (0x10120000u)   /*!< External Event Mode with Rising edge trigger detection
128  #define  GPIO_MODE_EVT_FALLING                  (0x10220000u)   /*!< External Event Mode with Falling edge trigger detection
129  #define  GPIO_MODE_EVT_RISING_FALLING           (0x10320000u)   /*!< External Event Mode with Rising/Falling edge trigger detectio
130⊖/**
```

stm32l4xx_hal_gpio.h GPIO_SPEED_FREQ_...

```
134 /** @defgroup GPIO_speed GPIO speed
135   * @brief GPIO Output Maximum frequency
136   * @{
137   */
138 #define  GPIO_SPEED_FREQ_LOW         (0x00000000u)  /*!< range up to 5 MHz, please refer to the product datasheet */
139 #define  GPIO_SPEED_FREQ_MEDIUM      (0x00000001u)  /*!< range  5 MHz to 25 MHz, please refer to the product datasheet */
140 #define  GPIO_SPEED_FREQ_HIGH        (0x00000002u)  /*!< range 25 MHz to 50 MHz, please refer to the product datasheet */
141 #define  GPIO_SPEED_FREQ_VERY_HIGH   (0x00000003u)  /*!< range 50 MHz to 80 MHz, please refer to the product datasheet */
```

stm32l4xx_hal_gpio.h GPIO_NOPULL, GPIO_PULLUP, GPIO_PULLDOWN

```
146  /** @defgroup GPIO_pull GPIO pull
147    * @brief GPIO Pull-Up or Pull-Down Activation
148    * @{
149    */
150  #define  GPIO_NOPULL        (0x00000000u)   /*!< No Pull-up or Pull-down activation  */
151  #define  GPIO_PULLUP        (0x00000001u)   /*!< Pull-up activation                  */
152  #define  GPIO_PULLDOWN      (0x00000002u)   /*!< Pull-down activation                */
153  /**
```

stm32l4xx_hal_gpio.h
__HAL_GPIO_EXTI_..
EXTI = External Interrupt

```
165
166  /**
167    * @brief  Check whether the specified EXTI line flag is set or not.
168    * @param  __EXTI_LINE__ specifies the EXTI line flag to check.
169    *         This parameter can be GPIO_PIN_x where x can be(0..15)
170    * @retval The new state of __EXTI_LINE__ (SET or RESET).
171    */
172  #define __HAL_GPIO_EXTI_GET_FLAG(__EXTI_LINE__)       (EXTI->PR1 & (__EXTI_LINE__))
173
174  /**
175    * @brief  Clear the EXTI's line pending flags.
176    * @param  __EXTI_LINE__ specifies the EXTI lines flags to clear.
177    *         This parameter can be any combination of GPIO_PIN_x where x can be (0..15)
178    * @retval None
179    */
180  #define __HAL_GPIO_EXTI_CLEAR_FLAG(__EXTI_LINE__)     (EXTI->PR1 = (__EXTI_LINE__))
181
182  /**
183    * @brief  Check whether the specified EXTI line is asserted or not.
184    * @param  __EXTI_LINE__ specifies the EXTI line to check.
185    *          This parameter can be GPIO_PIN_x where x can be(0..15)
186    * @retval The new state of __EXTI_LINE__ (SET or RESET).
187    */
188  #define __HAL_GPIO_EXTI_GET_IT(__EXTI_LINE__)         (EXTI->PR1 & (__EXTI_LINE__))
189
190  /**
191    * @brief  Clear the EXTI's line pending bits.
192    * @param  __EXTI_LINE__ specifies the EXTI lines to clear.
193    *          This parameter can be any combination of GPIO_PIN_x where x can be (0..15)
194    * @retval None
195    */
196  #define __HAL_GPIO_EXTI_CLEAR_IT(__EXTI_LINE__)       (EXTI->PR1 = (__EXTI_LINE__))
197
```

stm32l4xx_hal_gpio.h
Generate Software Interrupt

```
197
198⊖/**
199   * @brief  Generate a Software interrupt on selected EXTI line.
200   * @param  __EXTI_LINE__ specifies the EXTI line to check.
201   *         This parameter can be GPIO_PIN_x where x can be(0..15)
202   * @retval None
203   */
204 #define __HAL_GPIO_EXTI_GENERATE_SWIT(__EXTI_LINE__)  (EXTI->SWIER1 |= (__EXTI_LINE__))
205
```

stm32l4xx_hal_gpio.h IS_GPIO_...

```
210  /* Private macros ------------------------------------------------------------*/
211 /** @addtogroup GPIO_Private_Macros GPIO Private Macros
212    * @{
213    */
214  #define IS_GPIO_PIN_ACTION(ACTION)  (((ACTION) == GPIO_PIN_RESET) || ((ACTION) == GPIO_PIN_SET))
215
216 #define IS_GPIO_PIN(__PIN__)        ((((uint32_t)(__PIN__) & GPIO_PIN_MASK) != 0x00U) &&\
217                                      (((uint32_t)(__PIN__) & ~GPIO_PIN_MASK) == 0x00U))
218
219 #define IS_GPIO_MODE(__MODE__)      (((__MODE__) == GPIO_MODE_INPUT)                  ||\
220                                      ((__MODE__) == GPIO_MODE_OUTPUT_PP)              ||\
221                                      ((__MODE__) == GPIO_MODE_OUTPUT_OD)              ||\
222                                      ((__MODE__) == GPIO_MODE_AF_PP)                  ||\
223                                      ((__MODE__) == GPIO_MODE_AF_OD)                  ||\
224                                      ((__MODE__) == GPIO_MODE_IT_RISING)              ||\
225                                      ((__MODE__) == GPIO_MODE_IT_FALLING)             ||\
226                                      ((__MODE__) == GPIO_MODE_IT_RISING_FALLING)      ||\
227                                      ((__MODE__) == GPIO_MODE_EVT_RISING)             ||\
228                                      ((__MODE__) == GPIO_MODE_EVT_FALLING)            ||\
229                                      ((__MODE__) == GPIO_MODE_EVT_RISING_FALLING)     ||\
230                                      ((__MODE__) == GPIO_MODE_ANALOG)                 ||\
231                                      ((__MODE__) == GPIO_MODE_ANALOG_ADC_CONTROL))
232
233 #define IS_GPIO_SPEED(__SPEED__)    (((__SPEED__) == GPIO_SPEED_FREQ_LOW)        ||\
234                                      ((__SPEED__) == GPIO_SPEED_FREQ_MEDIUM)     ||\
235                                      ((__SPEED__) == GPIO_SPEED_FREQ_HIGH)       ||\
236                                      ((__SPEED__) == GPIO_SPEED_FREQ_VERY_HIGH))
237
238 #define IS_GPIO_PULL(__PULL__)      (((__PULL__) == GPIO_NOPULL)   ||\
239                                      ((__PULL__) == GPIO_PULLUP)   || \
240                                      ((__PULL__) == GPIO_PULLDOWN))
241 /**
```

GPIO API

Functions

stm32l4xx_hal_gpio.h
HAL_GPIO_Init(), HAL_GPIO_DeInit()

```
252
253⊖/** @addtogroup GPIO_Exported_Functions_Group1 Initialization/de-initialization functions
254   *  @brief    Initialization and Configuration functions
255   * @{
256   */
257
258  /* Initialization and de-initialization functions ******************************/
259  void              HAL_GPIO_Init(GPIO_TypeDef  *GPIOx, GPIO_InitTypeDef *GPIO_Init);
260  void              HAL_GPIO_DeInit(GPIO_TypeDef  *GPIOx, uint32_t GPIO_Pin);
261
```

stm32l4xx_hal_gpio.h
HAL_GPIO_...

```
269
270 /* IO operation functions *****************************************************/
271 GPIO_PinState     HAL_GPIO_ReadPin(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin);
272 void              HAL_GPIO_WritePin(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin, GPIO_PinState PinState);
273 void              HAL_GPIO_TogglePin(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin);
274 HAL_StatusTypeDef HAL_GPIO_LockPin(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin);
275 void              HAL_GPIO_EXTI_IRQHandler(uint16_t GPIO_Pin);
276 void              HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin);
277
```

GPIO

Hands-On Project

GPIO Hands-On Project Overview

- The goal of this project is to give you hands-on experience with GPIO
  - Using STM32L4 Discovery Kit IoT Node
  - Using STM32Cube IDE
- To confirm your experience, you will create a **PDF document** that you will submit for grading
  - The PDF document will capture the major steps you perform to complete this project
  - See example PDF posted with assignment for example PDF format

GPIO Hands-On Project User Stories

- User Story 1 - GPIO Output to drive LED2
    - At power, blink LED2 three times - 1 second on, 1 second off, three times
- User Story 2 - Blink Wifi and BLE LEDs at 1 second rated “forever”
- User Story 3 - GPIO as Interrupt from Blue Button
    - User can press Button (Blue Button) to generate interrupt EXTI13 and to toggle LED2 on/off

User Story 1 Code

```
113     MX_USB_OTG_FS_PCD_Init();
114     /* USER CODE BEGIN 2 */
115
116     // Toggle LED2 three times at sstartup
117     HAL_GPIO_TogglePin(LED2_GPIO_Port, LED2_Pin);
118     HAL_Delay(1000); //1000 msec = 1 sec
119
120     HAL_GPIO_TogglePin(LED2_GPIO_Port, LED2_Pin);
121     HAL_Delay(1000); //1000 msec = 1 sec
122
123     HAL_GPIO_TogglePin(LED2_GPIO_Port, LED2_Pin);
124     HAL_Delay(1000); //1000 msec = 1 sec
125
126
127     /* USER CODE END 2 */
128
```

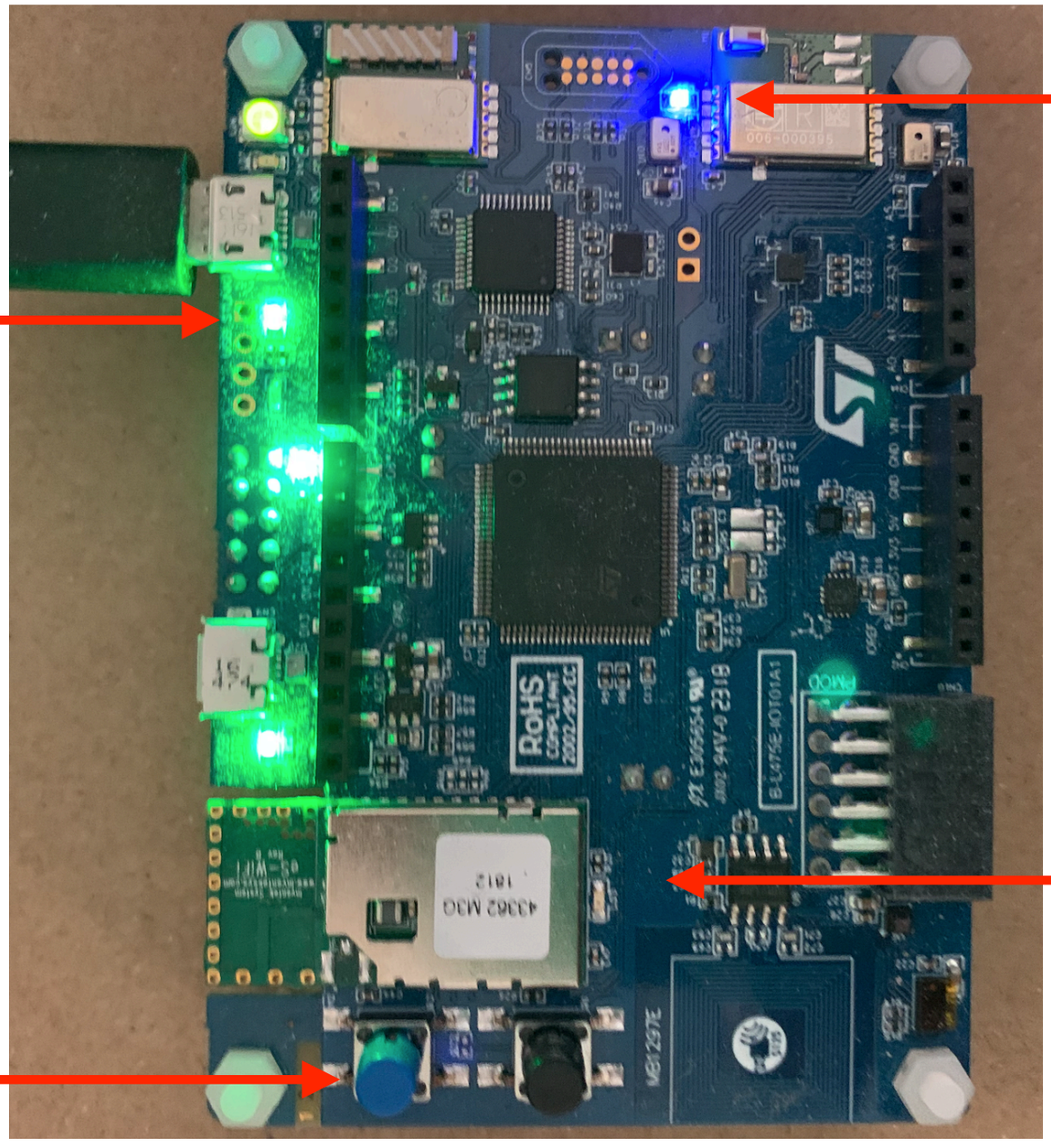
User Story 2 Code

```
128
129   /* Infinite loop */
130   /* USER CODE BEGIN WHILE */
131   while (1)
132   {
133     /* USER CODE END WHILE */
134
135     /* USER CODE BEGIN 3 */
136       HAL_GPIO_TogglePin(LED3_WIFI__LED4_BLE_GPIO_Port,LED3_WIFI__LED4_BLE_Pin);
137       HAL_Delay(1000); //500 msec = 1 sec
138   }
139   /* USER CODE END 3 */
140 }
141
```

User Story 3 Code

```
141
142 void HAL_GPIO_EXTI_Callback(uint16_t pin) {
143     if (pin == GPIO_PIN_13) {
144       HAL_GPIO_TogglePin(LED2_GPIO_Port, LED2_Pin);
145     }
146 }
147
```

Results



Summary

- Concepts
- Data Sheet - STM32L475
- User Manual - UM2153 - STM32L Discovery Kit for IoT
- Schematics - STM32L Discovery Kit for IoT
- API - STM32L HAL
    - Data Structures
    - Functions
- Hands-On Project