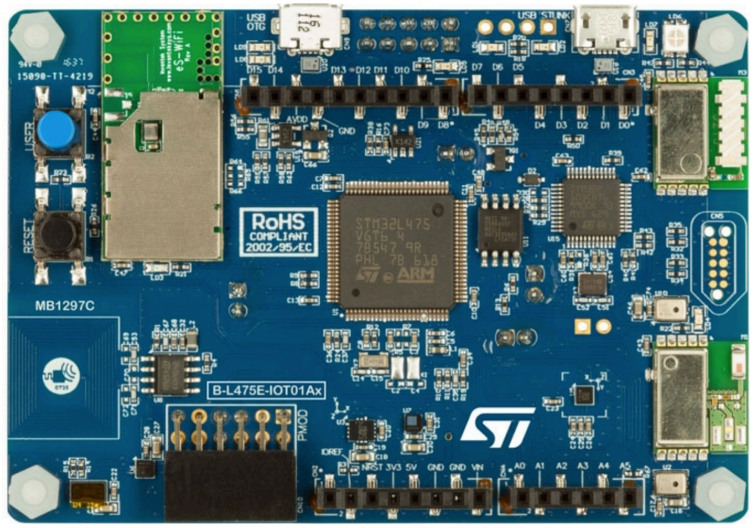
Embedded Systems Hardware Interfacing

USB

Norman McEntire



Contents

- Concepts
- Data Sheet - STM32L475
- User Manual - UM2153 - STM32L Discovery Kit for IoT
- Schematics - STM32L Discovery Kit for IoT
- API - STM32L LL (Low Level) No HAL
    - Data Structures
    - Functions
    - Middleware for USB
- Hands-On Project

References

- https://en.wikipedia.org/wiki/USB
- https://www.usb.org
- 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

USB

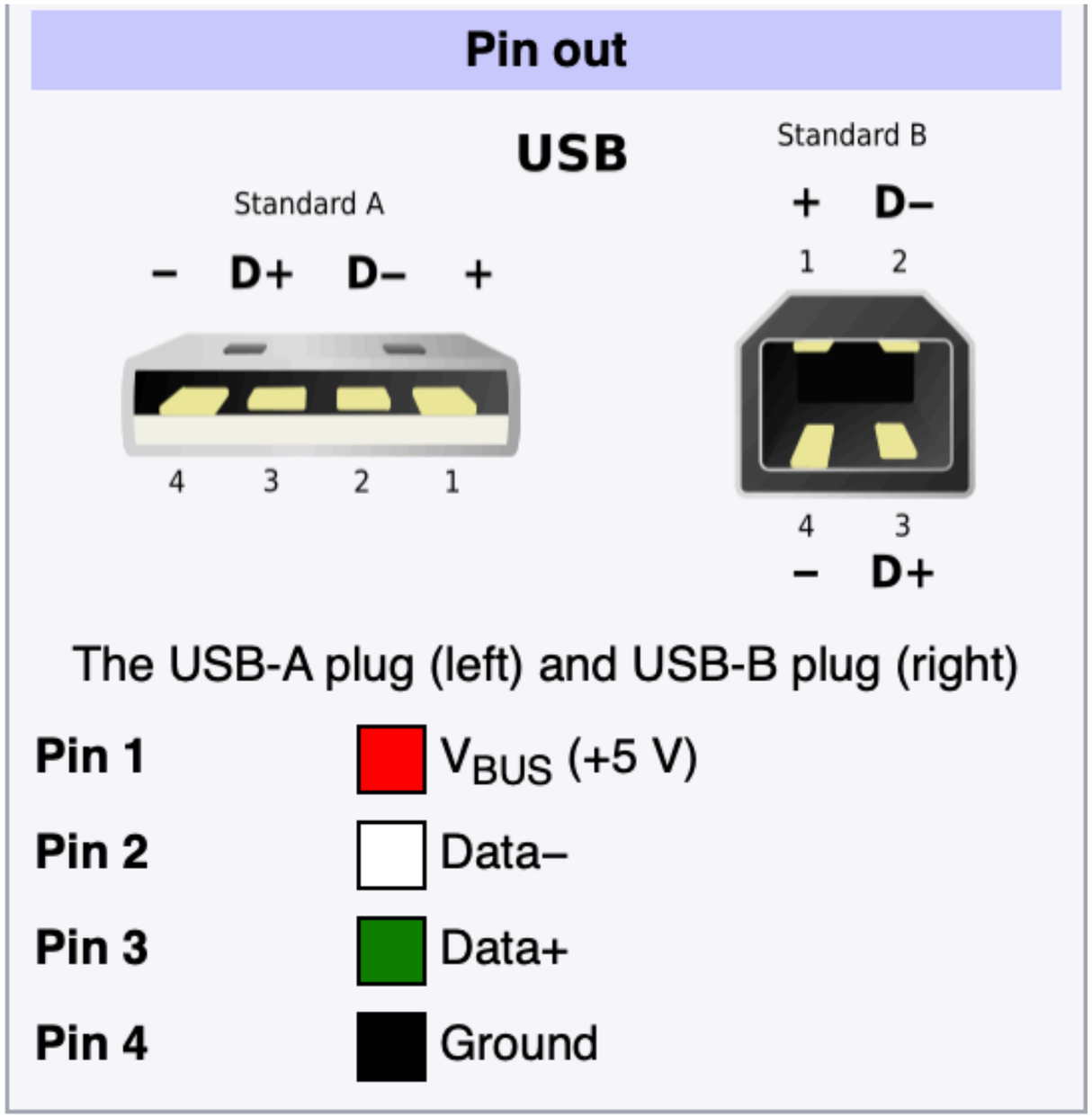
USB Concepts - Part 1

- Many SOCs include a USB Port
- USB - **U**niversal **S**erial **B**us
- Four Major Releases
  - USB 1.x
  - USB 2.0 <= Most widely supported
  - USB 3.x <= Most new devices use this
  - USB 4

USB Concepts - Part 2

- Three Major USB
  - USB **Host** (e.g. your PC)
  - USB **Device** (e.g. your Discovery Kit IoT Node)
  - USB **OTG** (e.g. your mobile device)
    - On The Go - can be host or device

USB Pinout

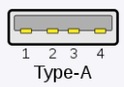
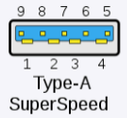
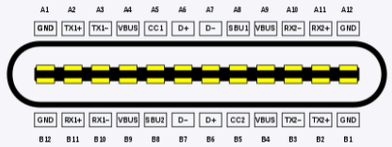
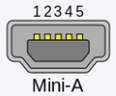
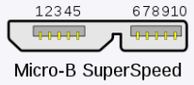


USB Data Rates

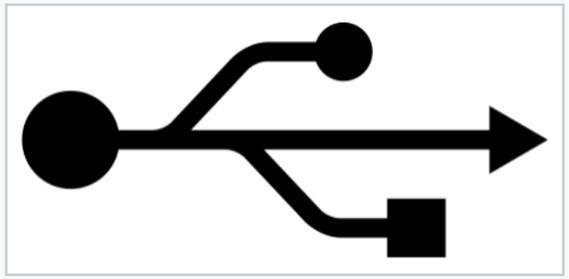
Connectors	USB 1.0 1996	USB 1.1 1998	USB 2.0 2001	USB 2.0 Revised	USB 3.0 2011	USB 3.1 2014	USB 3.2 2017	USB4 2019
Data rate	1.5 Mbit/s	1.5 Mbit/s (*Low Speed*) 12 Mbit/s (*Full Speed*)	1.5 Mbit/s (*Low Speed*) 12 Mbit/s (*Full Speed*) 480 Mbit/s (*High Speed*)		5 Gbit/s (*SuperSpeed*)	10 Gbit/s (*SuperSpeed+*)	20 Gbit/s (*SuperSpeed+*)	40 Gbit/s (*SuperSpeed+ and Thunderbolt 3*)

Low Speed: 1.5 Mbits/s
Full Speed: 12 Mbits/s
High Speed: 480 Mbits/s

SuperSpeed: 5 Gbits/s
SuperSpeed+: 10 Mbits/s
SuperSpeed+: 20 Mbits/s

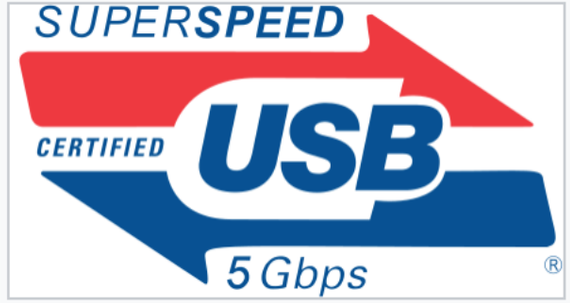
Connectors		USB 1.0 1996	USB 1.1 1998	USB 2.0 2001	USB 2.0 Revised	USB 3.0 2011	USB 3.1 2014	USB 3.2 2017	USB4 2019
Standard	A	Type A  Type-A			Type A  Type-A SuperSpeed			Deprecated	
	B	Type B  Type-B			Type B  Type-B SuperSpeed		Deprecated		
	C	N/A			Type C (*enlarged*) 				
Mini	A	N/A		Mini A  Mini-A			Deprecated		
	B			Mini B  Mini-B					
	AB	N/A		Mini AB  Mini-AB					
Micro	A	N/A							
	B	N/A		Micro B  Micro-B	Micro B  Micro-B SuperSpeed		Deprecated		
	AB			Micro AB 			Deprecated		

USB Logos



The basic USB *trident* logo[8]

The Hi-Speed USB logo

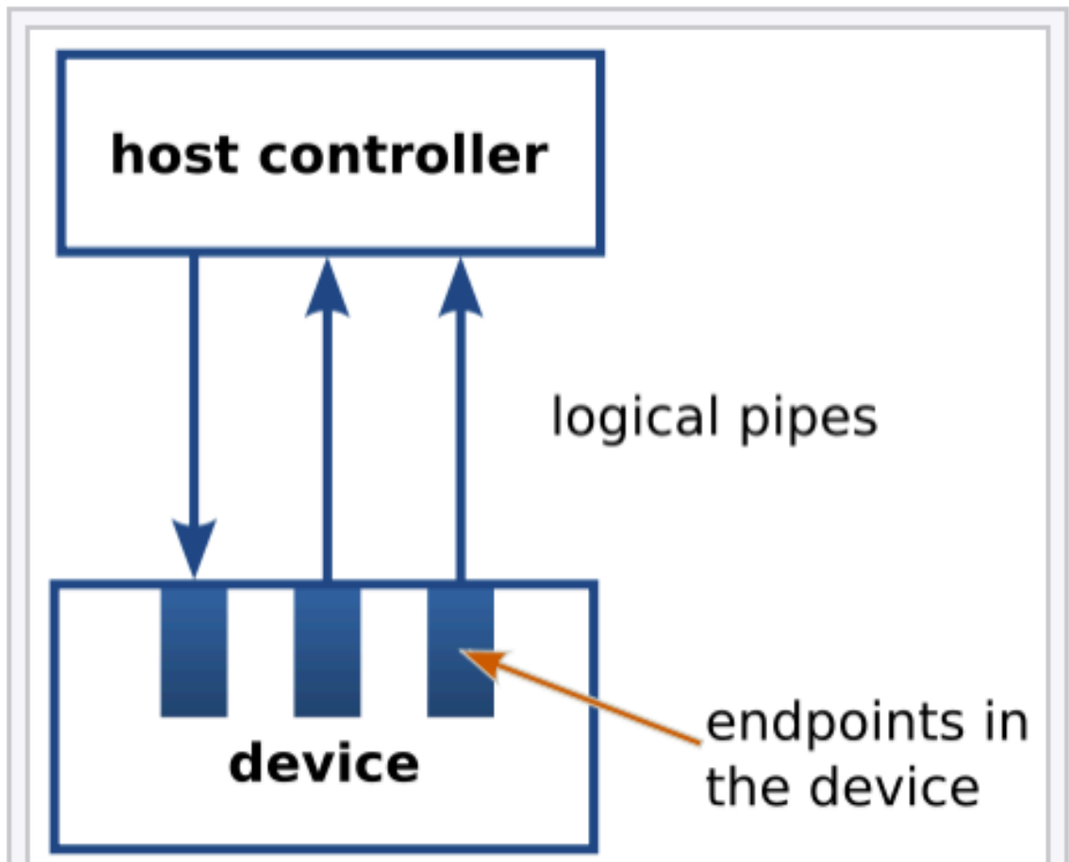


The SuperSpeed USB logo

The Certified USB4 40Gbps logo

Specification	Name	Previous name	USB-IF branding	Data rate	Transfer speed	Logo
USB 3.0	USB 3.2 Gen 1	USB 3.1 Gen 1	SuperSpeed USB 5Gbps	5 Gbit/s	500 MB/s	SS 5
USB 3.1	USB 3.2 Gen 2	USB 3.1 Gen 2	SuperSpeed USB 10Gbps	10 Gbit/s	1.21 GB/s	SS 10
USB 3.2	USB 3.2 Gen 2 × 2	N/A	SuperSpeed USB 20Gbps	20 Gbit/s	2.42 GB/s	SS 20

USB Host, USB device
Endpoints, Pipes



USB endpoints reside on the connected device: the channels to the host are referred to as pipes

Two Types of Pipes
Message and Stream

There are two types of pipe: stream and message.

- A *message* pipe is bi-directional and is used for *control* transfers. Message pipes are typically used for short, simple commands to the device, and for status responses from the device, used, for example, by the bus control pipe number 0.
- A *stream* pipe is a uni-directional pipe connected to a uni-directional endpoint that transfers data using an *isochronous*,[45] *interrupt*, or *bulk* transfer:

Isochronous transfers

At some guaranteed data rate (for fixed-bandwidth streaming data) but with possible data loss (e.g., realtime audio or video)

Interrupt transfers

Devices that need guaranteed quick responses (bounded latency) such as pointing devices, mice, and keyboards

Bulk transfers

Large sporadic transfers using all remaining available bandwidth, but with no guarantees on bandwidth or latency (e.g., file transfers)

USB Device Classes

Device classes [edit]

The functionality of a USB device is defined by a class code se
manufacturers.

Device classes include:[46]

Class	Usage	Description
00h	Device	Unspecified[47]
01h	Interface	Audio
02h	Both	Communications and CDC Control
03h	Interface	Human interface device (HID)
05h	Interface	Physical Interface Device (PID)
06h	Interface	Image (PTP/MTP)
07h	Interface	Printer
08h	Interface	Mass storage (MSC or UMS)

USB Power Standards

USB power standards

Specification	Current	Voltage	Power (max.)
Low-power device	100 mA	5 V[a]	0.50 W
Low-power SuperSpeed (USB 3.0) device	150 mA	5 V[a]	0.75 W
High-power device	500 mA[b]	5 V	2.5 W
High-power SuperSpeed (USB 3.0) device	900 mA[c]	5 V	4.5 W
Multi-lane SuperSpeed (USB 3.2 Gen 2) device	1.5 A[d]	5 V	7.5 W
Battery Charging (BC) 1.1	1.5 A	5 V	7.5 W
Battery Charging (BC) 1.2	5 A	5 V	25 W
USB-C	1.5 A	5 V	7.5 W
	3 A	5 V	15 W
Power Delivery 1.0 Micro-USB	3 A	20 V	60 W
Power Delivery 1.0 Type-A/B	5 A	20 V	100 W
Power Delivery 2.0/3.0 Type-C	5 A[e]	20 V	100 W

USB - Many Levels of Details

- Physical Layer
  - Signals, Voltages
- Protocol Layer
  - Packets
- Transaction Layer
  - SETUP Transaction
  - IN Transaction
  - OUT Transaction

USB Example
https://www.adafruit.com/product/954

USB to TTL Serial Cable - Debug / Console Cable for Raspberry Pi

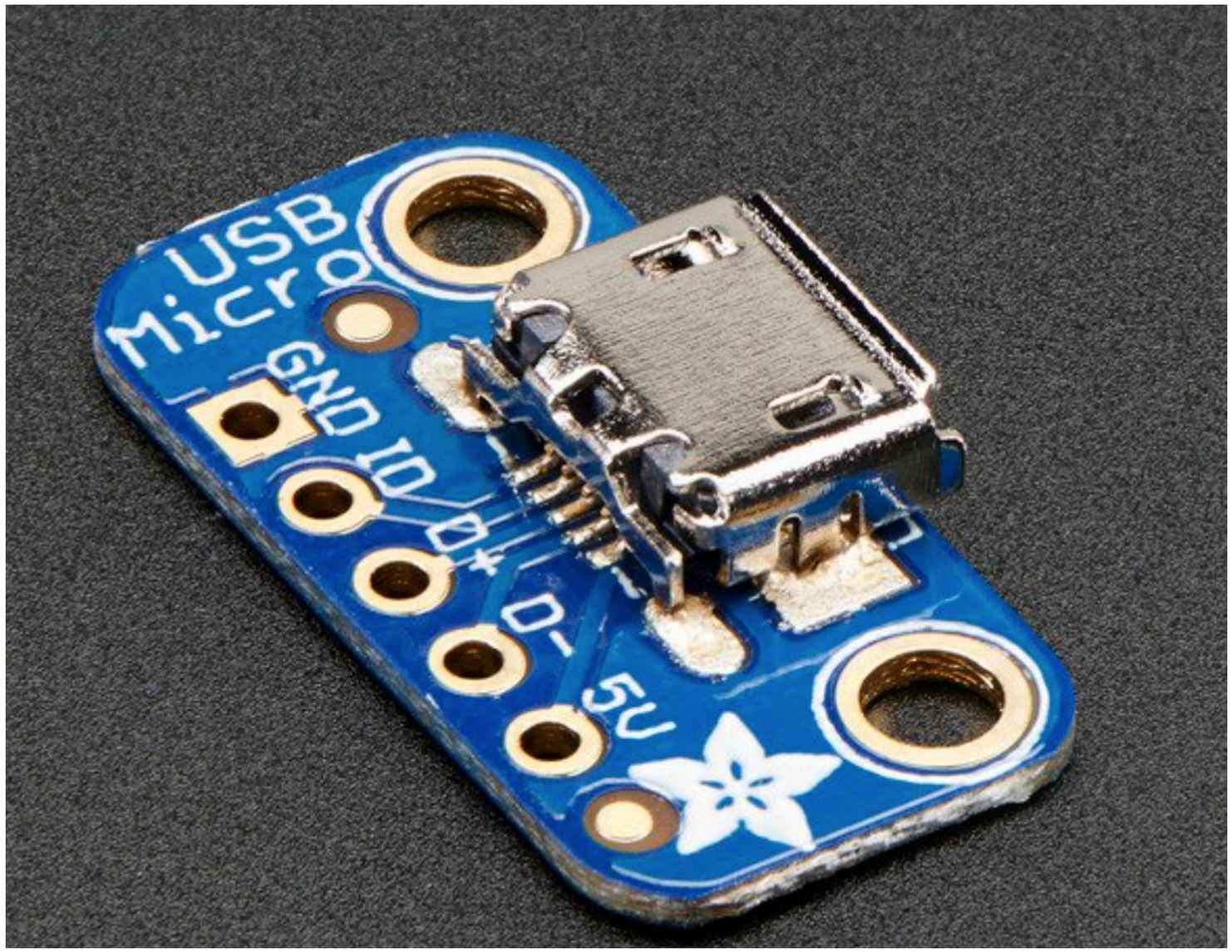
PRODUCT ID: 954

The cable is easiest way ever to connect to your microcontroller/Raspberry Pi/WiFi router serial console port. Inside the big USB plug is a USB<->Serial conversion chip and at the end of the 36" cable are four wire - red power, black ground, white RX into USB port, and green TX out of...

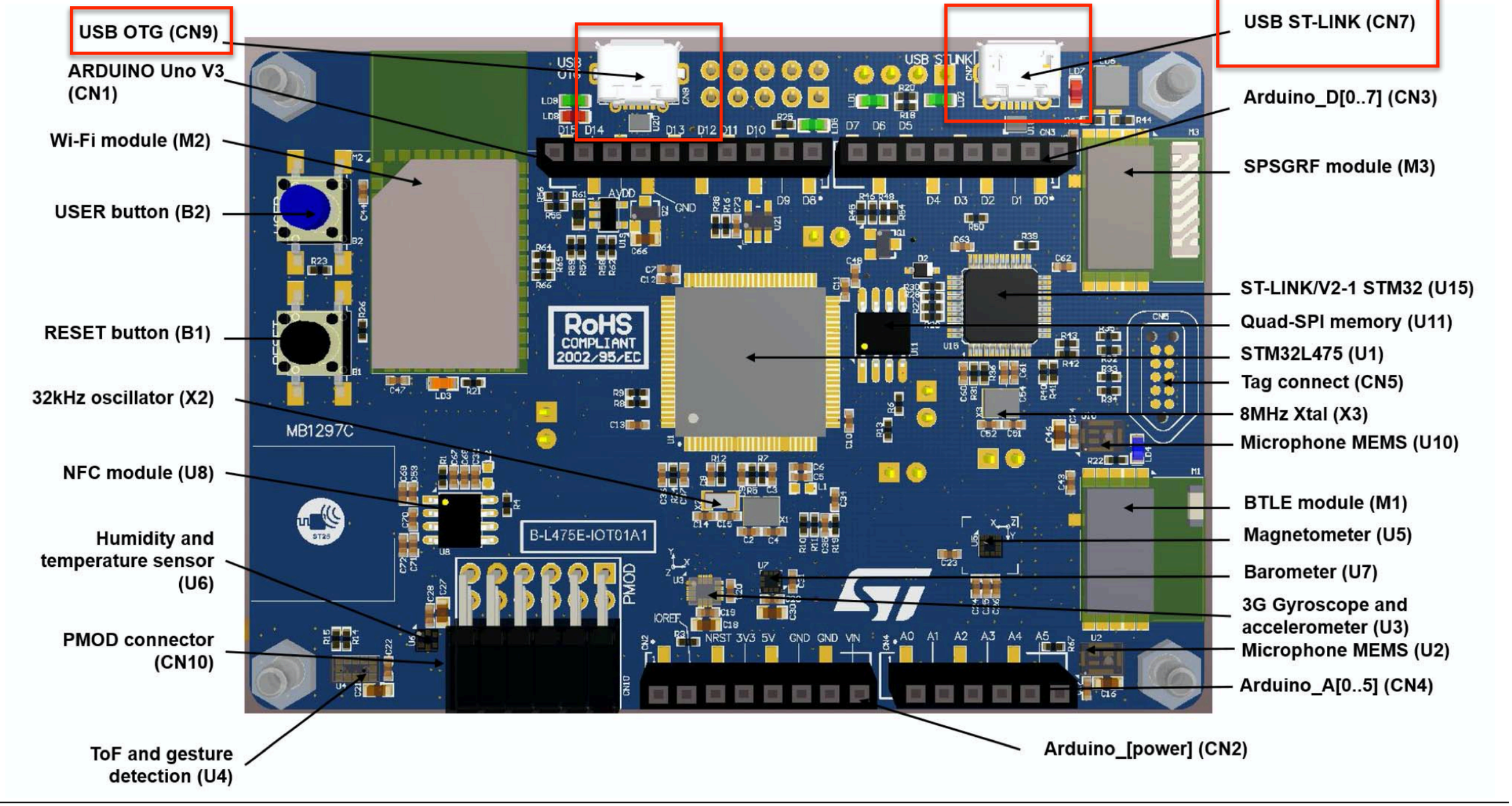
$9.95

USB Example
https://www.adafruit.com/product/1833

GND
ID
D+
D-
5V



USB Example Discovery Kit



Data Sheet

STM32l475

STM32L475 Data Sheet

STM32L475xx

life.augmented

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

- 20x communication interfaces
  - USB OTG 2.0 full-speed, LPM and BCD
  - 2x SAIs (serial audio interface)
  - 3x I2C FM+(1 Mbit/s), SMBus/PMBus
  - 5x USARTs (ISO 7816, LIN, IrDA, modem)
  - 1x LPUART (Stop 2 wake-up)
  - 3x SPIs (and 1x Quad SPI)
  - CAN (2.0B Active) and SDMMC interface
  - SWPMI single wire protocol master I/F
  - IRTIM (Infrared interface)

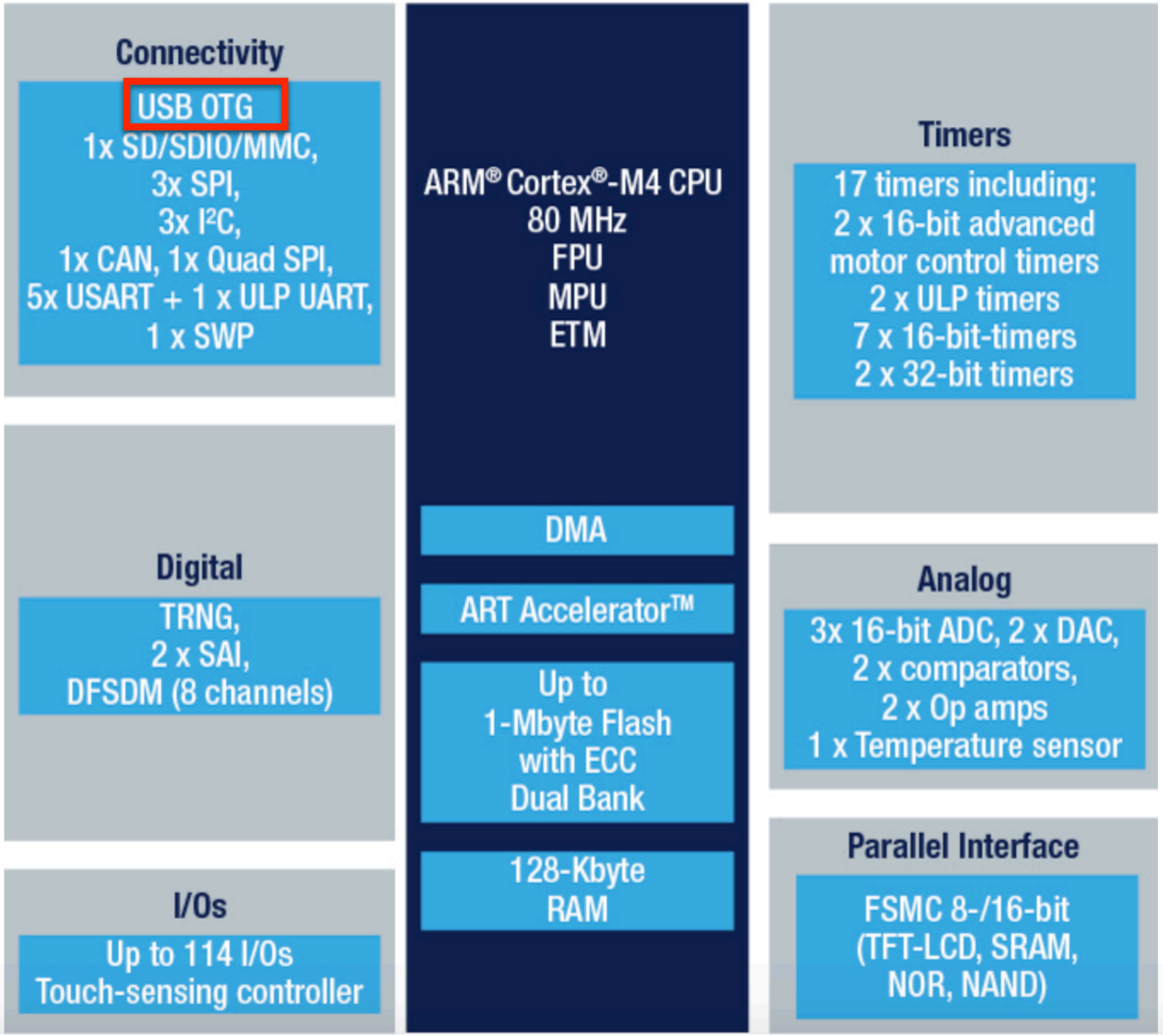
OTG = On The Go

LPM = Link Power Management

USART - Universal Serial/Async Receiver/Transmitter

LPUART - Low PowerUniversal Async Receiver/Transmitter

STM32L475



STM32L475 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

3.33 Universal serial bus on-the-go full-speed (OTG_FS)

The devices embed an USB OTG full-speed device/host/OTG peripheral with integrated transceivers. The USB OTG FS peripheral is compliant with the USB 2.0 specification and with the OTG 2.0 specification. It has software-configurable endpoint setting and supports suspend/resume. The USB OTG controller requires a dedicated 48 MHz clock that can be provided by the internal multispeed oscillator (MSI) automatically trimmed by 32.768 kHz external oscillator (LSE).This allows to use the USB device without external high speed crystal (HSE).

The major features are:

- Combined Rx and Tx FIFO size of 1.25 KB with dynamic FIFO sizing
- Supports the session request protocol (SRP) and host negotiation protocol (HNP)
- 1 bidirectional control endpoint + 5 IN endpoints + 5 OUT endpoints
- 12 host channels with periodic OUT support
- HNP/SNP/IP inside (no need for any external resistor)
- USB 2.0 LPM (Link Power Management) support
- Battery Charging Specification Revision 1.2 support
- Internal FS OTG PHY support

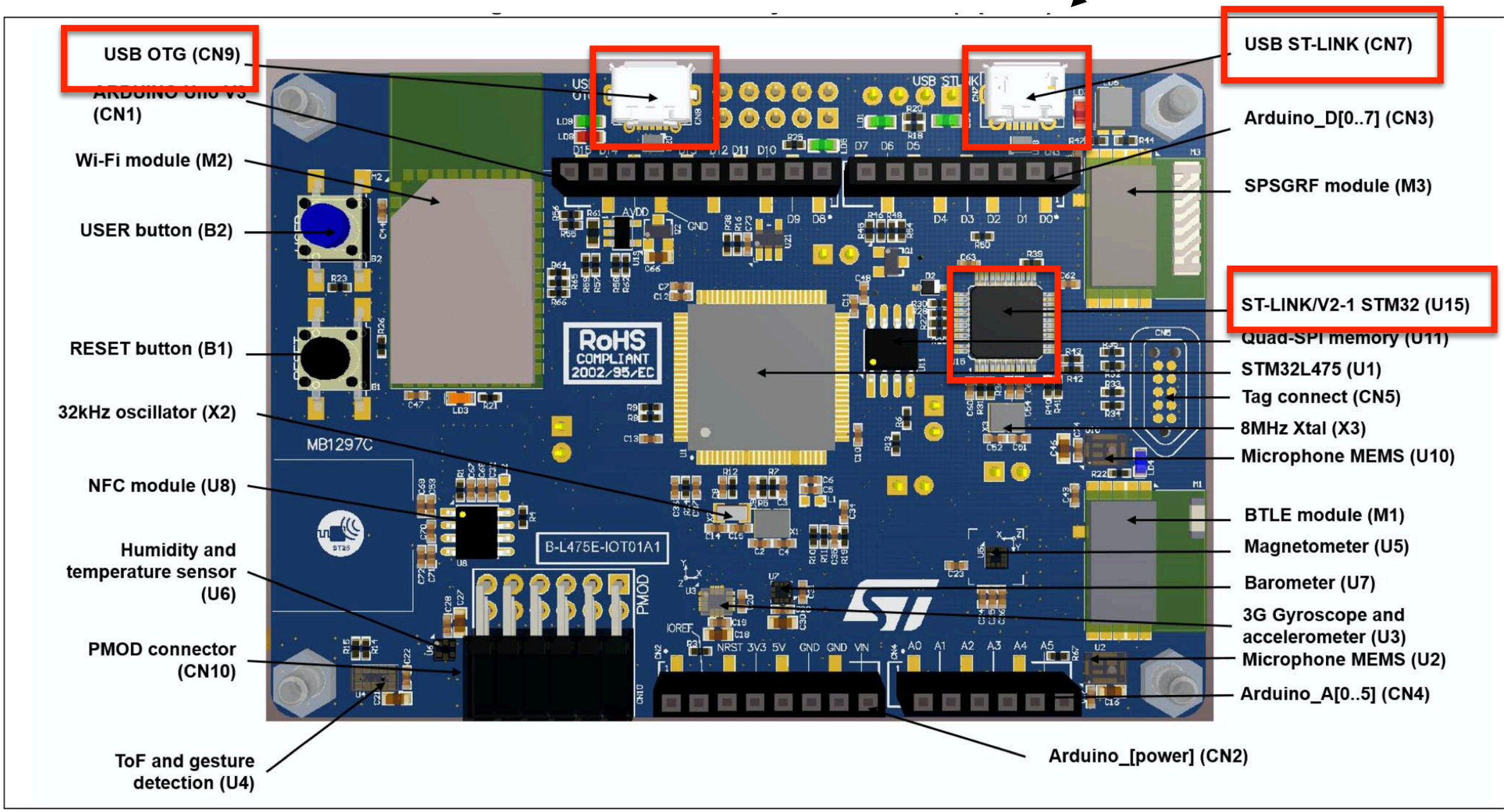
For OTG/Host modes, a power switch is needed in case bus-powered devices are connected.

User Manual

STM32L Discovery Kit IoT Node USB

Use of USB

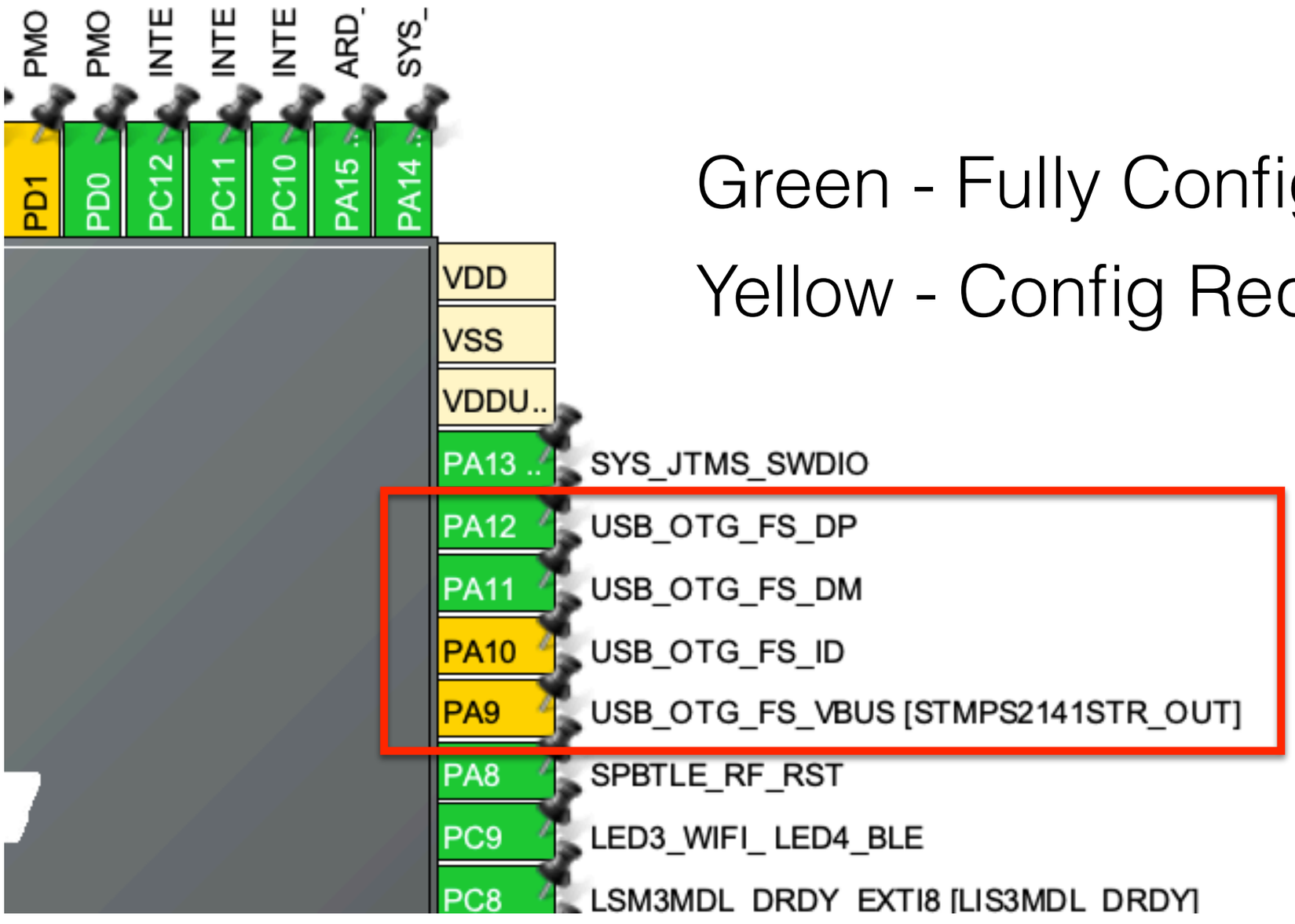
Virtual Console
USB Storage



USB OTG FS

OTG = On The Go

FS = Full Speed

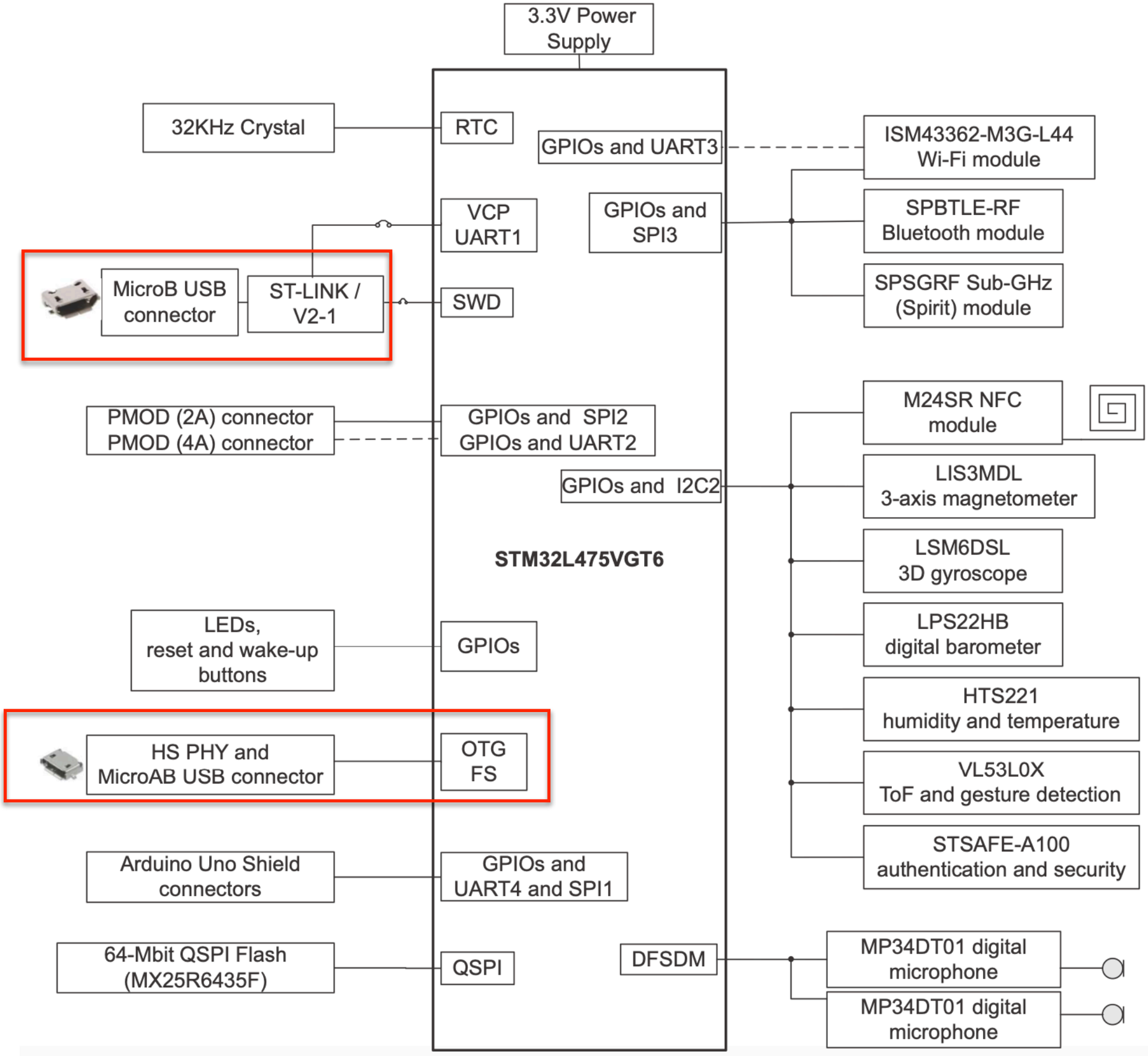


Green - Fully Configured

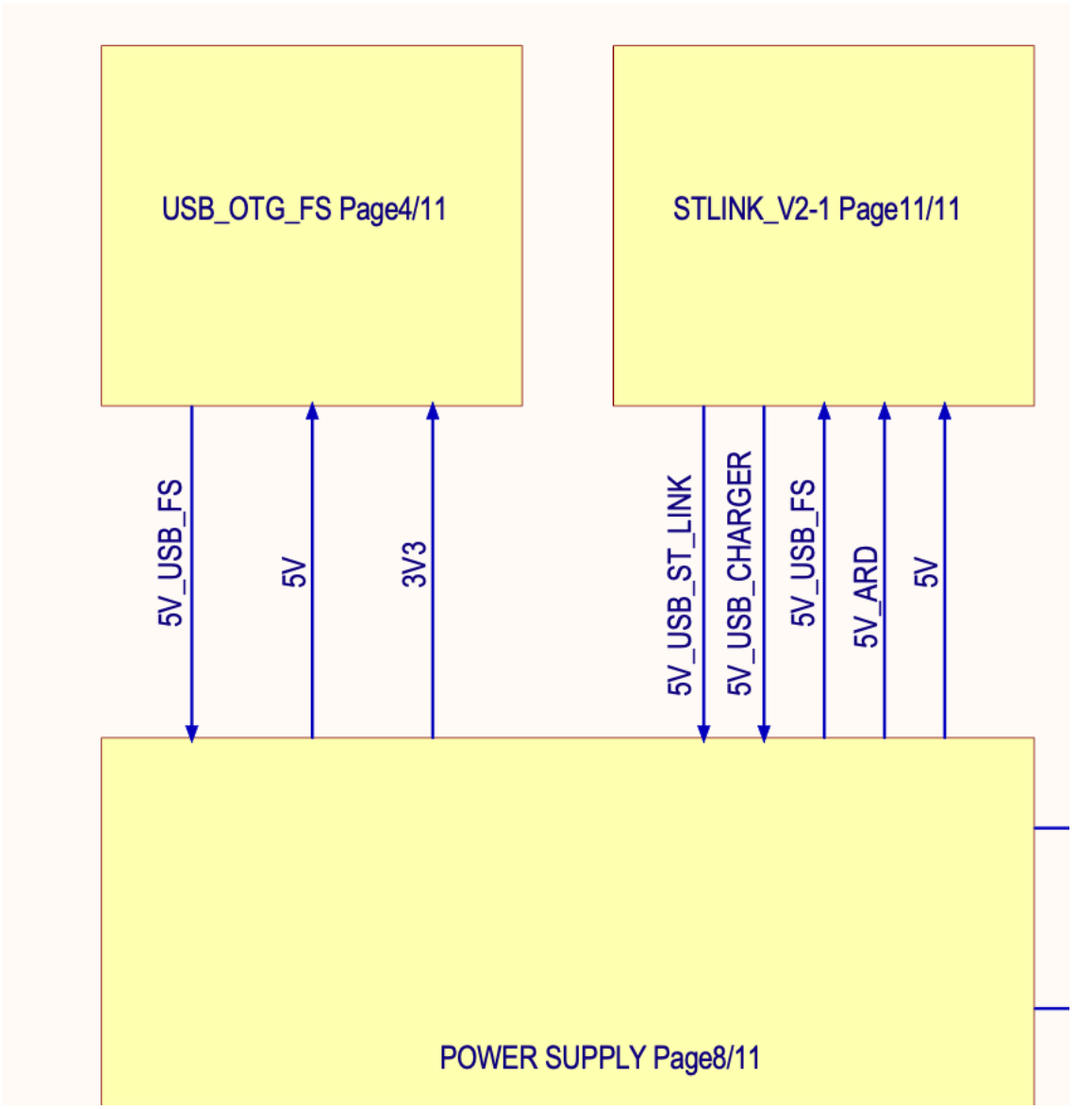
Yellow - Config Required

Schematics

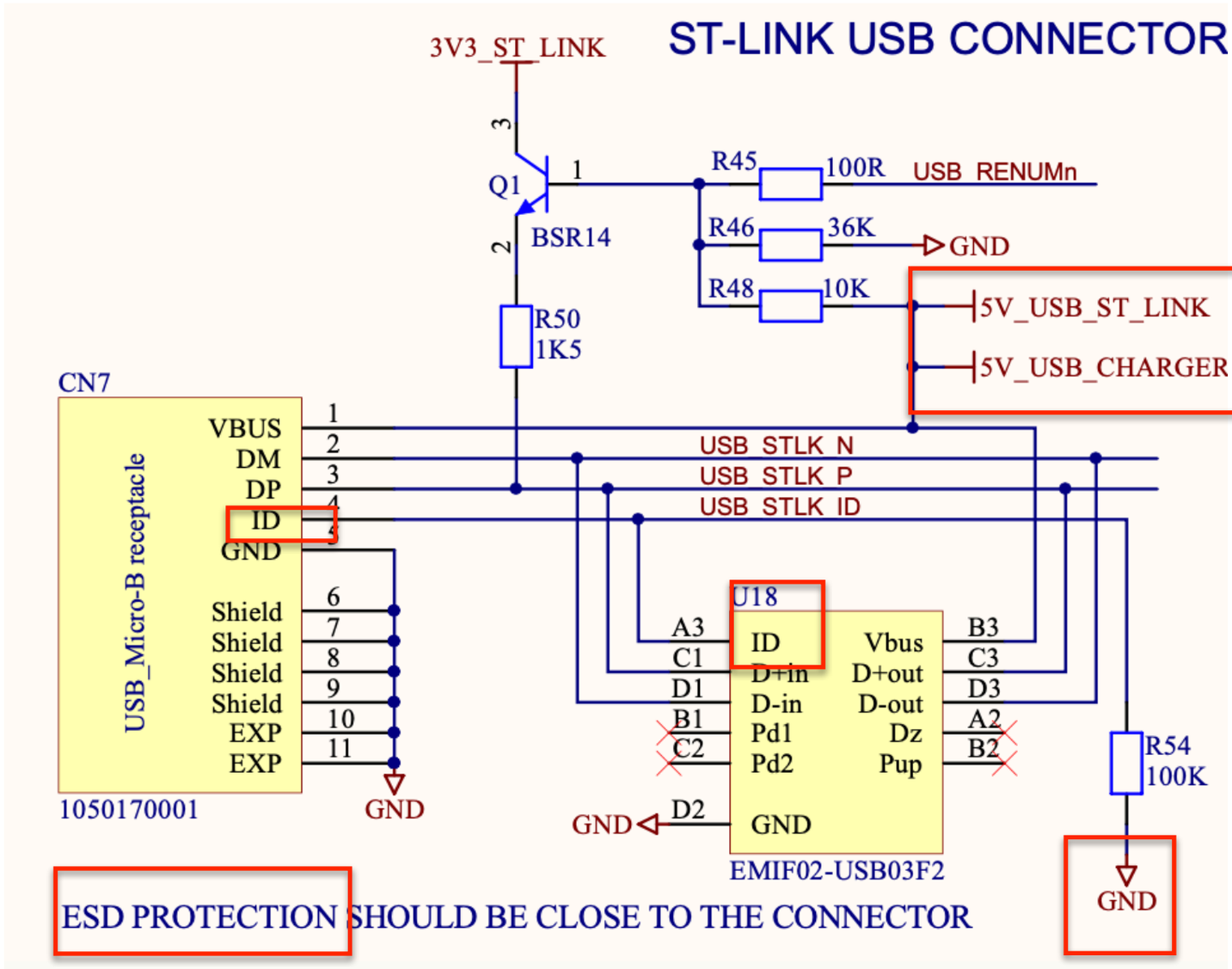
STM324L Discovery Kit IoT Node



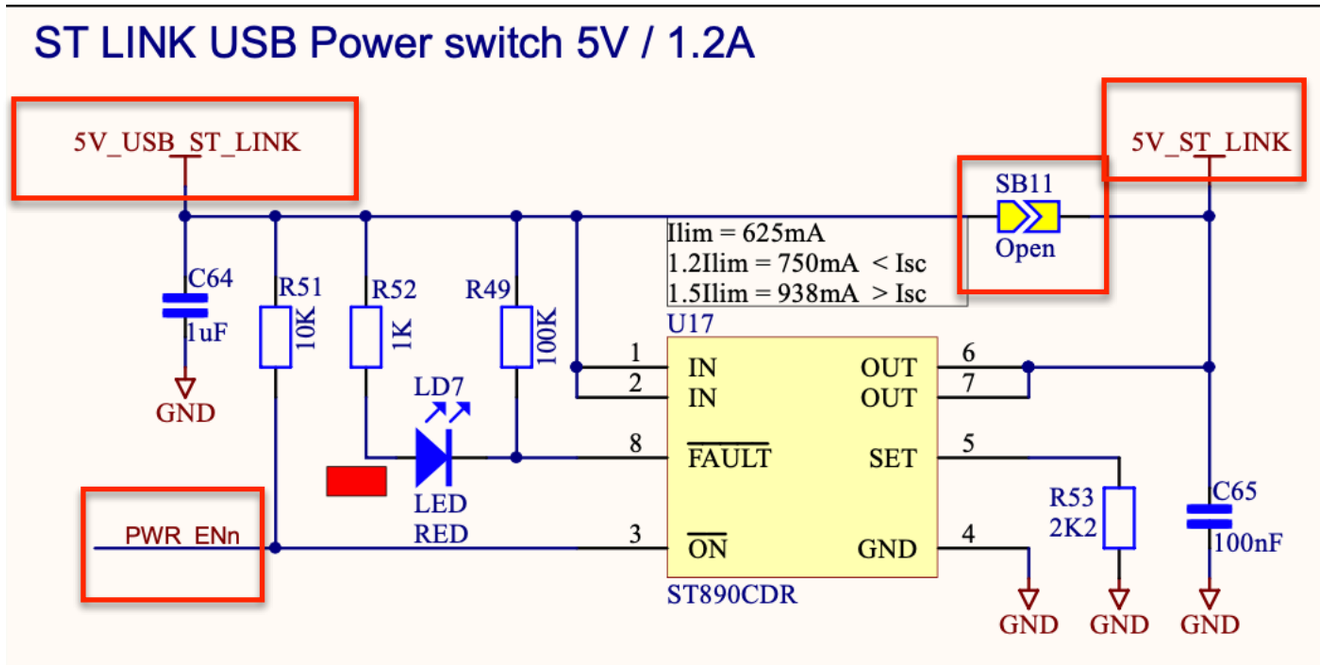
STM32L Discovery Kit
USB_OTF_FS and STLINK



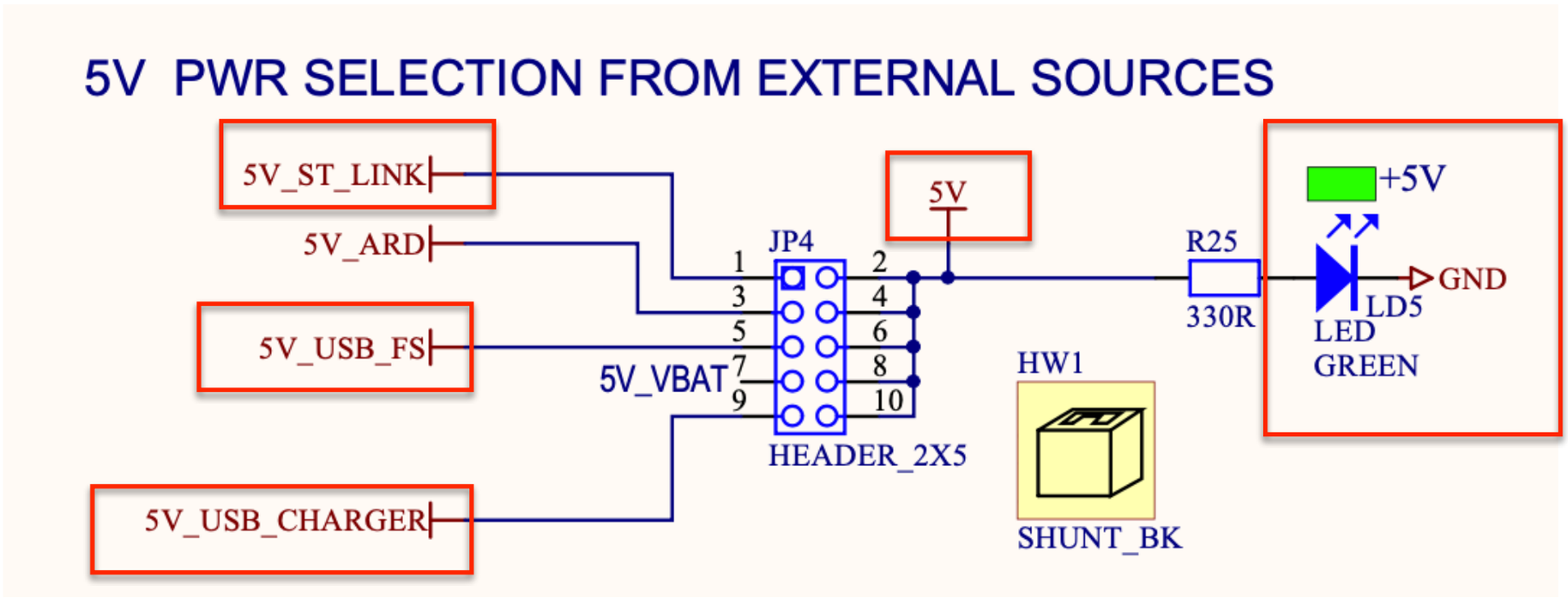
STM32L Discovery Kit
ST-LINK USB Connector



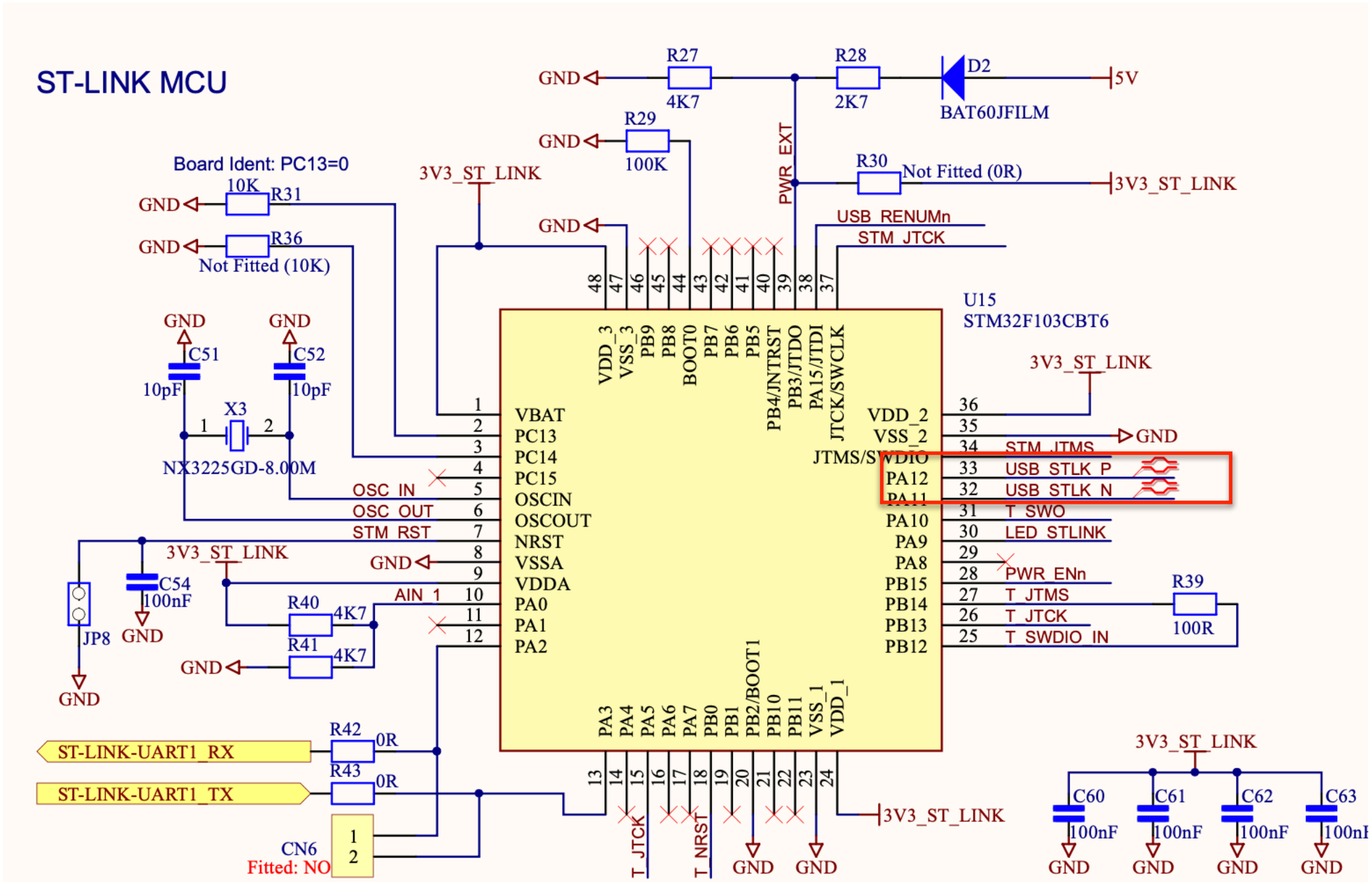
STM32L Discovery Kit
ST-LINK Power Switch



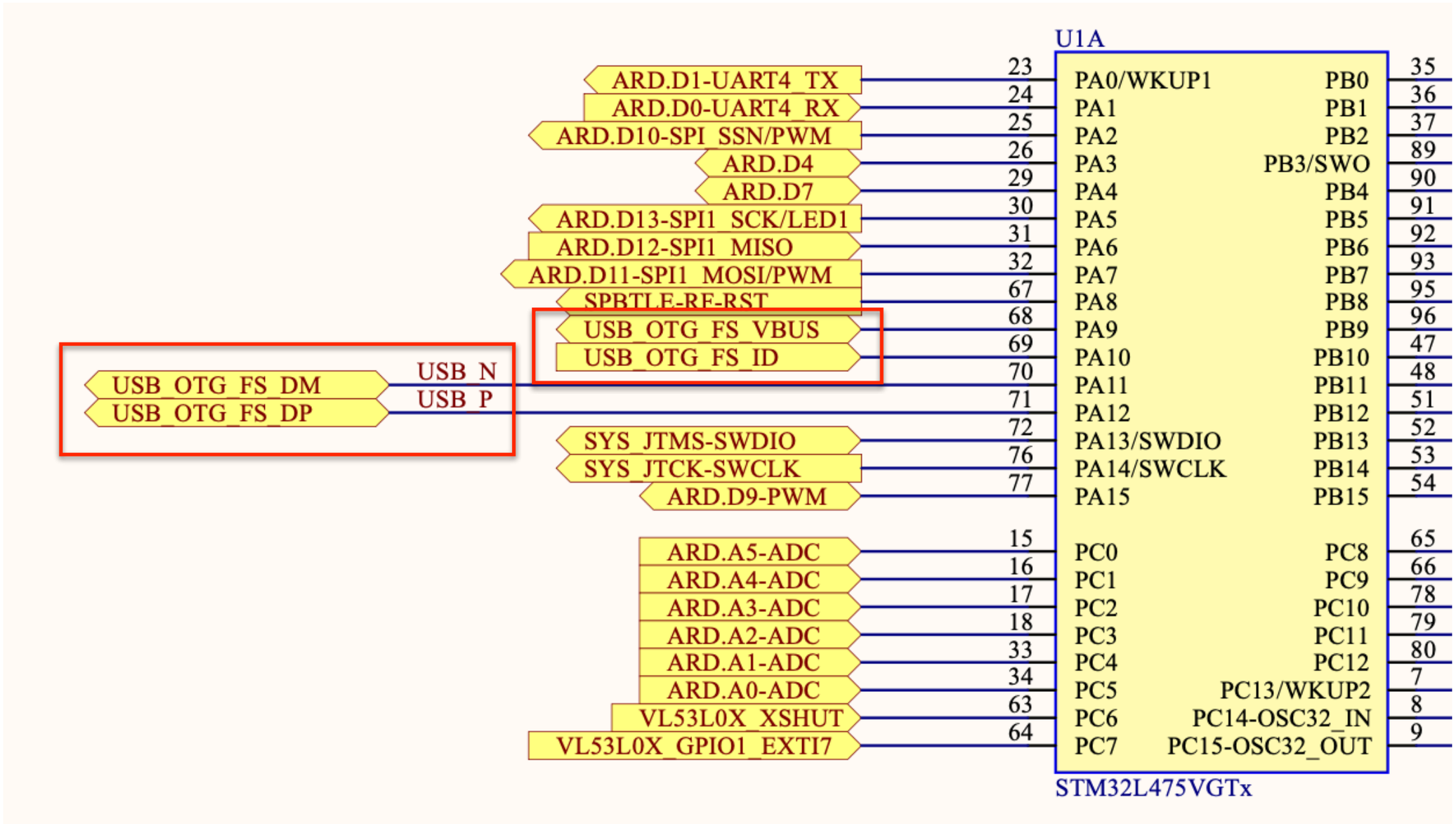
STM32L Discovery Kit
5V PWR



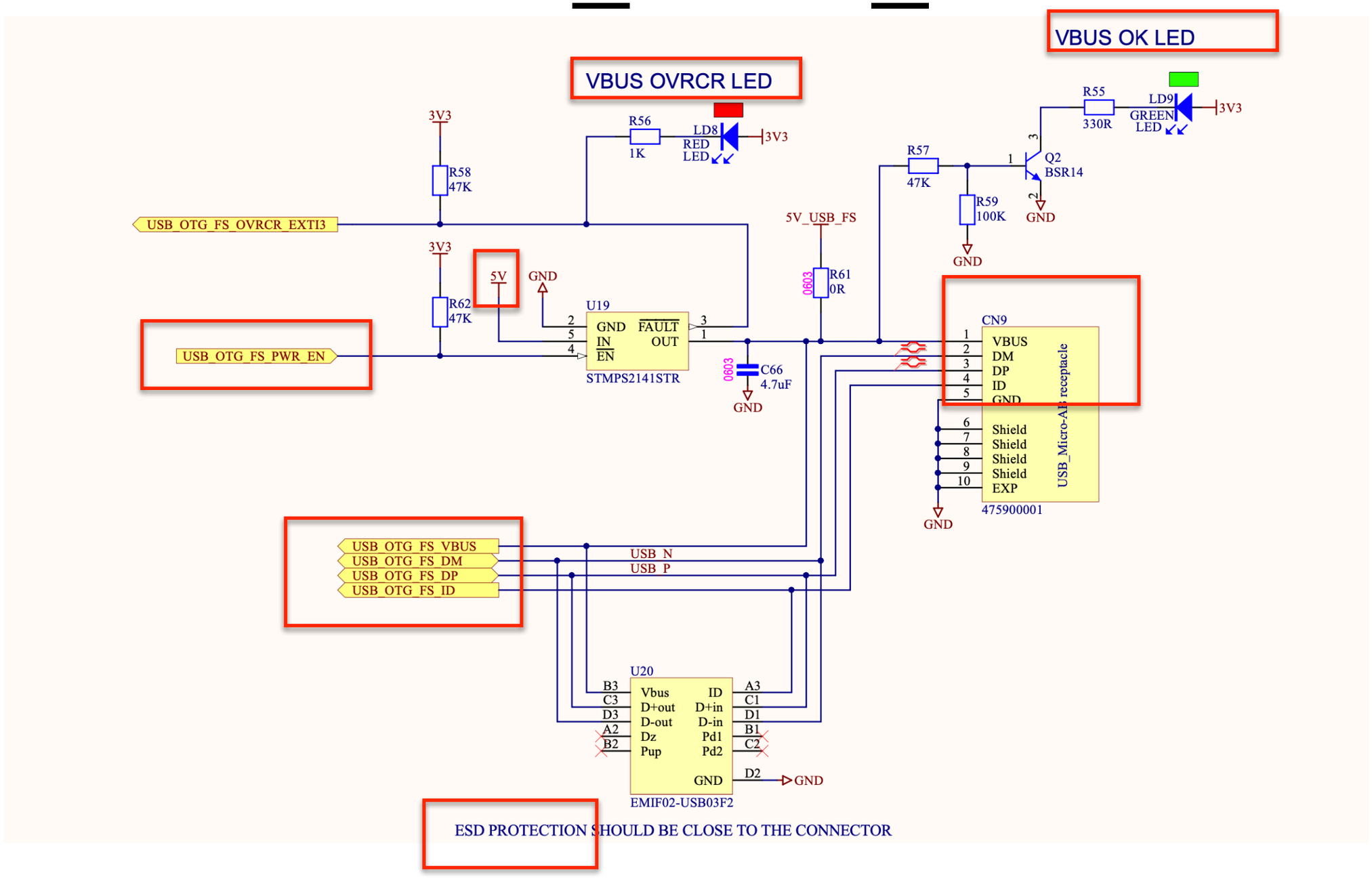
STM32L Discovery Kit
ST-LINK MCU



STM32L Discovery Kit
USB_OTG_FS



STM32L Discovery Kit
USB_OTG_FS

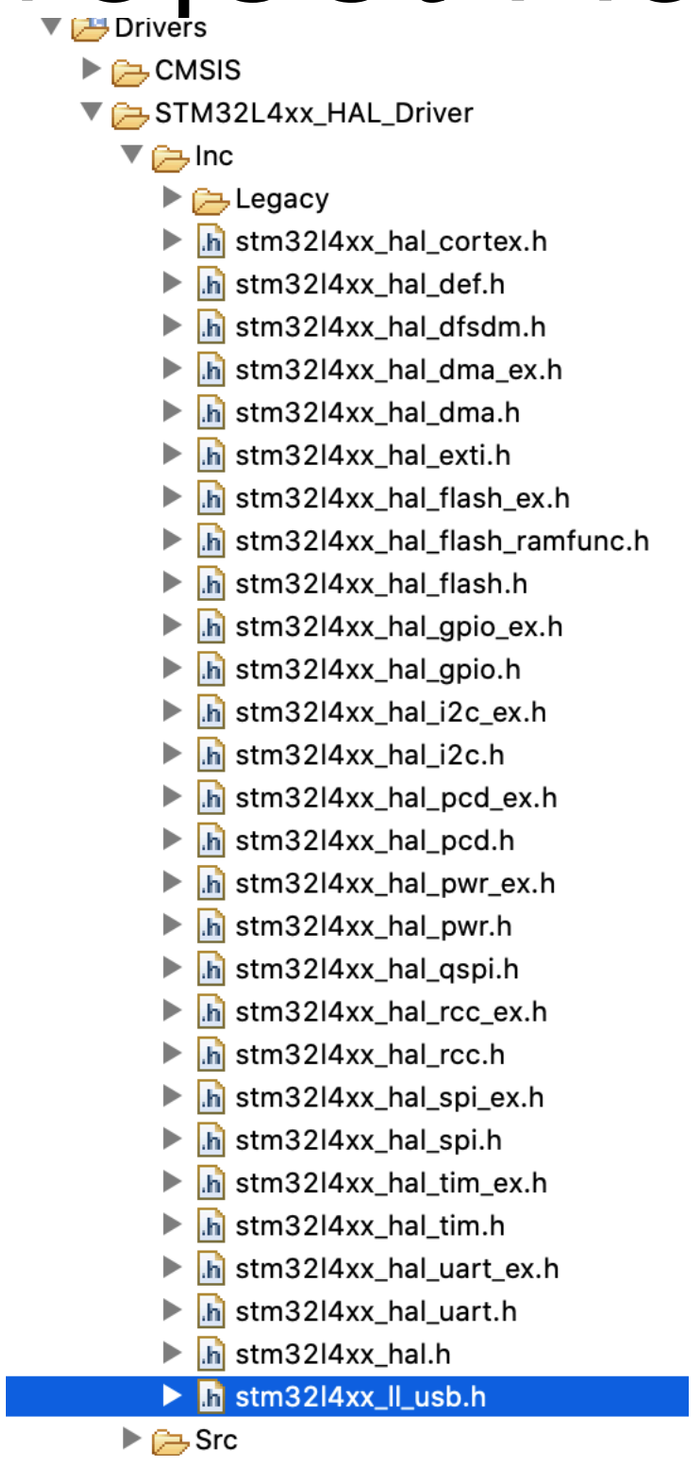


LL USB

Data Structures

Default Project Has LL USB

LL = Low Level



stm32l4xx_ll_usb.h Part 1

```
  */
/* Define to prevent recursive inclusion -----
#ifndef STM32L4xx_LL_USB_H
#define STM32L4xx_LL_USB_H

#ifdef __cplusplus
extern "C" {
#endif

/* Includes ------------------------------------
#include "stm32l4xx_hal_def.h"

#if defined (USB) || defined (USB_OTG_FS)
/** @addtogroup STM32L4xx_HAL_Driver
  * @{
  */

```

stm32l4xx_ll_usb.h
Part 2 - Modes

```
4     */
5   #if defined (USB_OTG_FS)
6
7   typedef enum
8   {
9     USB_DEVICE_MODE  = 0,
0     USB_HOST_MODE    = 1,
1     USB_DRD_MODE     = 2
2   } USB_ModeTypeDef;
3
```

stm32l4xx_ll_usb.h
Part 3 - USB States

```
54 /**
55    * @brief  URB States definition
56    */
57 typedef enum
58 {
59   URB_IDLE = 0,
60   URB_DONE,
61   URB_NOTREADY,
62   URB_NYET,
63   URB_ERROR,
64   URB_STALL
65 } USB_OTG_URBStateTypeDef;
66
```

stm32l4xx_ll_usb.h
Part 4 - Host Channel States

```
66
67  /**
68    * @brief  Host channel States  definition
69    */
70  typedef enum
71  {
72    HC_IDLE = 0,
73    HC_XFRC,
74    HC_HALTED,
75    HC_NAK,
76    HC_NYET,
77    HC_STALL,
78    HC_XACTERR,
79    HC_BBLERR,
80    HC_DATATGLERR
81  } USB_OTG_HCStateTypeDef;
82
```

stm32l4xx_ll_usb.h
Part 5 - USB_OTG_CfgTypeDef

```
82
83  /**
84    * @brief  USB OTG Initialization Structure definition
85    */
86  typedef struct
87  {
88    uint32_t dev_endpoints;           /*!< Device Endpoints number.
89                                           This parameter depends on the used USB core.
90                                           This parameter must be a number between Min_Data = 1 and Max_Data
91
92    uint32_t Host_channels;           /*!< Host Channels number.
93                                           This parameter Depends on the used USB core.
94                                           This parameter must be a number between Min_Data = 1 and Max_Data
95
96    uint32_t speed;                   /*!< USB Core speed.
97                                           This parameter can be any value of @ref USB_Core_Speed_
98
99    uint32_t dma_enable;              /*!< Enable or disable of the USB embedded DMA used only for OTG HS.
100
101   uint32_t ep0_mps;                 /*!< Set the Endpoint 0 Max Packet size.
102
103   uint32_t phy_itface;              /*!< Select the used PHY interface.
104                                          This parameter can be any value of @ref USB_Core_PHY
```

stm32l4xx_ll_usb.h
Part 6 - USB_OTG_**EP**TypeDef

```
120
121 typedef struct
122  {
123    uint8_t   num;                /*!< Endpoint number
124                                       This parameter must be a number between Min_Data = 1 and Max_Data =
125
126    uint8_t   is_in;              /*!< Endpoint direction
127                                       This parameter must be a number between Min_Data = 0 and Max_Data =
128
129    uint8_t   is_stall;           /*!< Endpoint stall condition
130                                       This parameter must be a number between Min_Data = 0 and Max_Data =
131
132   uint8_t   type;               /*!< Endpoint type
133                                       This parameter can be any value of @ref USB_EP_Type_
134
135   uint8_t   data_pid_start;     /*!< Initial data PID
136                                       This parameter must be a number between Min_Data = 0 and Max_Data =
137
138   uint8_t   even_odd_frame;     /*!< IFrame parity
139                                       This parameter must be a number between Min_Data = 0 and Max_Data =
140
141   uint16_t  tx_fifo_num;        /*!< Transmission FIFO number
```

EP = Endpoint

stm32l4xx_ll_usb.h
Part 7 - USB_OTG_HC_TypeDef

```
156 typedef struct
157 {
158   uint8_t   dev_addr;          /*!< USB device address.
159                                     This parameter must be a number between Min_Data = 1 and Max_Data = 2
160
161   uint8_t   ch_num;            /*!< Host channel number.
162                                     This parameter must be a number between Min_Data = 1 and Max_Data = 1
163
164   uint8_t   ep_num;            /*!< Endpoint number.
165                                     This parameter must be a number between Min_Data = 1 and Max_Data = 1
166
167   uint8_t   ep_is_in;          /*!< Endpoint direction
168                                     This parameter must be a number between Min_Data = 0 and Max_Data = 1
169
170   uint8_t   speed;             /*!< USB Host speed.
171                                     This parameter can be any value of @ref USB_Core_Speed_
172
173   uint8_t   do_ping;           /*!< Enable or disable the use of the PING protocol for HS mode.
174
175   uint8_t   process_ping;      /*!< Execute the PING protocol for HS mode.
176
177   uint8_t   ep_type;           /*!< Endpoint Type.
```

USB LL API

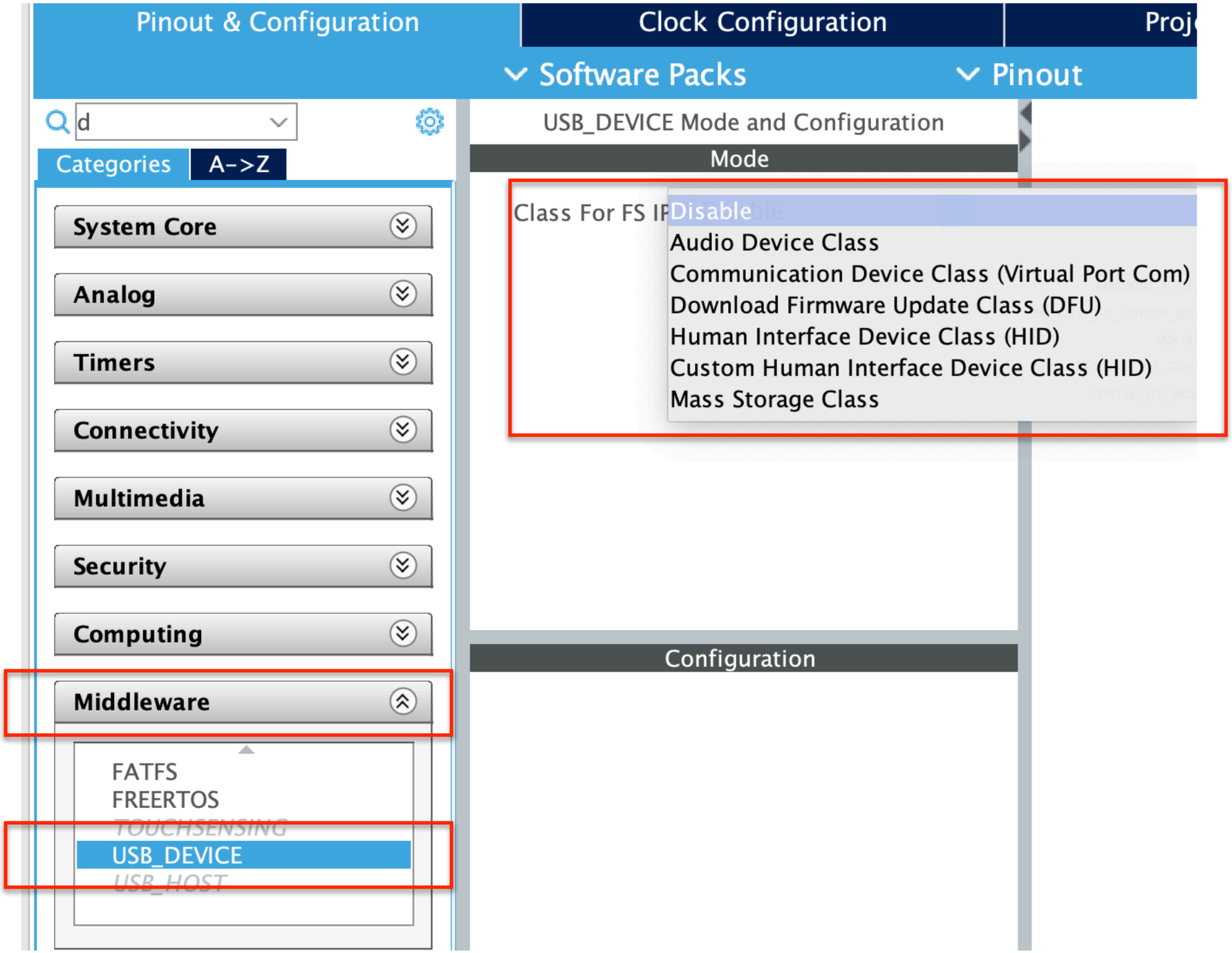
Functions

USB_...

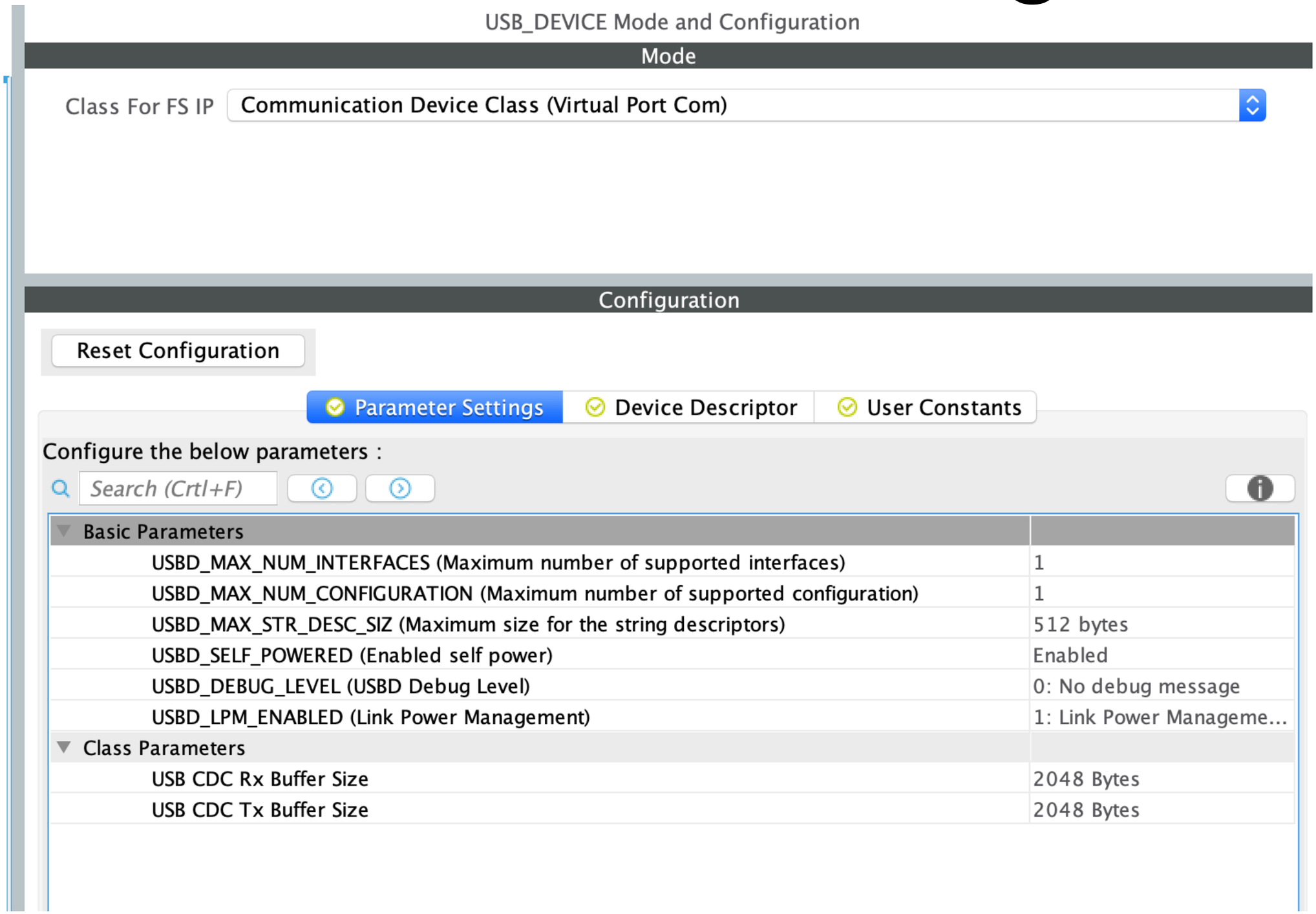
```
541 #if defined (USB_OTG_FS)
542 HAL_StatusTypeDef USB_CoreInit(USB_OTG_GlobalTypeDef *USBx, USB_OTG_CfgTypeDef cfg);
543 HAL_StatusTypeDef USB_DevInit(USB_OTG_GlobalTypeDef *USBx, USB_OTG_CfgTypeDef cfg);
544 HAL_StatusTypeDef USB_EnableGlobalInt(USB_OTG_GlobalTypeDef *USBx);
545 HAL_StatusTypeDef USB_DisableGlobalInt(USB_OTG_GlobalTypeDef *USBx);
546 HAL_StatusTypeDef USB_SetTurnaroundTime(USB_OTG_GlobalTypeDef *USBx, uint32_t hclk, uint8_t speed);
547 HAL_StatusTypeDef USB_SetCurrentMode(USB_OTG_GlobalTypeDef *USBx, USB_ModeTypeDef mode);
548 HAL_StatusTypeDef USB_SetDevSpeed(USB_OTG_GlobalTypeDef *USBx, uint8_t speed);
549 HAL_StatusTypeDef USB_FlushRxFifo(USB_OTG_GlobalTypeDef *USBx);
550 HAL_StatusTypeDef USB_FlushTxFifo(USB_OTG_GlobalTypeDef *USBx, uint32_t num);
551 HAL_StatusTypeDef USB_ActivateEndpoint(USB_OTG_GlobalTypeDef *USBx, USB_OTG_EPTypeDef *ep);
552 HAL_StatusTypeDef USB_DeactivateEndpoint(USB_OTG_GlobalTypeDef *USBx, USB_OTG_EPTypeDef *ep);
553 HAL_StatusTypeDef USB_ActivateDedicatedEndpoint(USB_OTG_GlobalTypeDef *USBx, USB_OTG_EPTypeDef *ep);
554 HAL_StatusTypeDef USB_DeactivateDedicatedEndpoint(USB_OTG_GlobalTypeDef *USBx, USB_OTG_EPTypeDef *ep)
555 HAL_StatusTypeDef USB_EPStartXfer(USB_OTG_GlobalTypeDef *USBx, USB_OTG_EPTypeDef *ep);
556 HAL_StatusTypeDef USB_EP0StartXfer(USB_OTG_GlobalTypeDef *USBx, USB_OTG_EPTypeDef *ep);
557 HAL_StatusTypeDef USB_WritePacket(USB_OTG_GlobalTypeDef *USBx, uint8_t *src, uint8_t ch_ep_num, uint1
558 void             *USB_ReadPacket(USB_OTG_GlobalTypeDef *USBx, uint8_t *dest, uint16_t len);
559 HAL_StatusTypeDef USB_EPSetStall(USB_OTG_GlobalTypeDef *USBx, USB_OTG_EPTypeDef *ep);
560 HAL_StatusTypeDef USB_EPClearStall(USB_OTG_GlobalTypeDef *USBx, USB_OTG_EPTypeDef *ep);
561 HAL_StatusTypeDef USB_SetDevAddress(USB_OTG_GlobalTypeDef *USBx, uint8_t address);
562 HAL_StatusTypeDef USB_DevConnect(USB_OTG_GlobalTypeDef *USBx);
563 HAL_StatusTypeDef USB_DevDisconnect(USB_OTG_GlobalTypeDef *USBx);
564 HAL_StatusTypeDef USB_StopDevice(USB_OTG_GlobalTypeDef *USBx);
565 HAL_StatusTypeDef USB_ActivateSetup(USB_OTG_GlobalTypeDef *USBx);
```

Middleware - USB device

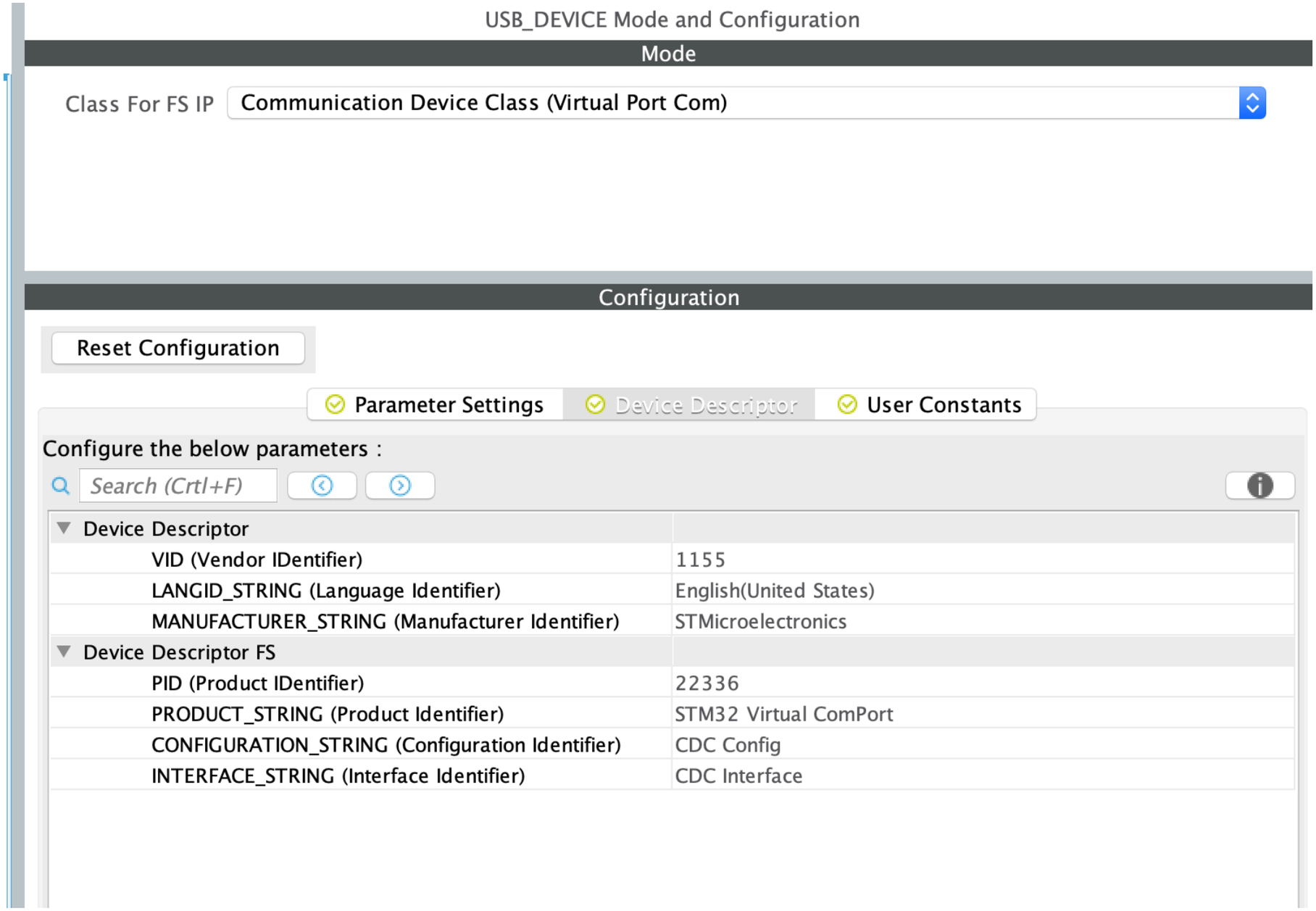
Middleware



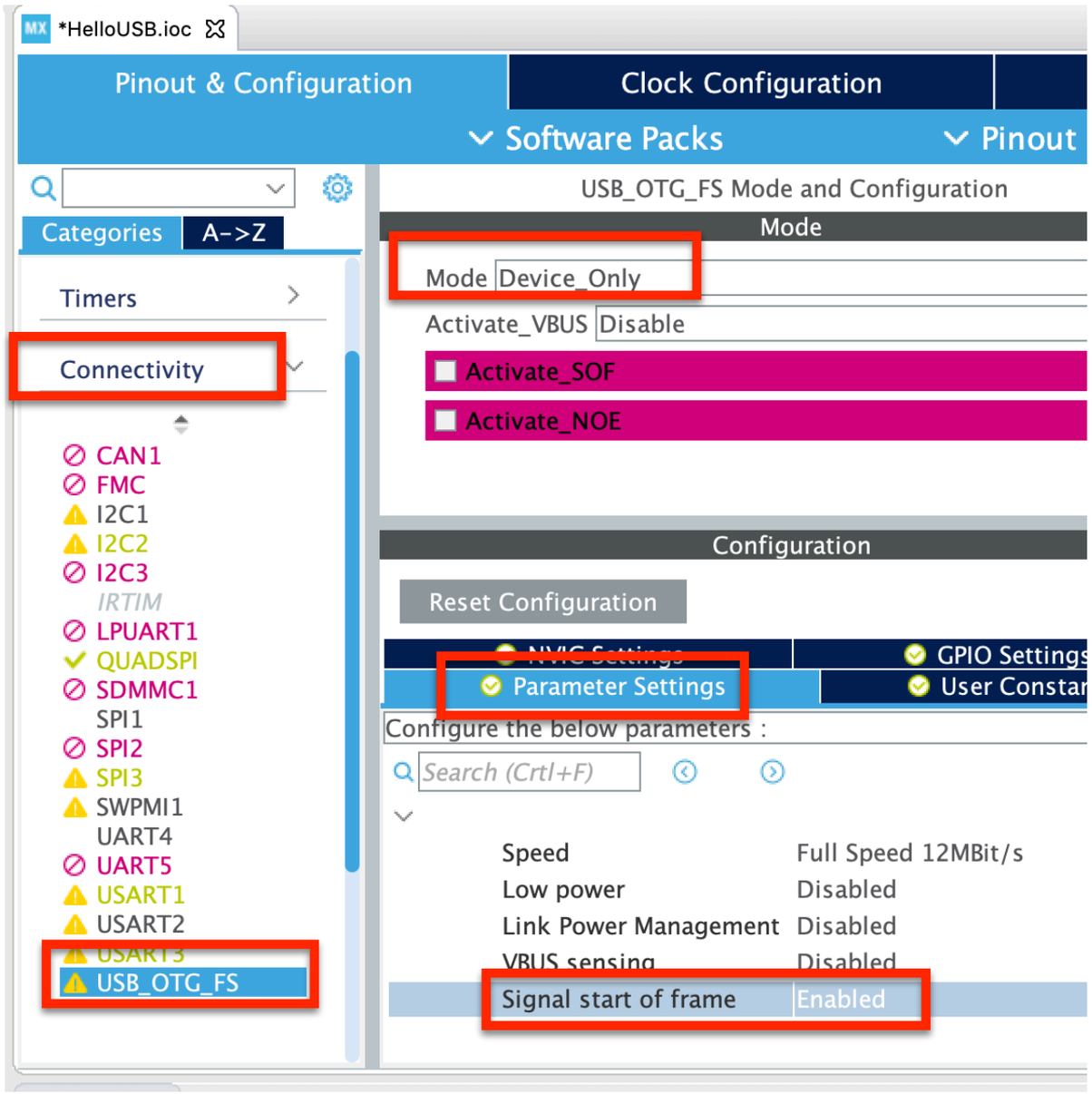
Virtual Port ComDevice Parameter Settings



Virtual Port ComDevice Device Descriptor



Connectivity - USB_OTG_FS - Signal Start of Frame - **Enabled**



USB

Hands-On Project

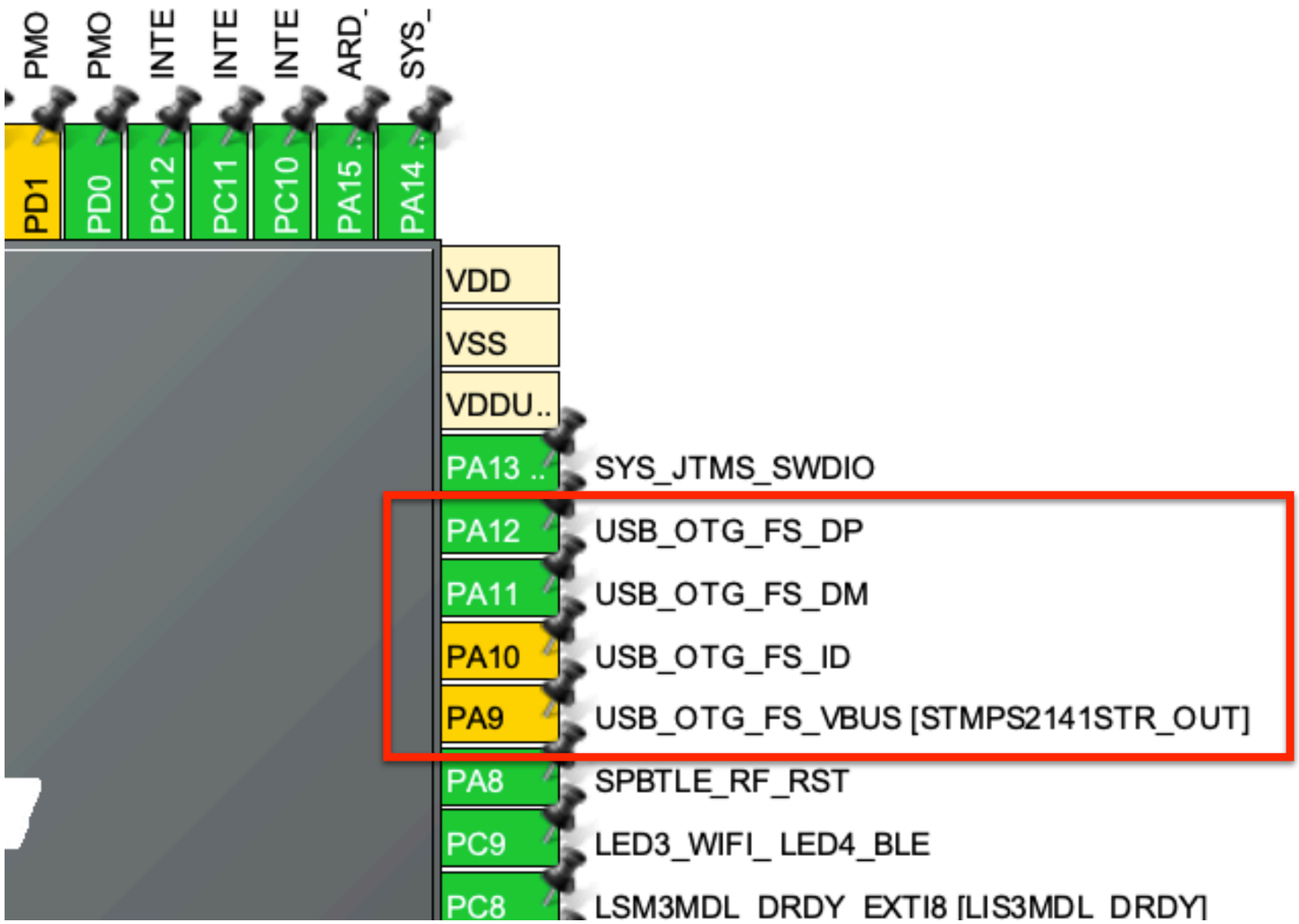
USB Hands-On Project Overview

- The goal of this project is to give you hands-on experience with adding USB Virtual Com Port to your device
  - Use the Middleware included with STMCubeIDE
- 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

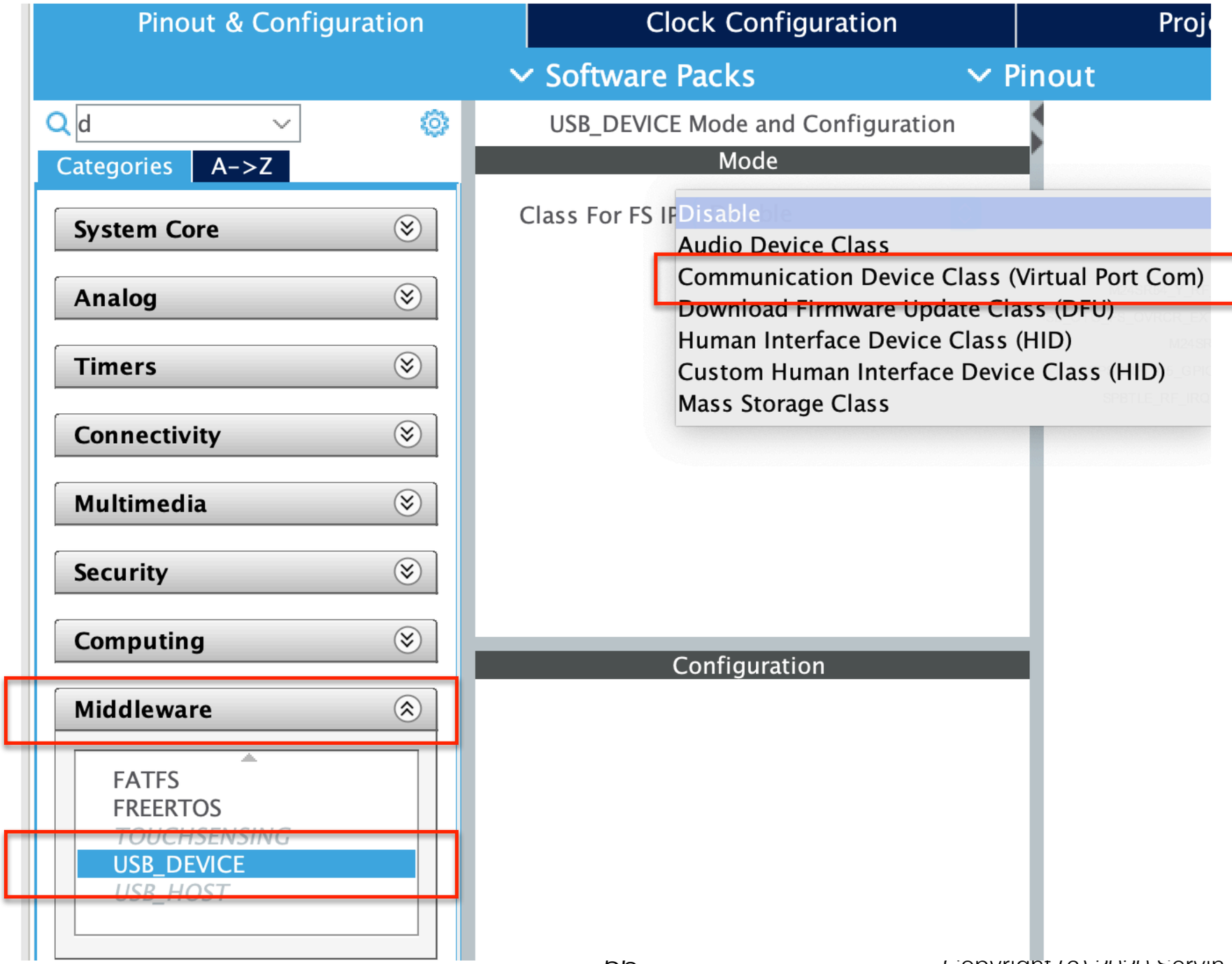
USB Hands-On Project User Stories

- **User Story 1** - Add the USB Virtual Console Code to your project
    - USB device
- **User Story 2** - When you plug your device into a USB port, you should see a new COM port appears

Project Setup - Part 1
Keep Defaults



Add Middleware for Virtual Com Port



Virtual Port ComDevice Parameter Settings - **Use Defaults**

USB_DEVICE Mode and Configuration

Mode

Class For FS IP: Communication Device Class (Virtual Port Com)

Configuration

Reset Configuration

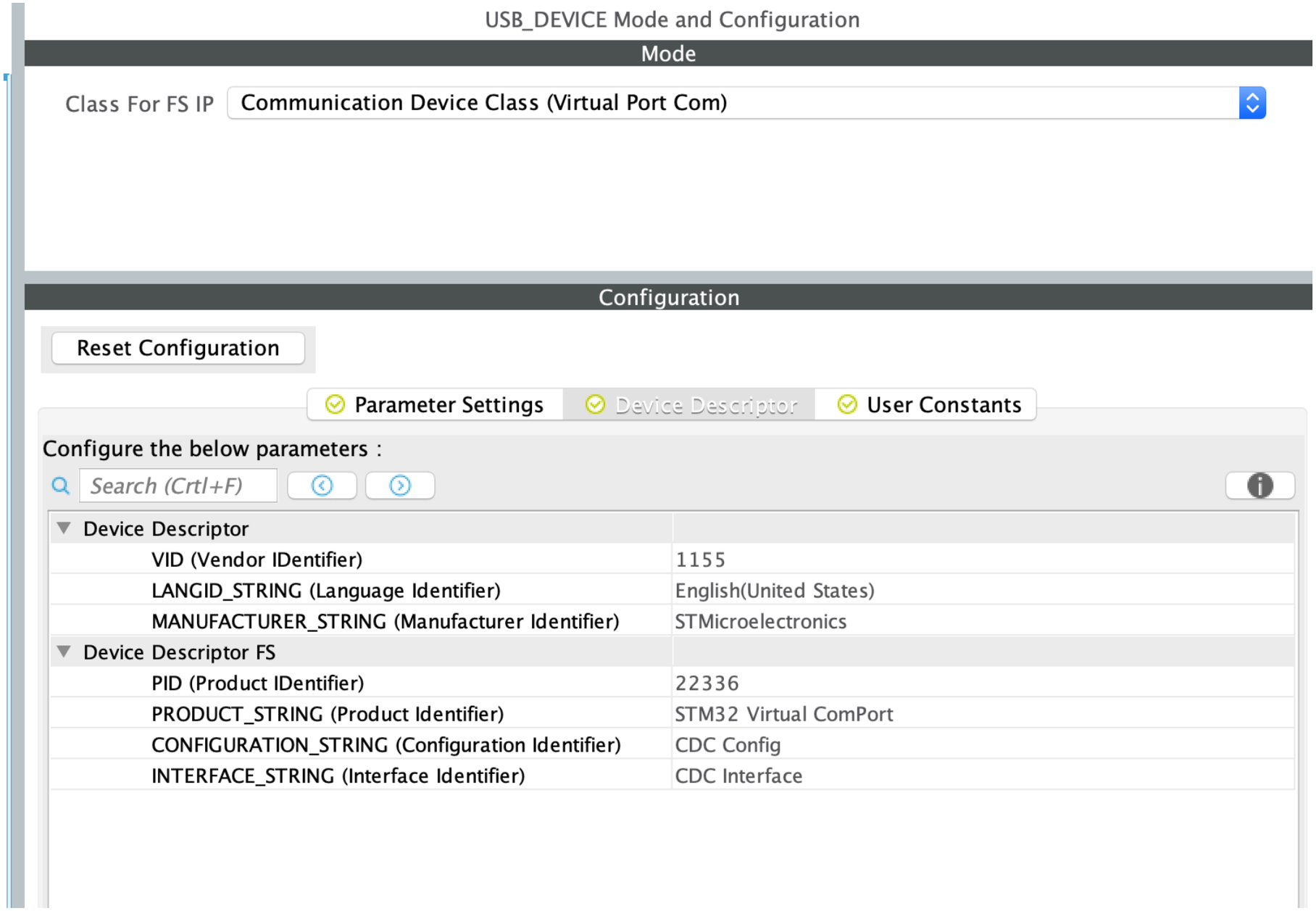
Parameter Settings | Device Descriptor | User Constants

Configure the below parameters :

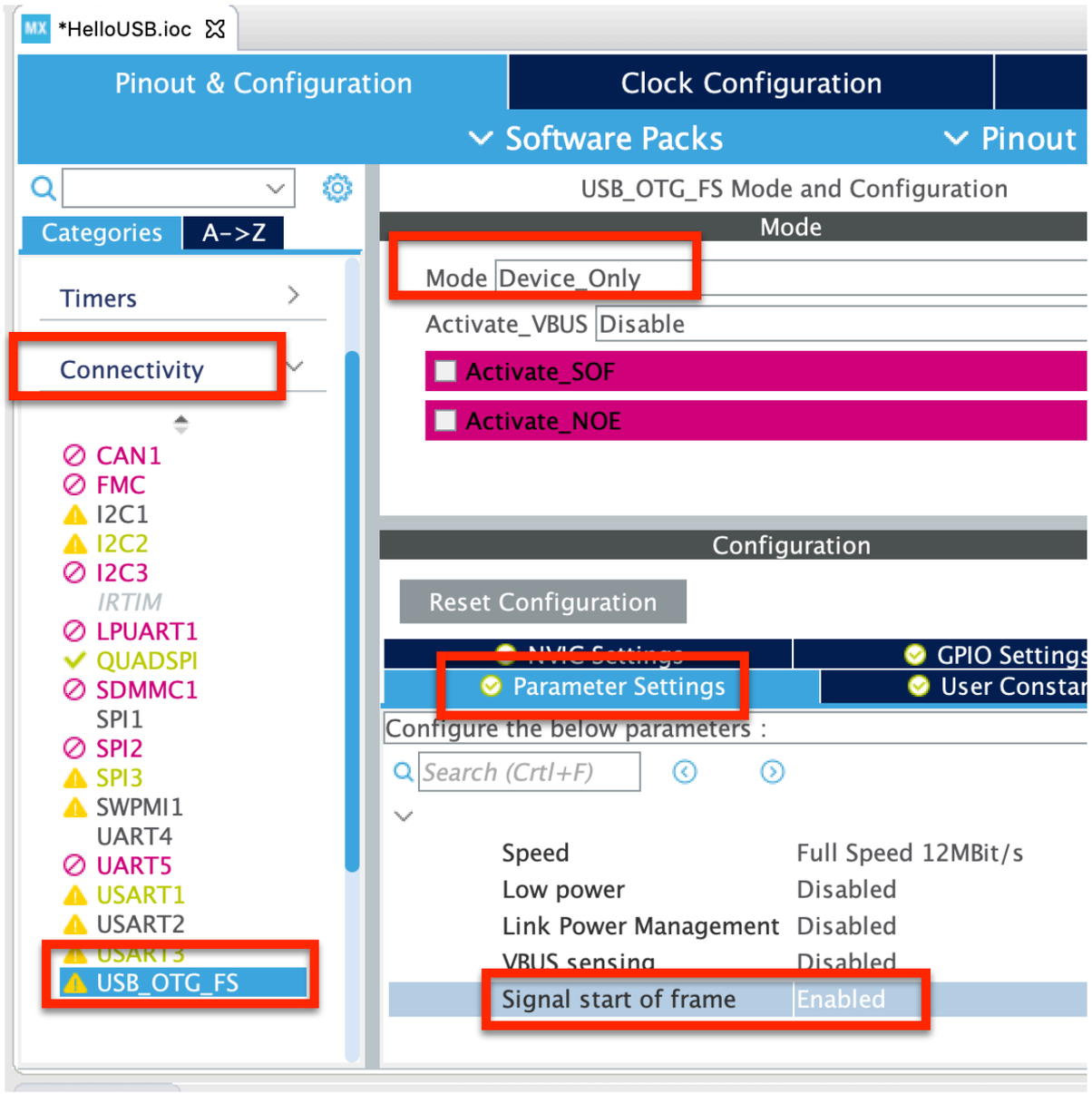
Search (Crtl+F)

Basic Parameters	
USBD_MAX_NUM_INTERFACES (Maximum number of supported interfaces)	1
USBD_MAX_NUM_CONFIGURATION (Maximum number of supported configuration)	1
USBD_MAX_STR_DESC_SIZ (Maximum size for the string descriptors)	512 bytes
USBD_SELF_POWERED (Enabled self power)	Enabled
USBD_DEBUG_LEVEL (USBD Debug Level)	0: No debug message
USBD_LPM_ENABLED (Link Power Management)	1: Link Power Manageme...
Class Parameters	
USB CDC Rx Buffer Size	2048 Bytes
USB CDC Tx Buffer Size	2048 Bytes

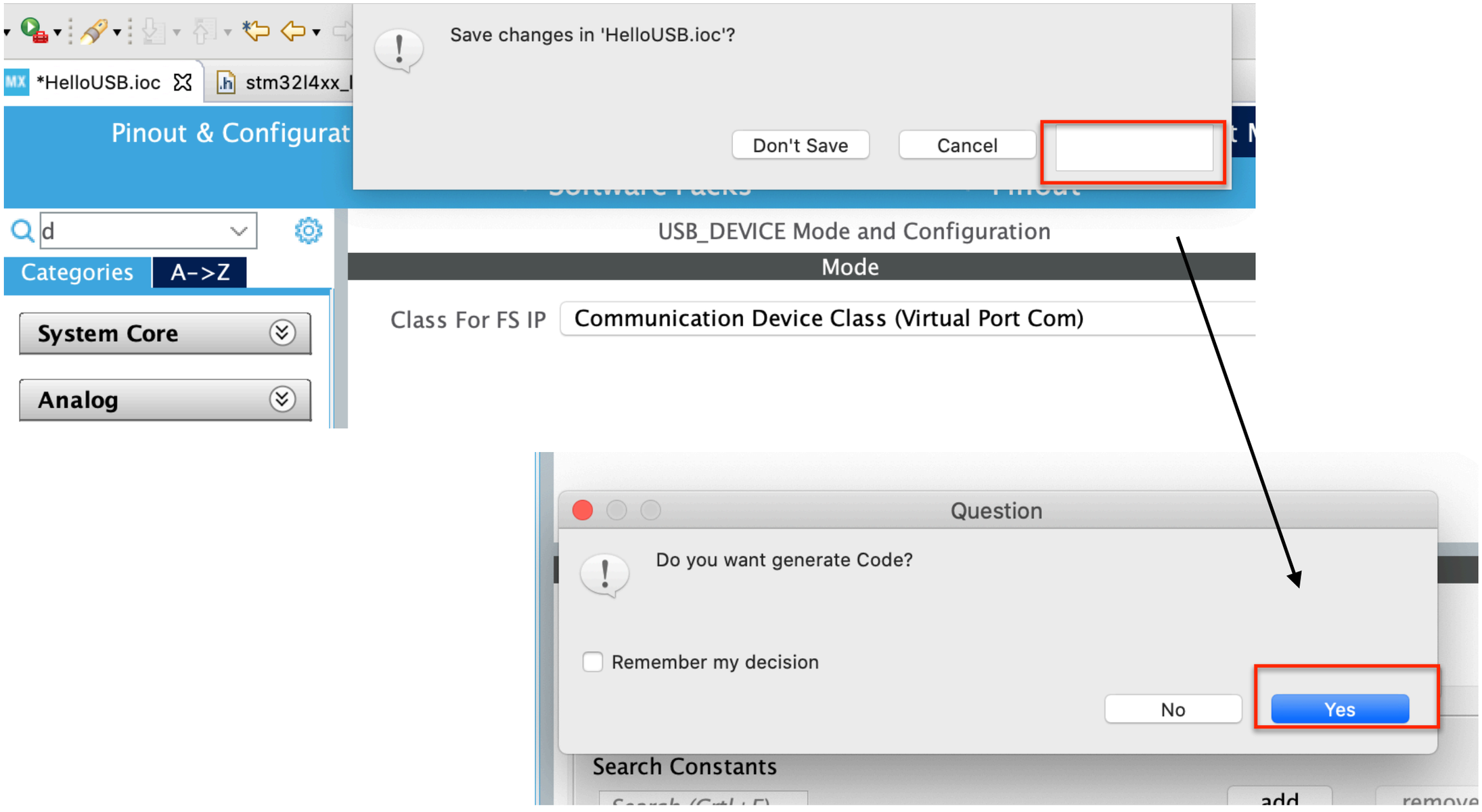
Virtual Port ComDevice Device Descriptor - **Use Defaults**



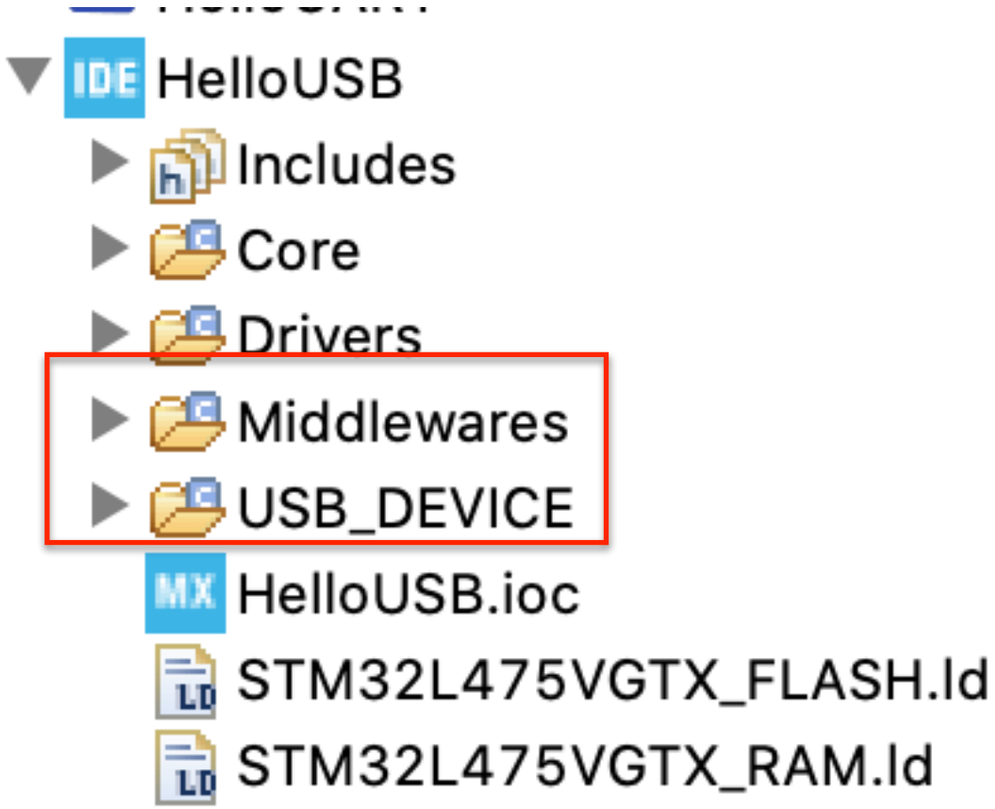
Connectivity - USB_OTG_FS - Signal Start of Frame - **Enabled**

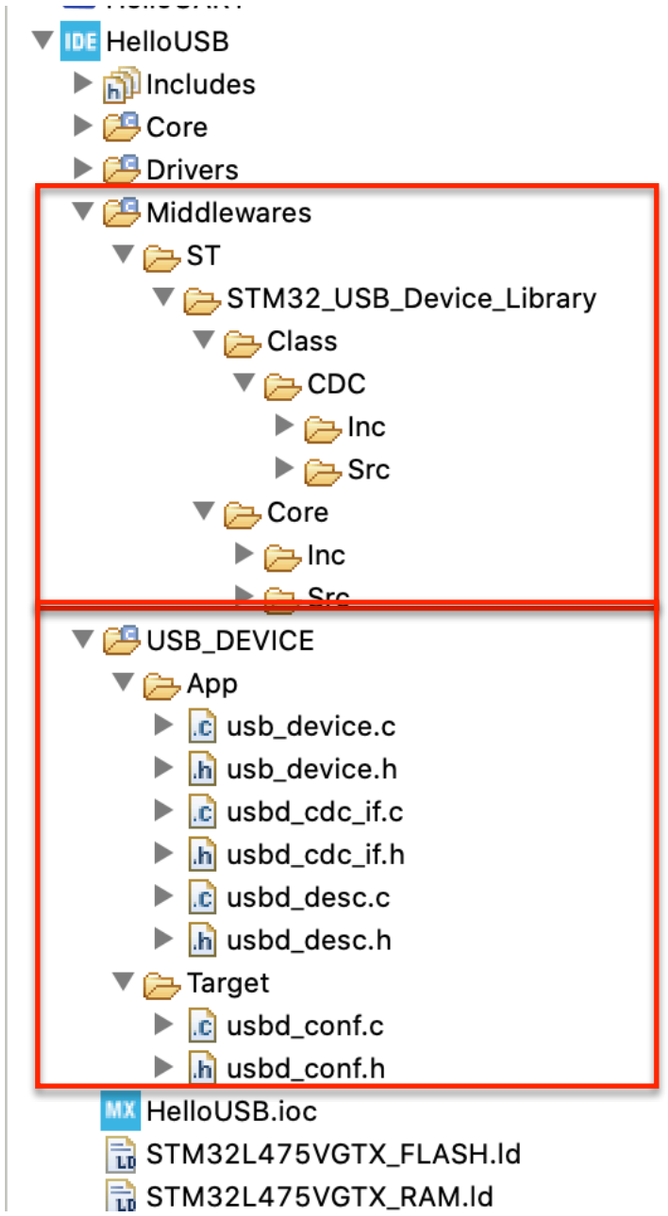


Save Changes and Generate Code



Result





Add Code to main.c

```
190   int count = 0;
191   while (1)
192   {
193
194     count++;
195
196     char buf[100];
197     snprintf(buf, sizeof(buf), "count: %d\r\n", count);
198
199     uint8_t status = CDC_Transmit_FS((uint8_t*) buf, strlen(buf));
200
201     char buf2[100];
202     snprintf(buf2, sizeof(buf), "%i. CDC_Transmit_FS: status: %u\r\n", count, status);
203     HAL_UART_Transmit(&huart1, (uint8_t *)buf2, strlen(buf2), 1000);
204
205     HAL_Delay(1000);
206
207     /* USER CODE END WHILE */
208
209     /* USER CODE BEGIN 3 */
210   }
211   /* USER CODE END 3 */
212 }
```

Build and Run

- When you plug your USB port into a PC you will see a new COM Port appear!

- When the COM port appears, start a new terminal session at 115200 and you will see “count: …” appearing on the console!

Bonus Option
Add code to do_usb_receive()

```
 90
 91 nt8_t do_usb_init() {
 92     return 0;
 93
 94
 95 nt8_t do_usb_control(uint8_t cmd, uint8_t *pbuf, uint16_t length) {
 96     return 0;
 97
 98
 99 nt8_t do_usb_receive(uint8_t *buf, uint32_t *length) {
100     return 0;
101
102
103 nt8_t do_usb_transmit_complete(uint8_t *buf, uint32_t *length, uint8_t epnum) {
104     return 0;
105
106
```

```
172
173   //extern USBD_CDC_ItfTypeDef USBD_Interface_fops_FS;
174
175   USBD_Interface_fops_FS.Init = do_usb_init;
176   USBD_Interface_fops_FS.Control = do_usb_control;
177   USBD_Interface_fops_FS.Receive = do_usb_receive;
178   USBD_Interface_fops_FS.TransmitCplt = do_usb_transmit_complete;
179
```

Summary

- Concepts
- Data Sheet - STM32L475
- User Manual - UM2153 - STM32L Discovery Kit for IoT
- Schematics - STM32L Discovery Kit for IoT
- API - STM32L LL (Low Level) No HAL
    - Data Structures
    - Functions
    - Middleware for USB
- Hands-On Project