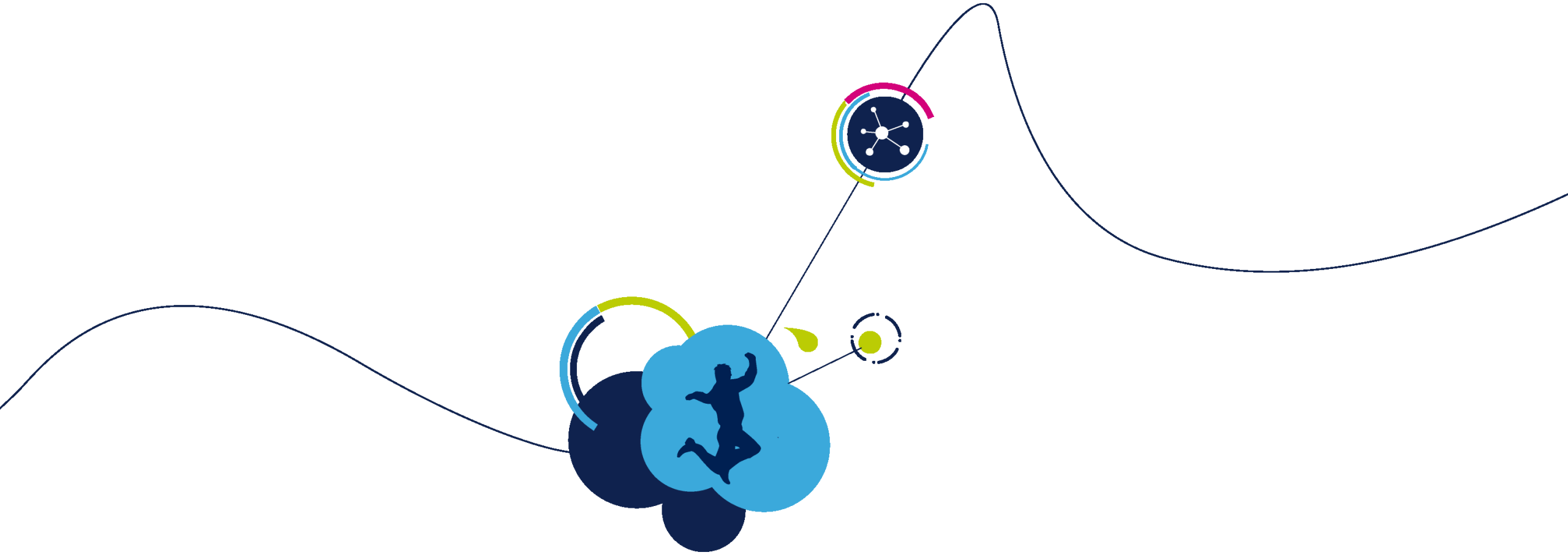
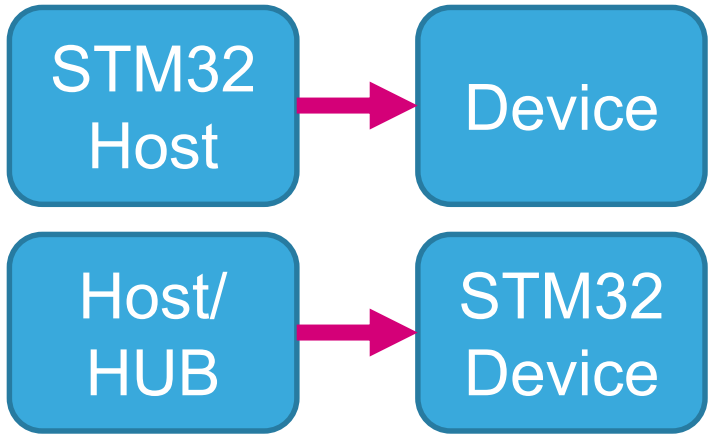


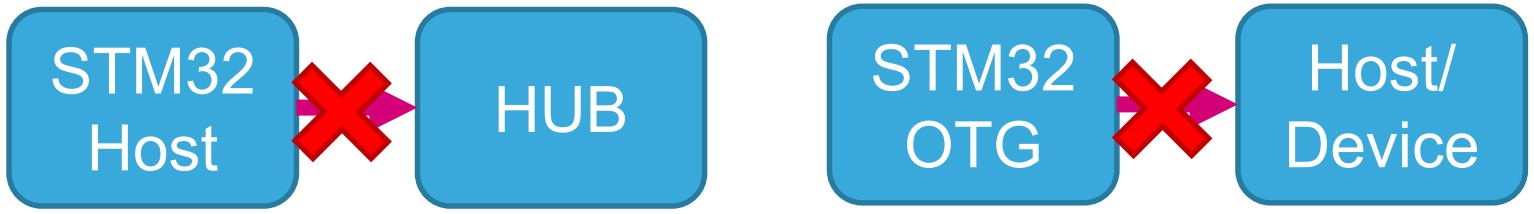
STM32 USB library

USB library overview

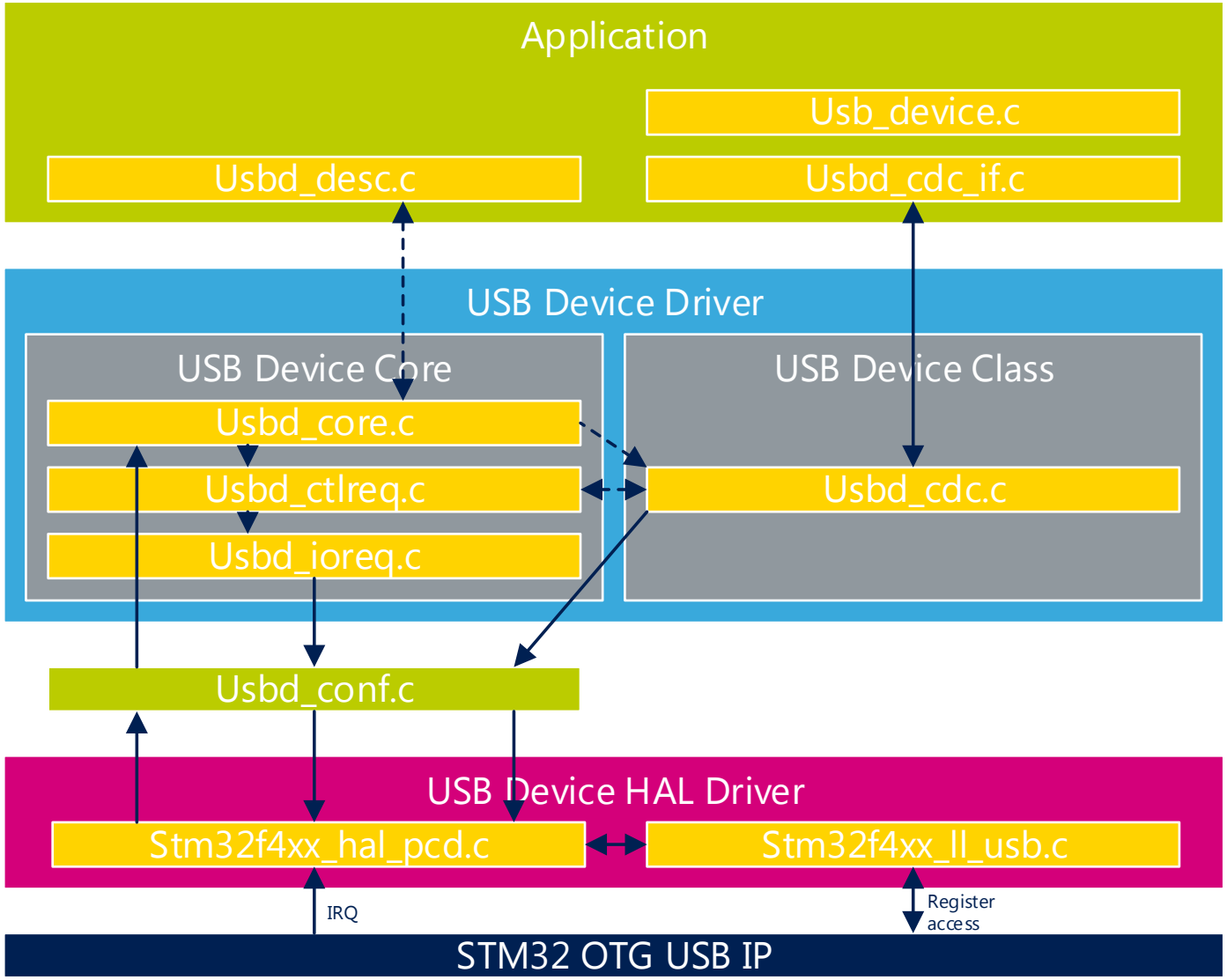
- Integration with CubeMX
  - Integration with FatFs when mass-storage host is used
- Supported configurations:



- Unsupported configurations (supported in HW, but not in library):



Device library organization



USB Device Handle connections


```c
/* USB Device handle structure */
typedef struct _USBD_HandleTypeDef
{
  uint8_t                 id;
  uint32_t                dev_config;
  uint32_t                dev_default_config;
  uint32_t                dev_config_status;
  USBD_SpeedTypeDef       dev_speed;
  USBD_EndpointTypeDef    ep_in[15];
  USBD_EndpointTypeDef    ep_out[15];
  uint32_t                ep0_state;
  uint32_t                ep0_data_len;
  uint8_t                 dev_state;
  uint8_t                 dev_old_state;
  uint8_t                 dev_address;
  uint8_t                 dev_connection_status;
  uint8_t                 dev_test_mode;
  uint32_t                dev_remote_wakeup;

  USBD_SetupReqTypedef    request;
  USBD_DescriptorsTypeDef *pDesc;
  USBD_ClassTypeDef       *pClass;
  void                    *pClassData;
  void                    *pUserData;
  void                    *pData;
} USBD_HandleTypeDef;
```

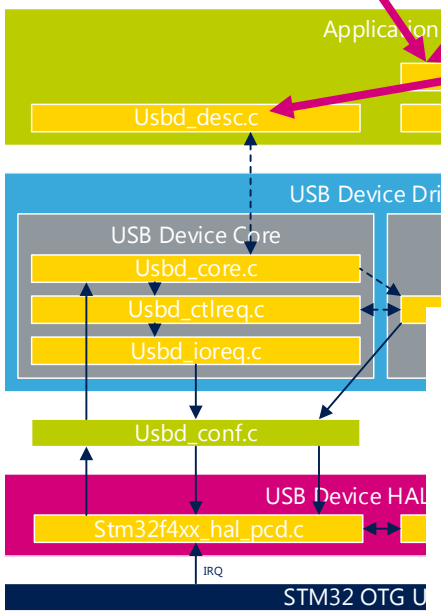
```c
/* USB Device descriptors structure */
typedef struct
{
  uint8_t  *(*GetDeviceDescriptor)( USBD_SpeedTypeDef speed , uint16_t *length);
  uint8_t  *(*GetLangIDStrDescriptor)( USBD_SpeedTypeDef speed , uint16_t *length);
  uint8_t  *(*GetManufacturerStrDescriptor)( USBD_SpeedTypeDef speed , uint16_t *length);
  uint8_t  *(*GetProductStrDescriptor)( USBD_SpeedTypeDef speed , uint16_t *length);
  uint8_t  *(*GetSerialStrDescriptor)( USBD_SpeedTypeDef speed , uint16_t *length);
  uint8_t  *(*GetConfigurationStrDescriptor)( USBD_SpeedTypeDef speed , uint16_t *length);
  uint8_t  *(*GetInterfaceStrDescriptor)( USBD_SpeedTypeDef speed , uint16_t *length);
#if (USBD_LPM_ENABLED == 1)
  uint8_t  *(*GetBOSDescriptor)( USBD_SpeedTypeDef speed , uint16_t *length);
#endif
} USBD_DescriptorsTypeDef;
```

```c
typedef struct _Device_cb
{
  uint8_t  (*Init)             (struct _USBD_HandleTypeDef *pdev , uint8_t cfgidx);
  uint8_t  (*DeInit)           (struct _USBD_HandleTypeDef *pdev , uint8_t cfgidx);
 /* Control Endpoints*/
  uint8_t  (*Setup)            (struct _USBD_HandleTypeDef *pdev , USBD_SetupReqTypedef  *req)
  uint8_t  (*EP0_TxSent)       (struct _USBD_HandleTypeDef *pdev );
  uint8_t  (*EP0_RxReady)      (struct _USBD_HandleTypeDef *pdev );
  /* Class Specific Endpoints*/
  uint8_t  (*DataIn)           (struct _USBD_HandleTypeDef *pdev , uint8_t epnum);
  uint8_t  (*DataOut)          (struct _USBD_HandleTypeDef *pdev , uint8_t epnum);
  uint8_t  (*SOF)              (struct _USBD_HandleTypeDef *pdev);
  uint8_t  (*IsoINIncomplete)  (struct _USBD_HandleTypeDef *pdev , uint8_t epnum);
  uint8_t  (*IsoOUTIncomplete) (struct _USBD_HandleTypeDef *pdev , uint8_t epnum);

  uint8_t  *(*GetHSConfigDescriptor)(uint16_t *length);
  uint8_t  *(*GetFSConfigDescriptor)(uint16_t *length);
  uint8_t  *(*GetOtherSpeedConfigDescriptor)(uint16_t *length);
  uint8_t  *(*GetDeviceQualifierDescriptor)(uint16_t *length);
#if (USBD_SUPPORT_USER_STRING == 1)
  uint8_t  *(*GetUsrStrDescriptor)(struct _USBD_HandleTypeDef *pdev ,uint8_t index,  uint16_t
#endif

} USBD_ClassTypeDef;
```

Connection between descriptors and handle

- In usb_device.c is created connection between handle and descriptors from usbd_desc over function:

```
USBD_Init(&hUsbDeviceFS, &FS_Desc, DEVICE_FS);
```




```c
/* USB Device descriptors structure */
typedef struct
{
  uint8_t  *(*GetDeviceDescriptor)( USBD_SpeedTypeDef speed , uint16_t *length);
  uint8_t  *(*GetLangIDStrDescriptor)( USBD_SpeedTypeDef speed , uint16_t *length);
  uint8_t  *(*GetManufacturerStrDescriptor)( USBD_SpeedTypeDef speed , uint16_t *length);
  uint8_t  *(*GetProductStrDescriptor)( USBD_SpeedTypeDef speed , uint16_t *length);
  uint8_t  *(*GetSerialStrDescriptor)( USBD_SpeedTypeDef speed , uint16_t *length);
  uint8_t  *(*GetConfigurationStrDescriptor)( USBD_SpeedTypeDef speed , uint16_t *length);
  uint8_t  *(*GetInterfaceStrDescriptor)( USBD_SpeedTypeDef speed , uint16_t *length);
#if (USBD_LPM_ENABLED == 1)
  uint8_t  *(*GetBOSDescriptor)( USBD_SpeedTypeDef speed , uint16_t *length);
#endif
} USBD_DescriptorsTypeDef;
```

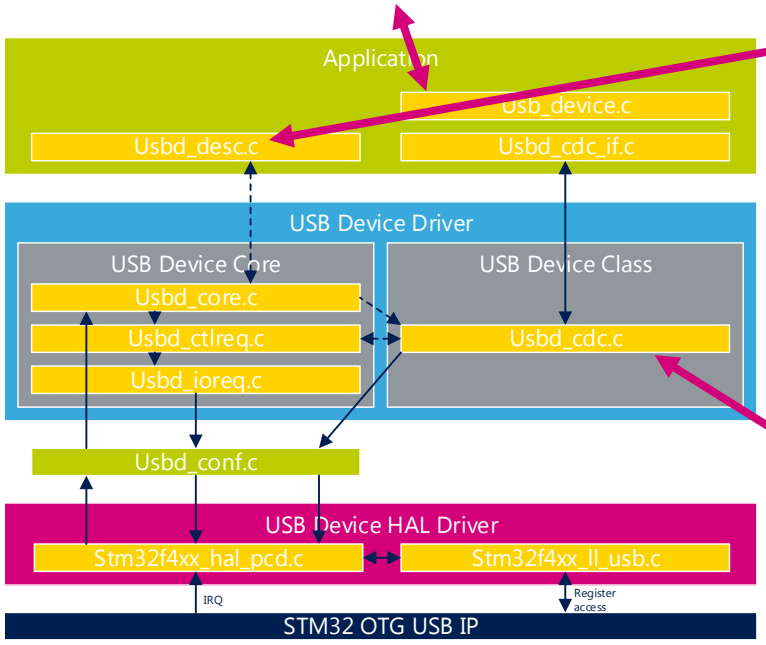
```c
/* USB Device handle structure */
typedef struct _USBD_HandleTypeDef
{
  uint8_t                 id;
  uint32_t                dev_config;
  uint32_t                dev_default_config;
  uint32_t                dev_config_status;
  USBD_SpeedTypeDef       dev_speed;
  USBD_EndpointTypeDef    ep_in[15];
  USBD_EndpointTypeDef    ep_out[15];
  uint32_t                ep0_state;
  uint32_t                ep0_data_len;
  uint8_t                 dev_state;
  uint8_t                 dev_old_state;
  uint8_t                 dev_address;
  uint8_t                 dev_connection_status;
  uint8_t                 dev_test_mode;
  uint32_t                dev_remote_wakeup;

  USBD_SetupReqTypedef    request;
  USBD_DescriptorsTypeDef *pDesc;
  USBD_ClassTypeDef       *pClass;
  void                    *pClassData;
  void                    *pUserData;
  void                    *pData;
} USBD_HandleTypeDef;
```

life.augmented

Connection between handle class

- In usb_device.c is created connection between handle and class functions from usbd_cdc.c(or different class) over function:

```
USBD_RegisterClass(&hUsbDeviceFS, &USBD_CDC);
```




```
typedef struct _Device_cb
{
  uint8_t  (*Init)              (struct _USBD_HandleTyp
  uint8_t  (*DeInit)            (struct _USBD_HandleTyp
  /* Control Endpoints*/
  uint8_t  (*Setup)             (struct _USBD_HandleTyp
  uint8_t  (*EP0_TxSent)        (struct _USBD_HandleTyp
  uint8_t  (*EP0_RxReady)       (struct _USBD_HandleTyp
  /* Class Specific Endpoints*/
  uint8_t  (*DataIn)            (struct _USBD_HandleTyp
  uint8_t  (*DataOut)           (struct _USBD_HandleTyp
  uint8_t  (*SOF)               (struct _USBD_HandleTyp
  uint8_t  (*IsoINIncomplete)   (struct _USBD_HandleTyp
  uint8_t  (*IsoOUTIncomplete)  (struct _USBD_HandleTyp

  uint8_t  *(*GetHSConfigDescriptor)(uint16_t *length)
  uint8_t  *(*GetFSConfigDescriptor)(uint16_t *length)
  uint8_t  *(*GetOtherSpeedConfigDescriptor)(uint16_t
  uint8_t  *(*GetDeviceQualifierDescriptor)(uint16_t *
#if (USBD_SUPPORT_USER_STRING == 1)
  uint8_t  *(*GetUsrStrDescriptor)(struct _USBD_HandleTypeDef *pdev ,uint8_t index,  uint16_t *length);
#endif
```

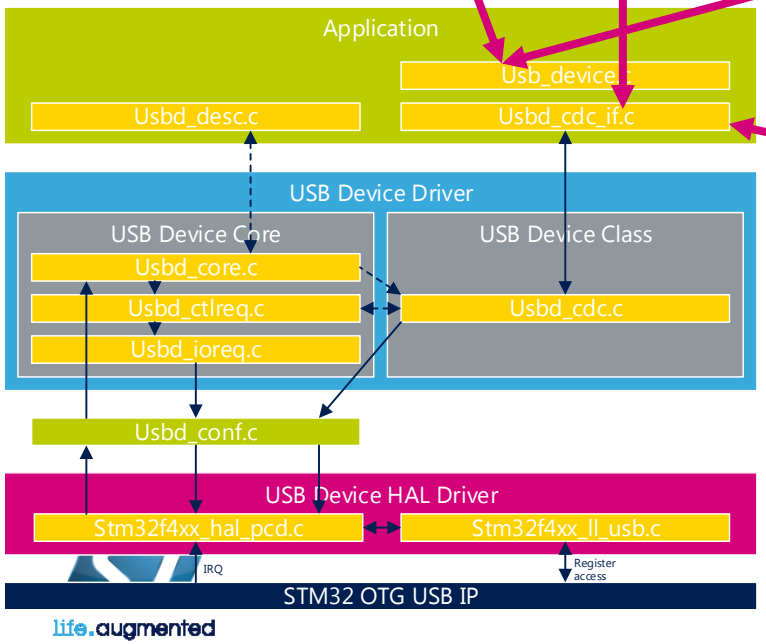
```
/* USB Device handle structure */
typedef struct _USBD_HandleTypeDef
{
  uint8_t                 id;
  uint32_t                dev_config;
  uint32_t                dev_default_config;
  uint32_t                dev_config_status;
  USBD_SpeedTypeDef       dev_speed;
  USBD_EndpointTypeDef    ep_in[15];
  USBD_EndpointTypeDef    ep_out[15];
  uint32_t                ep0_state;
  uint32_t                ep0_data_len;
  uint8_t                 dev_state;
  uint8_t                 dev_old_state;
  uint8_t                 dev_address;
  uint8_t                 dev_connection_status;
  uint8_t                 dev_test_mode;
  uint32_t                dev_remote_wakeup;

  USBD_SetupReqTypedef    request;
  USBD_DescriptorsTypeDef *pDesc;
  USBD_ClassTypeDef       *pClass;
  void                    *pClassData;
  void                    *pUserData;
  void                    *pData;
} USBD_HandleTypeDef;
```

life.augmented

Connection between class and their api interface (cdc)

- In usb_device.c is created connection between class and class api functions from usbd_cdc_if.c(or different class) over function:

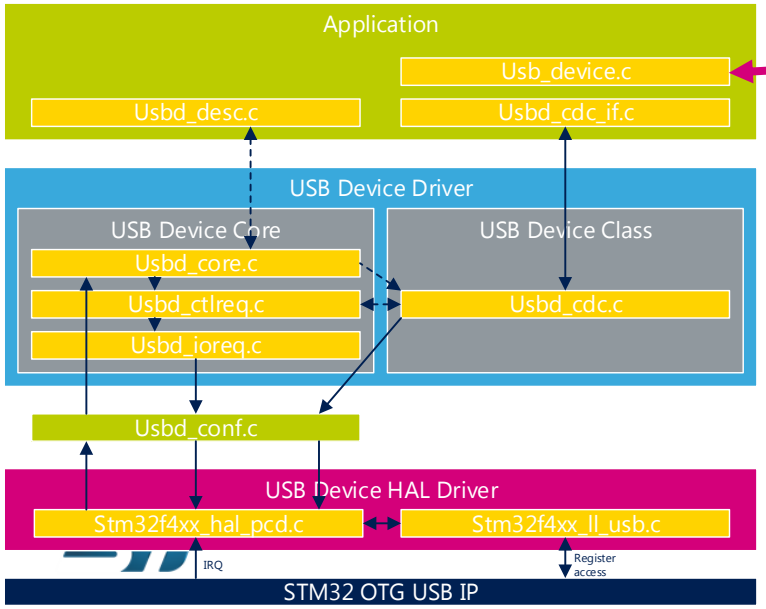
```
USBD_CDC_RegisterInterface(&hUsbDeviceFS,
&USBD_Interface_fops_FS);
```




```
USBD_CDC_ItfTypeDef USBD_Interface_fops_FS =
{
  CDC_Init_FS,
  CDC_DeInit_FS,
  CDC_Control_FS,
  CDC_Receive_FS
};
```

Start the USB device library

- The library function is started by function USBD_Start with handle as parameter.
- Function USBD_stop will stop the USB.
- USBD_Start/Stop is defined in usbd_core.c file




```
USBD_Start(&hUsbDeviceFS);
```

Device parameters


```
USBD_StatusTypeDef  USBD_LL_Init (USBD_HandleTypeDef *pdev)
{
  /* Init USB_IP */
  if (pdev->id == DEVICE_FS) {
  /* Link The driver to the stack */
  hpcd_USB_OTG_FS.pData = pdev;
  pdev->pData = &hpcd_USB_OTG_FS;

  hpcd_USB_OTG_FS.Instance = USB_OTG_FS;
  hpcd_USB_OTG_FS.Init.dev_endpoints = 6;
  hpcd_USB_OTG_FS.Init.speed = PCD_SPEED_FULL;
  hpcd_USB_OTG_FS.Init.dma_enable = DISABLE;
  hpcd_USB_OTG_FS.Init.ep0_mps = DEP0CTL_MPS_64;
  hpcd_USB_OTG_FS.Init.phy_itface = PCD_PHY_EMBEDDE
  hpcd_USB_OTG_FS.Init.Sof_enable = DISABLE;
  hpcd_USB_OTG_FS.Init.low_power_enable = DISABLE;
  hpcd_USB_OTG_FS.Init.lpm_enable = DISABLE;
  hpcd_USB_OTG_FS.Init.vbus_sensing_enable = DISABLE;
  hpcd_USB_OTG_FS.Init.use_dedicated_ep1 = DISABLE;
  if (HAL_PCD_Init(&hpcd_USB_OTG_FS) != HAL_OK)
  {
    _Error_Handler(__FILE__, __LINE__);
  }

  HAL_PCDEx_SetRxFiFo(&hpcd_USB_OTG_FS, 0x80);
  HAL_PCDEx_SetTxFiFo(&hpcd_USB_OTG_FS, 0, 0x40);
  HAL_PCDEx_SetTxFiFo(&hpcd_USB_OTG_FS, 1, 0x80);
  }
  return USBD_OK;
}
```

Created by CubeMX in usbd_conf.c

IP selection HS IP or FS IP

Number of used endpoints.

Speed in device mode LS/FS/HS

DMA used to endpoint TX/RX

EP0 max packet size

Device parameters


```
USBD_StatusTypeDef  USBD_LL_Init (USBD_HandleTypeDef *pdev)
{
  /* Init USB_IP */
  if (pdev->id == DEVICE_FS) {
  /* Link The driver to the stack */
  hpcd_USB_OTG_FS.pData = pdev;
  pdev->pData = &hpcd_USB_OTG_FS;

  hpcd_USB_OTG_FS.Instance = USB_OTG_FS;
  hpcd_USB_OTG_FS.Init.dev_endpoints = 6;
  hpcd_USB_OTG_FS.Init.speed = PCD_SPEED_FULL;
  hpcd_USB_OTG_FS.Init.dma_enable = DISABLE;
  hpcd_USB_OTG_FS.Init.ep0_mps = DEP0CTL_MPS_64;
  hpcd_USB_OTG_FS.Init.phy_itface = PCD_PHY_EMBEDDED;
  hpcd_USB_OTG_FS.Init.Sof_enable = DISABLE;
  hpcd_USB_OTG_FS.Init.low_power_enable = DISABLE;
  hpcd_USB_OTG_FS.Init.lpm_enable = DISABLE;
  hpcd_USB_OTG_FS.Init.vbus_sensing_enable = DISABLE;
  hpcd_USB_OTG_FS.Init.use_dedicated_ep1 = DISABLE;
  if (HAL_PCD_Init(&hpcd_USB_OTG_FS) != HAL_OK)
  {
    _Error_Handler(__FILE__, __LINE__);
  }

  HAL_PCDEx_SetRxFiFo(&hpcd_USB_OTG_FS, 0x80);
  HAL_PCDEx_SetTxFiFo(&hpcd_USB_OTG_FS, 0, 0x40);
  HAL_PCDEx_SetTxFiFo(&hpcd_USB_OTG_FS, 1, 0x80);
  }
  return USBD_OK;
}
```

SOF interrupt generation (each 1ms/125us)

VBUS sensing feature used

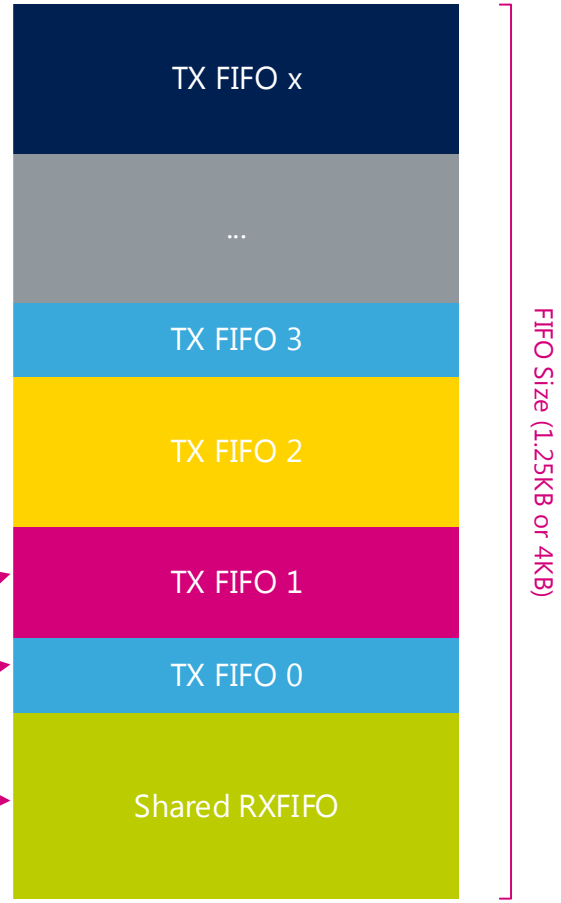
Used specific EP1 interrupts (HS only)

Device parameters – FIFO allocation


```
USBD_StatusTypeDef  USBD_LL_Init (USBD_HandleTypeDef *pdev)
{
  /* Init USB_IP */
  if (pdev->id == DEVICE_FS) {
  /* Link The driver to the stack */
  hpcd_USB_OTG_FS.pData = pdev;
  pdev->pData = &hpcd_USB_OTG_FS;

  {
    _Error_Handler(__FILE__, __LI
  }

  HAL_PCDEx_SetRxFiFo(&hpcd_USB_OTG_FS, 0x80);
  HAL_PCDEx_SetTxFiFo(&hpcd_USB_OTG_FS, 0, 0x40);
  HAL_PCDEx_SetTxFiFo(&hpcd_USB_OTG_FS, 1, 0x80);
  }
  return USBD_OK;
}
```

Bigger value will provide bigger transfer speed and less interrupts when big amount of data is sent/received

The sum of all fifo sizes must be 1.25KB or 4KB
Argument in words – 32 bits!!

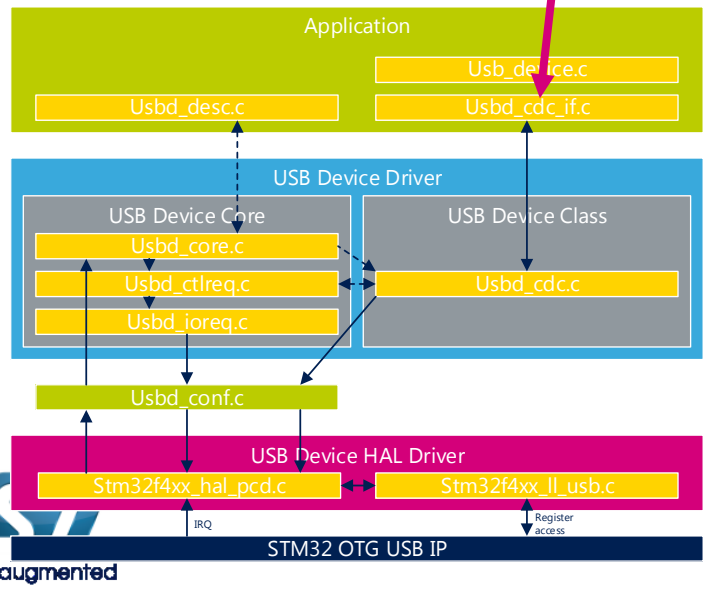


EP number

Bulk Tx


```
uint8_t CDC_Transmit_FS(uint8_t* Buf, uint16_t Len)
```

Function can be called from main application



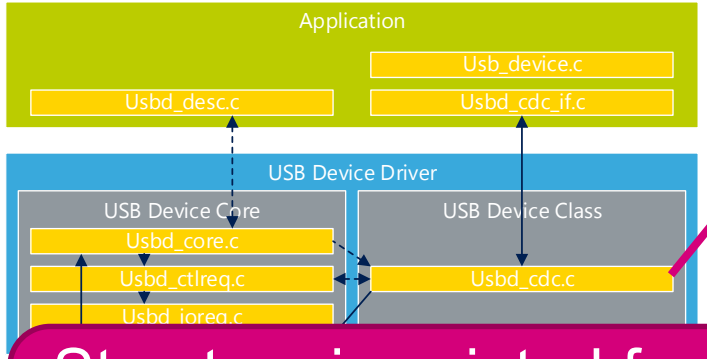
Bulk Tx


```
uint8_t CDC_Transmit_FS(uint8_t* Buf, uint16_t Len)
```

First the CDC structure is used to store buffer source for TX

And data size

```
USBD_CDC_SetTxBuffer(&hUsbDeviceFS, Buf, Len);
```




```
typedef struct
{
  uint32_t data[CDC_DATA_HS_MAX_PACKET_SIZE/4];      /* Force 32bits alignment */
  uint8_t  CmdOpCode;
  uint8_t  CmdLength;
  uint8_t  *RxBuffer;
  uint8_t  *TxBuffer;
  uint32_t RxLength;
  uint32_t TxLength;

  __IO uint32_t TxState;
  __IO uint32_t RxState;
}
USBD_CDC_HandleTypeDef;
```

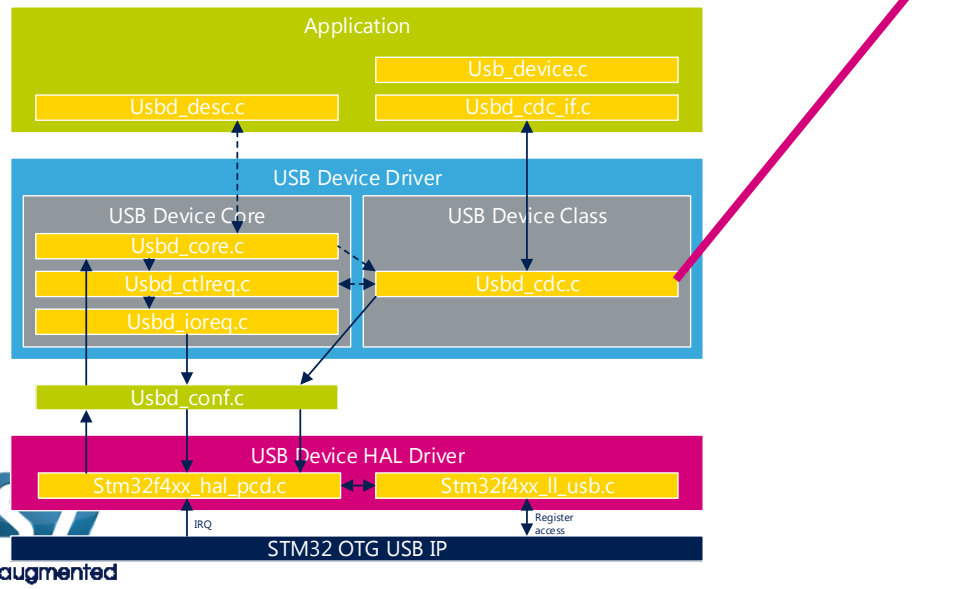
Structure is pointed from main USB handle it is dynamically allocated (from HEAP) during enumeration

Bulk Tx


```
uint8_t CDC_Transmit_FS(uint8_t* Buf, uint16_t Len)
```

Transmit function is called to pass our data to EP

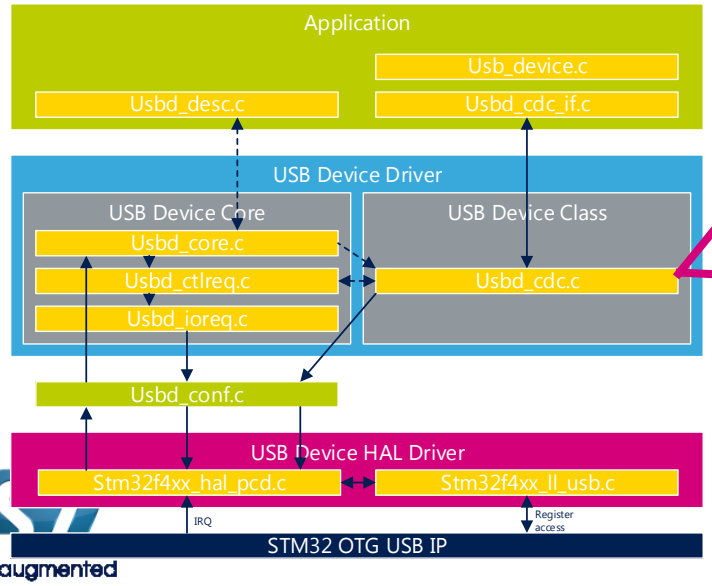
```
uint8_t  USBD_CDC_TransmitPacket(USBD_HandleTypeDef *pdev)
```



Bulk Tx


```
uint8_t CDC_Transmit_FS(uint8_t* Buf, uint16_t Len)
```

```
uint8_t  USBD_CDC_TransmitPacket(USBD_HandleTypeDef *pdev)
```




```
typedef struct
{
  uint32_t data[CDC_DATA_HS_MAX_PACKET_SIZE/4];      /* Force 32bits alignment */
  uint8_t  CmdOpCode;
  uint8_t  CmdLength;
  uint8_t  *RxBuffer;
  uint8_t  *TxBuffer;
  uint32_t RxLength;
  uint32_t TxLength;

  __IO uint32_t TxState;
  __IO uint32_t RxState;
}
USBD_CDC_HandleTypeDef;
```

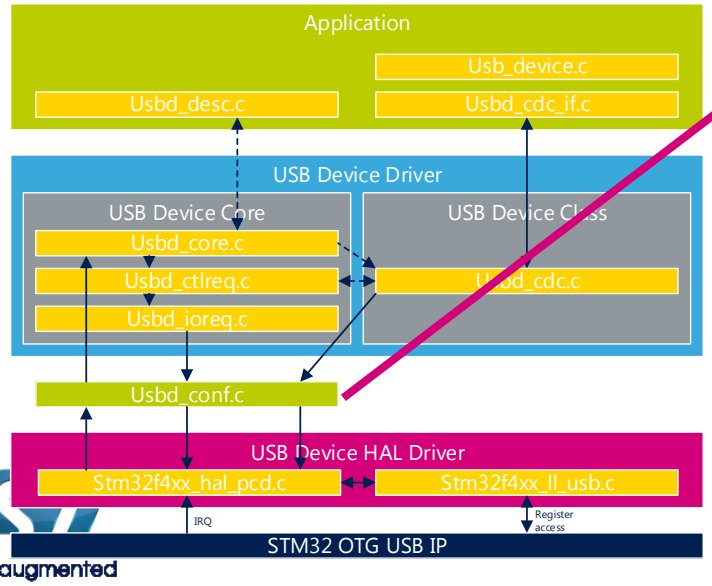
TxState is set to 1 marking outgoing TX, !buffer now cannot be changed by Application!

life.augmented

Bulk Tx


```
uint8_t CDC_Transmit_FS(uint8_t* Buf, uint16_t Len)
```

```
USBD_StatusTypeDef  USBD_LL_Transmit (USBD_HandleTypeDef *pdev,
                                      uint8_t  ep_addr,
                                      uint8_t  *pbuf,
                                      uint16_t  size)
```

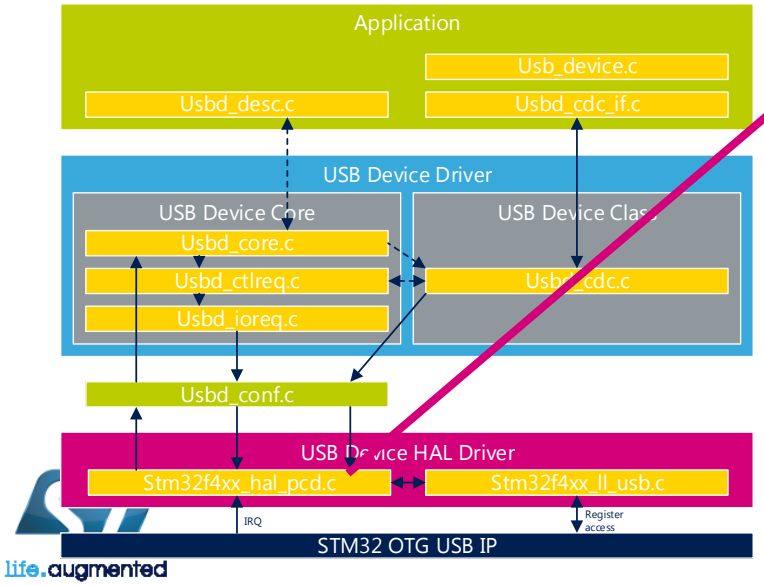


USBD_LL_Transmit is called to send data

Bulk Tx


```
uint8_t CDC_Transmit_FS(uint8_t* Buf, uint16_t Len)
```

```
HAL_StatusTypeDef HAL_PCD_EP_Transmit(PCD_HandleTypeDef *hpcd,
uint8_t ep_addr, uint8_t *pBuf, uint32_t len)
```

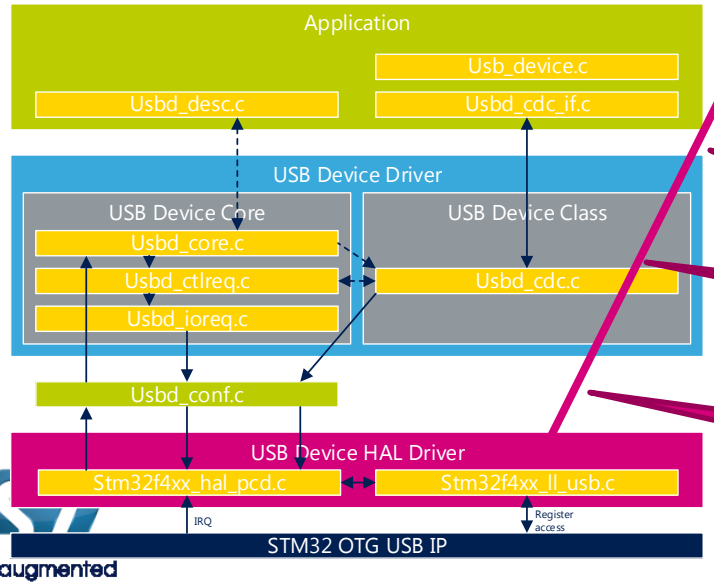


This calls HAL_PDC_EP_Transmit

Bulk Tx


```
uint8_t CDC_Transmit_FS(uint8_t* Buf, uint16_t Len)
```

```
HAL_StatusTypeDef USB_EPStartXfer(USB_OTG_GlobalTypeDef *USBx
, USB_OTG_EPTypeDef *ep, uint8_t dma)
```



At the end is called USB_EPStartXfer

This function set EP registers, how much bytes will be sent, how many packets…

Then EP Fifo empty interrupt is enabled

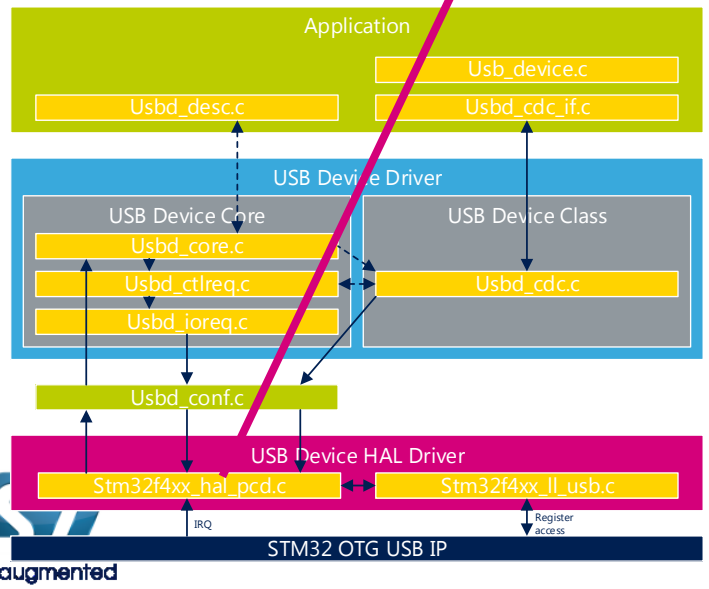
Now EP is enabled

Bulk Tx


```
uint8_t CDC_Transmit_FS(uint8_t* Buf, uint16_t Len)
```

Empty fifo interrupt is handled by HAL_PCD_IRQHandler

```
void HAL_PCD_IRQHandler(PCD_HandleTypeDef *hpcd)
```

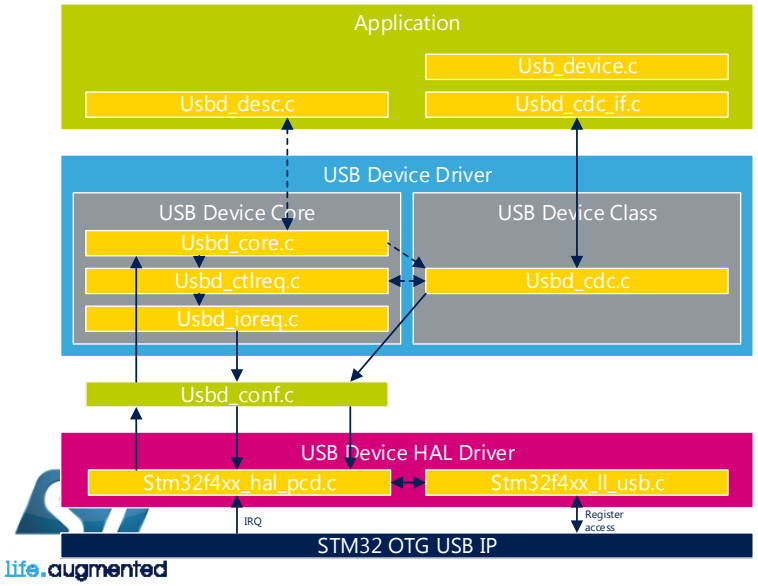


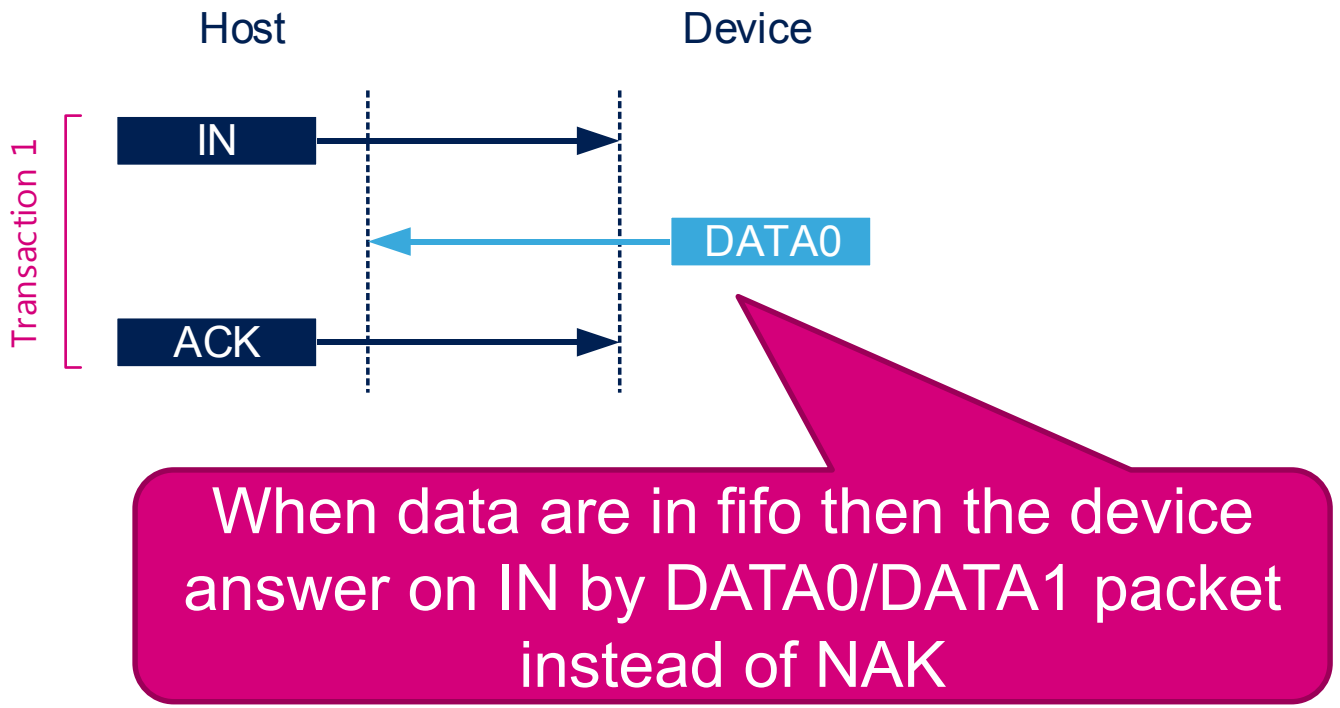
This function calls PCD_WriteEmptyTxFifo which transfers data from buffer into EP FIFO

```
static HAL_StatusTypeDef
PCD_WriteEmptyTxFifo(PCD_HandleTyp
eDef *hpcd, uint32_t epnum)
```

Bulk Tx


```
uint8_t CDC_Transmit_FS(uint8_t* Buf, uint16_t Len)
```



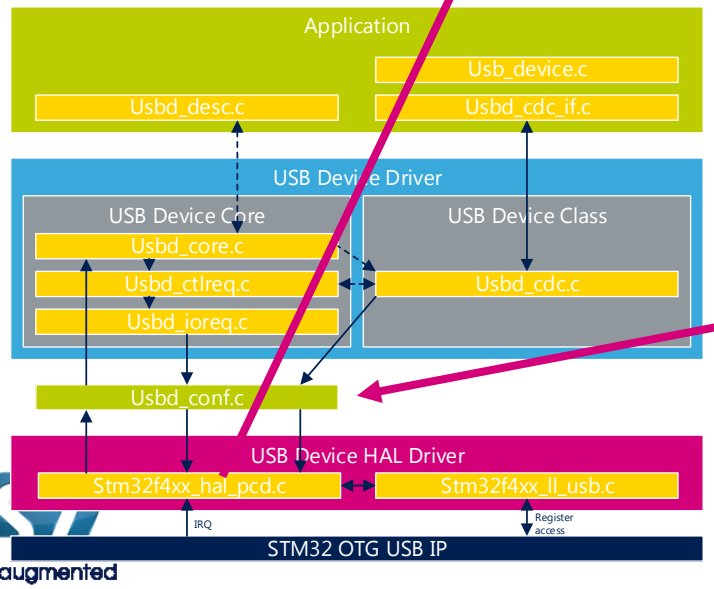


Bulk Tx


```
uint8_t CDC_Transmit_FS(uint8_t* Buf, uint16_t Len)
```

When all bytes ar transferd(previously set in register) Txcomplete interrupt is trigger

```
void HAL_PCD_IRQHandler(PCD_HandleTypeDef *hpcd)
```



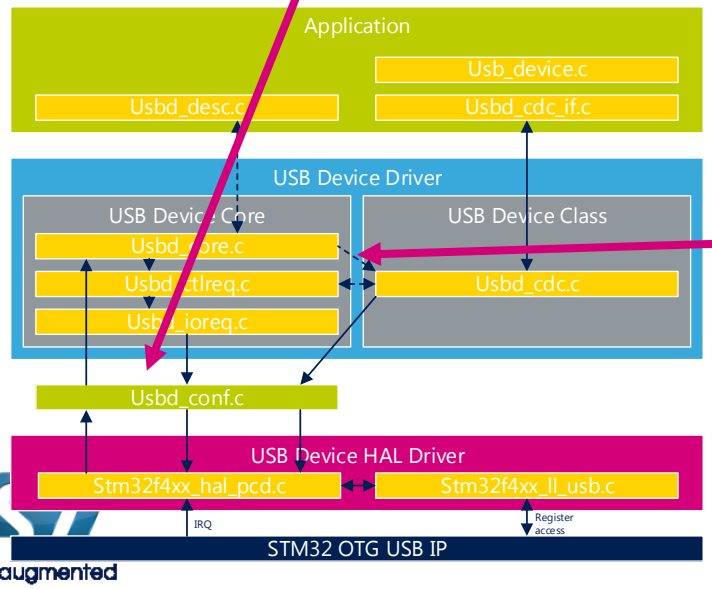
HAL_PCD_DataInStageCallback is called

```
void
HAL_PCD_DataInStageCallback(PCD_Han
dleTypeDef *hpcd, uint8_t epnum)
```

Bulk Tx


```
uint8_t CDC_Transmit_FS(uint8_t* Buf, uint16_t Len)
```

```
void HAL_PCD_DataInStageCallback(PCD_HandleTypeDef *hpcd,
uint8_t epnum)
```



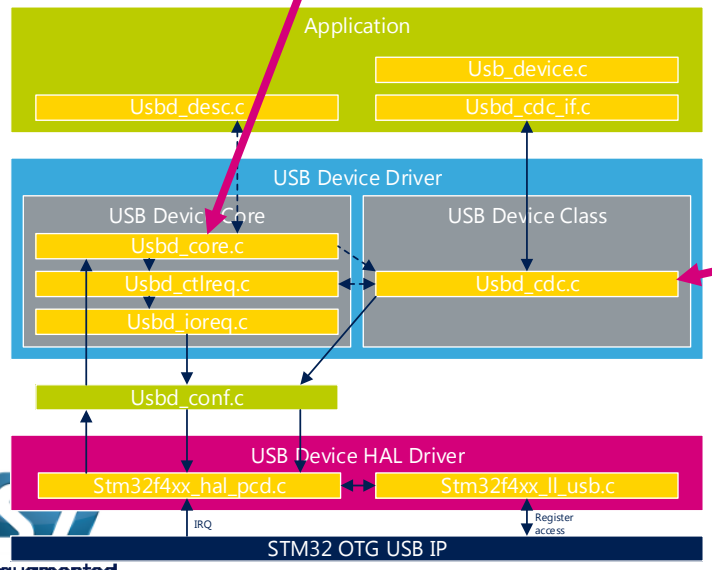
This calls USBD_LL_DataINStage

```
USBD_StatusTypeDef
USBD_LL_DataInStage(USBD_HandleType
Def *pdev ,uint8_t epnum, uint8_t
*pdata)
```

Bulk Tx


```
uint8_t CDC_Transmit_FS(uint8_t* Buf, uint16_t Len)
```

```
USBD_StatusTypeDef USBD_LL_DataInStage(USBD_HandleTypeDef *pdev
,uint8_t epnum, uint8_t *pdata)
```



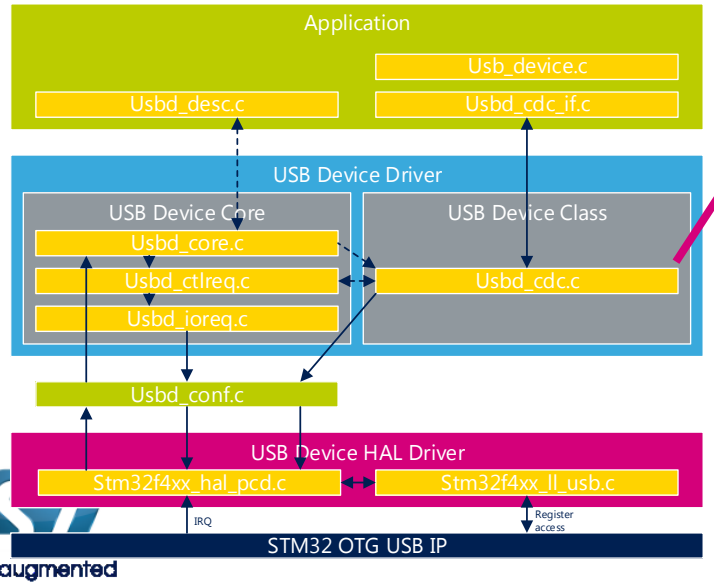

```
typedef struct _Device_cb
{
  uint8_t  (*Init)            (struct _USBD_HandleTypeDef *pdev , uint8_t cfgidx);
  uint8_t  (*DeInit)          (struct _USBD_HandleTypeDef *pdev , uint8_t cfgidx);
 /* Control Endpoints*/
  uint8_t  (*Setup)           (struct _USBD_HandleTypeDef *pdev , USBD_SetupReqTypedef  *req);
  uint8_t  (*EP0_TxSent)      (struct _USBD_HandleTypeDef *pdev );
  uint8_t  (*EP0_RxReady)     (struct _USBD_HandleTypeDef *pdev );
  /* Class Specific Endpoints*/
  uint8_t  (*DataIn)          (struct _USBD_HandleTypeDef *pdev , uint8_t epnum);
  uint8_t  (*DataOut)         (struct _USBD_HandleTypeDef *pdev , uint8_t epnum);
  uint8_t  (*SOF)             (struct _USBD_HandleTypeDef *pdev);
  uint8_t  (*IsoINIncomplete) (struct _USBD_HandleTypeDef *pdev , uint8_t epnum);
  uint8_t  (*IsoOUTIncomplete) (struct _USBD_HandleTypeDef *pdev , uint8_t epnum);
  uint8_t  *(*GetUsrStrDescriptor)(struct _USBD_HandleTypeDef *pdev ,uint8_t index,  uint16_t *length);
#endif

} USBD_ClassTypeDef;
```

This function calls out pointer in DataIN function defined our class structure

Bulk Tx


```
uint8_t CDC_Transmit_FS(uint8_t* Buf, uint16_t Len)
```

```
static uint8_t  USBD_CDC_DataIn (USBD_HandleTypeDef *pdev, uint8_t
epnum)
```



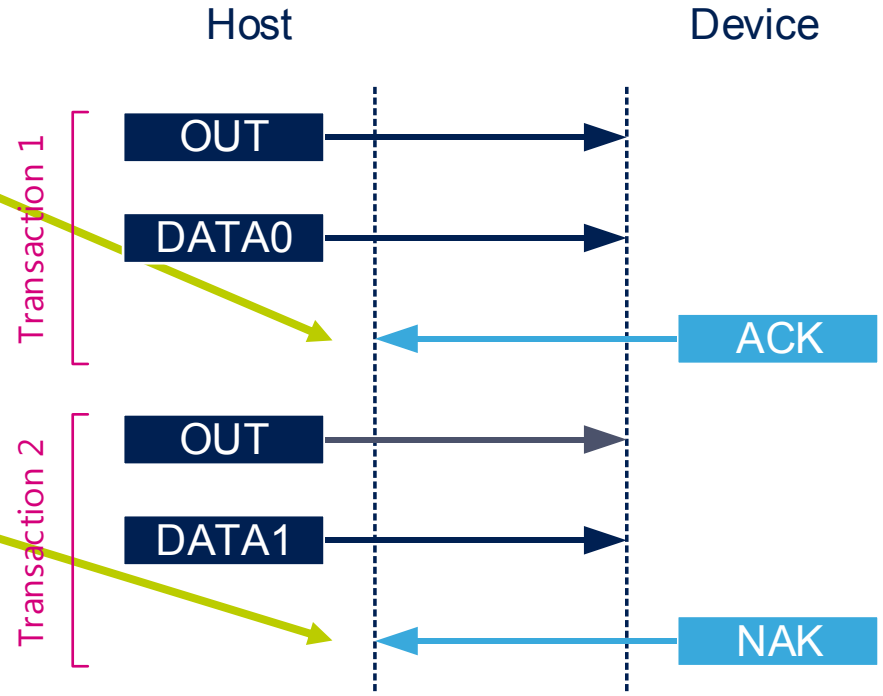
In our case it is called USBD_CDC_DataIn

Here the TxState is set to 0 which allow to use CDC_Transmit_FS function again.

In this moment we can use the buffer again

Bulk RX

- EP can receive only preconfigured amount od data or less
- Then is EP deactivated.
- If not activated it will not receive more data.
- They are NAKed
- Our libraries enable EP after enumeration
- But after successful receive it must be enabled by user code

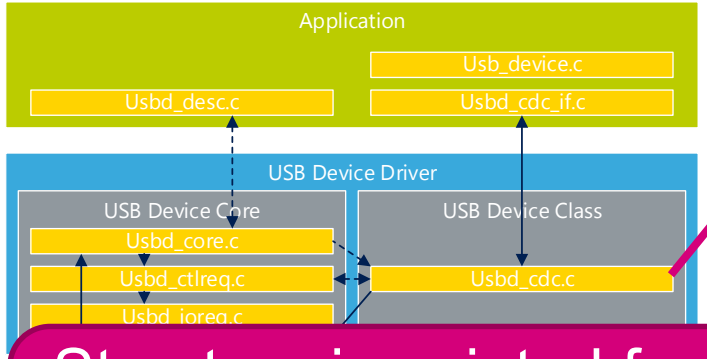


Bulk RX

Prepare buffer where we will store our data

First the CDC structure is used to store buffer source for TX

```
uint8_t USBD_CDC_SetRxBuffer (USBD_HandleTypeDef *pdev, uint8_t *pbuff)
```




```
typedef struct
{
  uint32_t data[CDC_DATA_HS_MAX_PACKET_SIZE/4];      /* Force 32bits alignment */
  uint8_t  CmdOpCode;
  uint8_t  CmdLength;
  uint8_t  *RxBuffer;
  uint8_t  *TxBuffer;
  uint32_t RxLength;
  uint32_t TxLength;

  __IO uint32_t TxState;
  __IO uint32_t RxState;
}
USBD_CDC_HandleTypeDef;
```

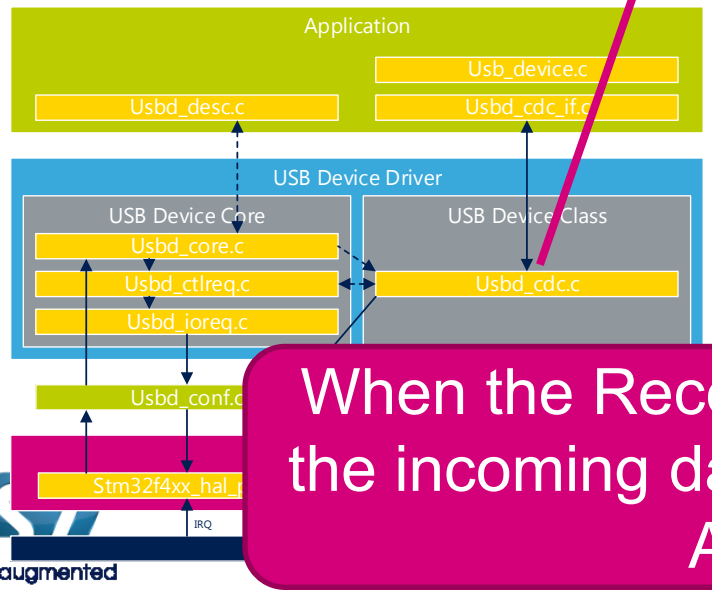
Here Rx size is defined by CDC max packet size(64B in FS)

Structure is pointed from main USB handle it is dynamically allocated (from HEAP) during enumeration

Bulk RX

Allow to start receiving packets.
Before all data are NAKed

```
uint8_t  USBD_CDC_ReceivePacket(USBD_HandleTypeDef *pdev)
```



When the Receive packet is called the incoming data on our EP will be ACKed

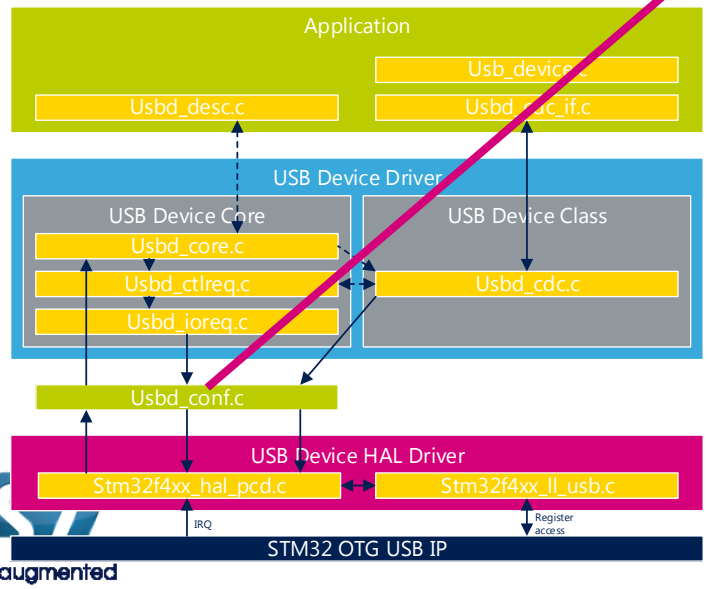
life.augmented

Bulk RX


```
uint8_t  USBD_CDC_ReceivePacket(USBD_HandleTypeDef *pdev)
```

Function will prepare the EP for receive

USBD_LL_PrepareReceive



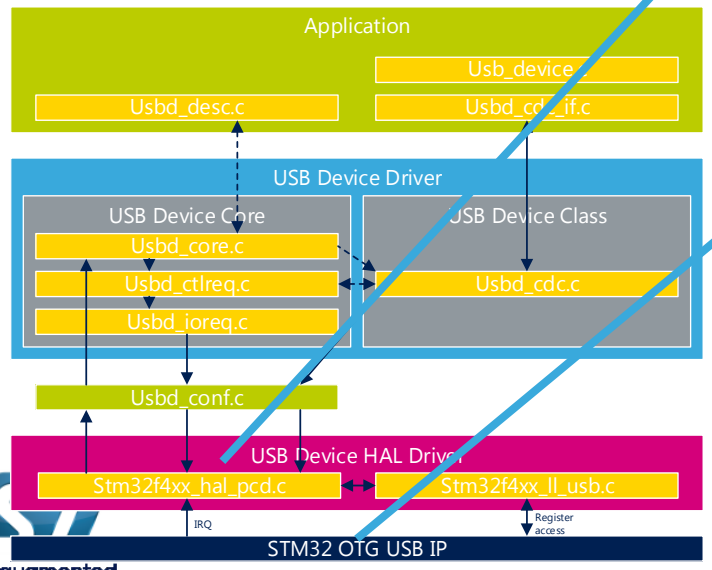
Bulk RX

`USBD_LL_PrepareReceive`

`HAL_PCD_EP_Receive`

`USB_EPStartXfer`

LL driver to configure registers on USB IP



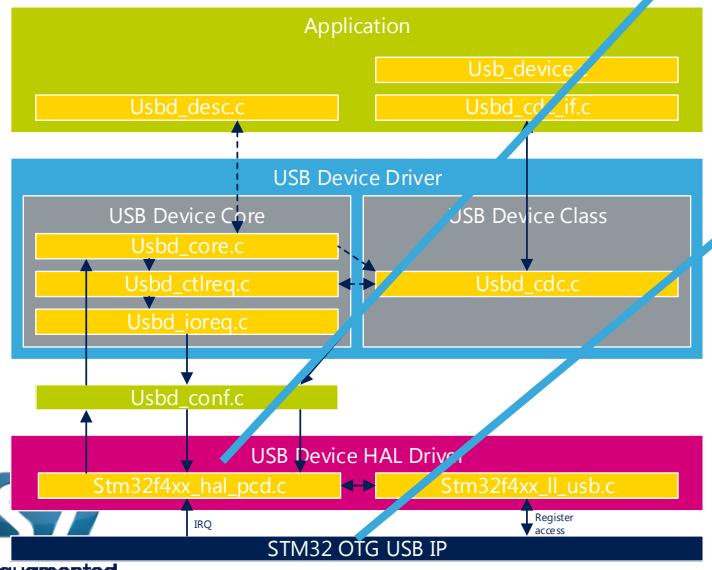
Bulk RX

`USBD_LL_PrepareReceive`

`HAL_PCD_EP_Receive`

`USB_EPStartXfer`

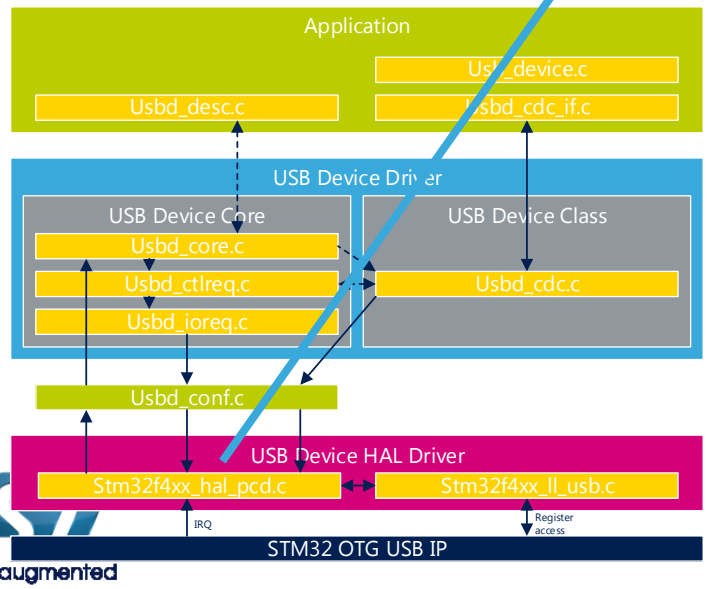
LL driver to configure registers on USB IP

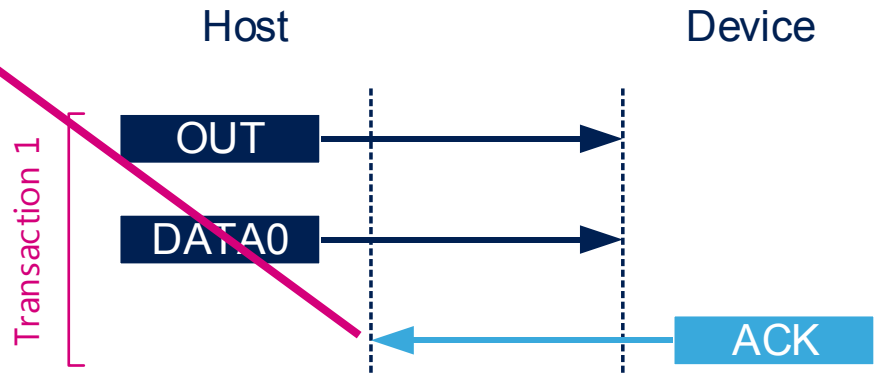


Bulk RX

When data are ACKed the USB IRQ is called and inside is called PCD handler to handle interrupt routine

```
HAL_PCD_IRQHandler(&hpcd_USB_OTG_FS);
```



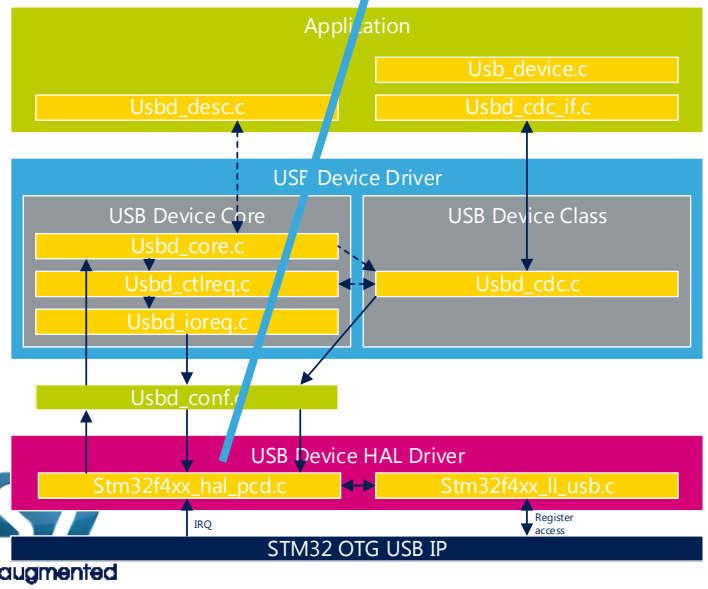


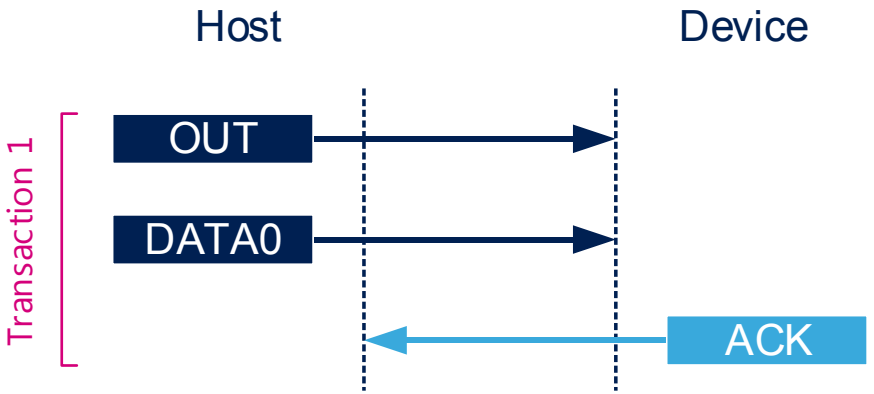
Bulk RX


```
HAL_PCD_IRQHandler(&hpcd_USB_OTG_FS);
```

```
USB_ReadPacket
```

First the packed is read from RX fifo



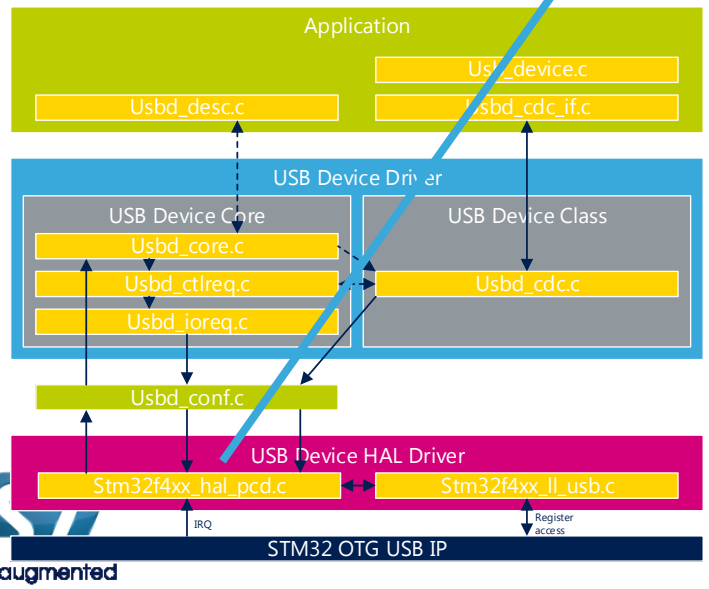


Bulk RX


```
HAL_PCD_IRQHandler(&hpcd_USB_OTG_FS);
```

```
HAL_PCD_DataInStageCallback(hpcd, epnum);
```

OUT packet transfe complete interrupt routines



Bulk RX

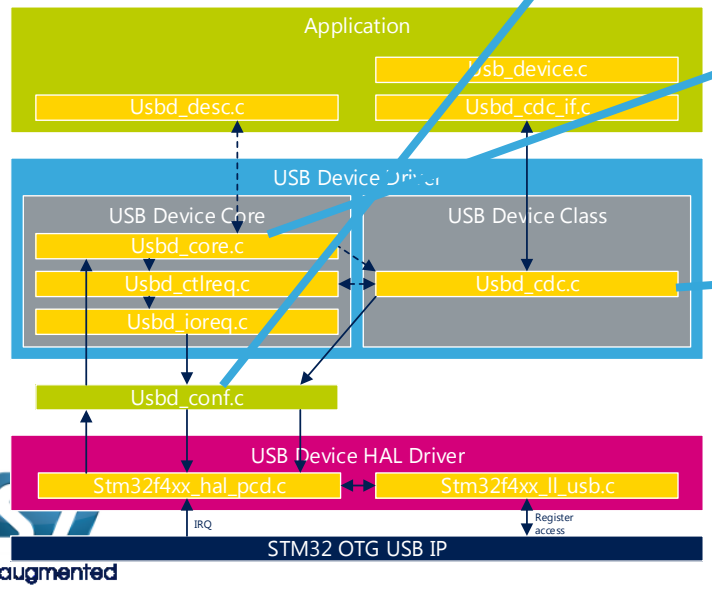
HAL_PCD_IRQHandler(&hpcd_USB_OTG_FS);

```
HAL_PCD_DataInStageCallback(hpcd, epnum);
```

USBD_LL_DataOutStage

```
pdev->pClass->DataOut(pdev, epnum);
```

Call class structure, in our case CDC

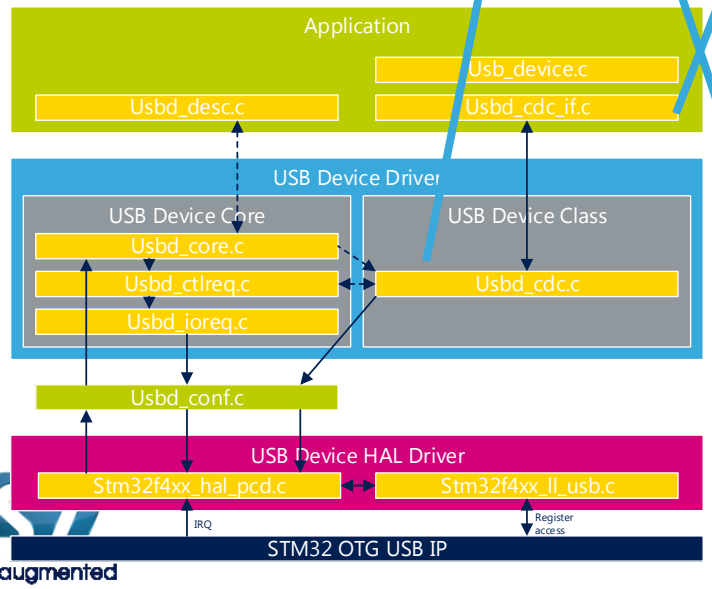


Bulk RX


```
((USBD_CDC_ItfTypeDef *)pdev->pUserData)->Receive(hcdc->RxBuffer, &hcdc->RxLength);
```

```
static int8_t CDC_Receive_FS (uint8_t* Buf, uint32_t *Len)
```

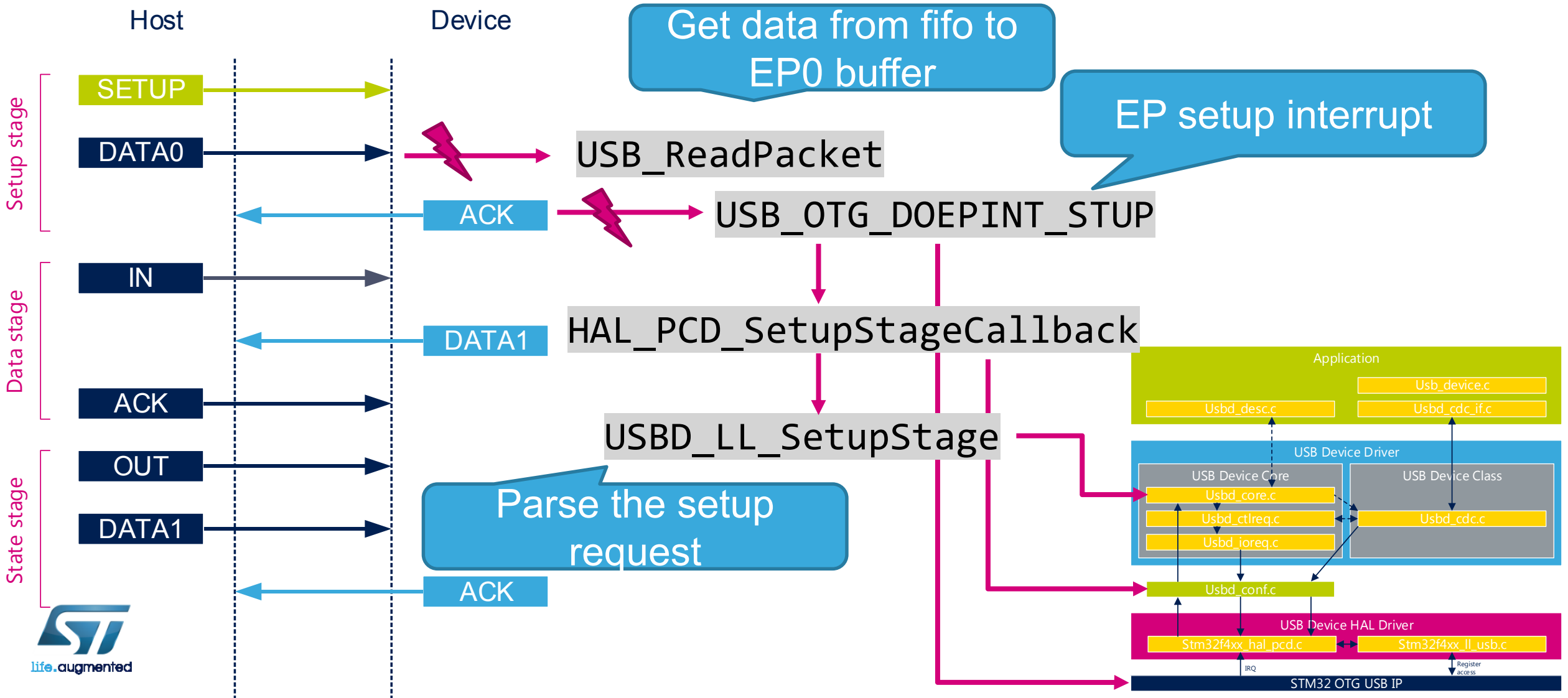
Receive IF function into our application




```
USBD_CDC_ItfTypeDef USBD_Interface_fops_FS =
{
  CDC_Init_FS,
  CDC_DeInit_FS,
  CDC_Control_FS,
  CDC_Receive_FS
};
```

USBD_CDC_ReceivePacket must be called again to be able receive more packets

Control packet handling

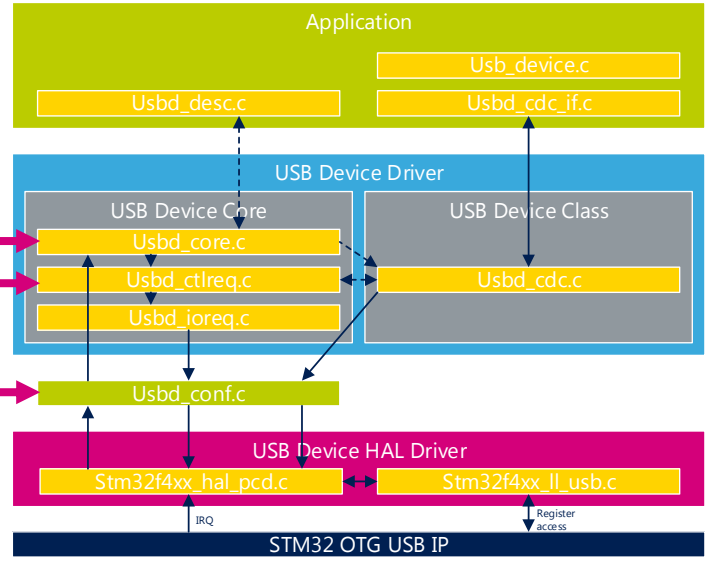


life.augmented

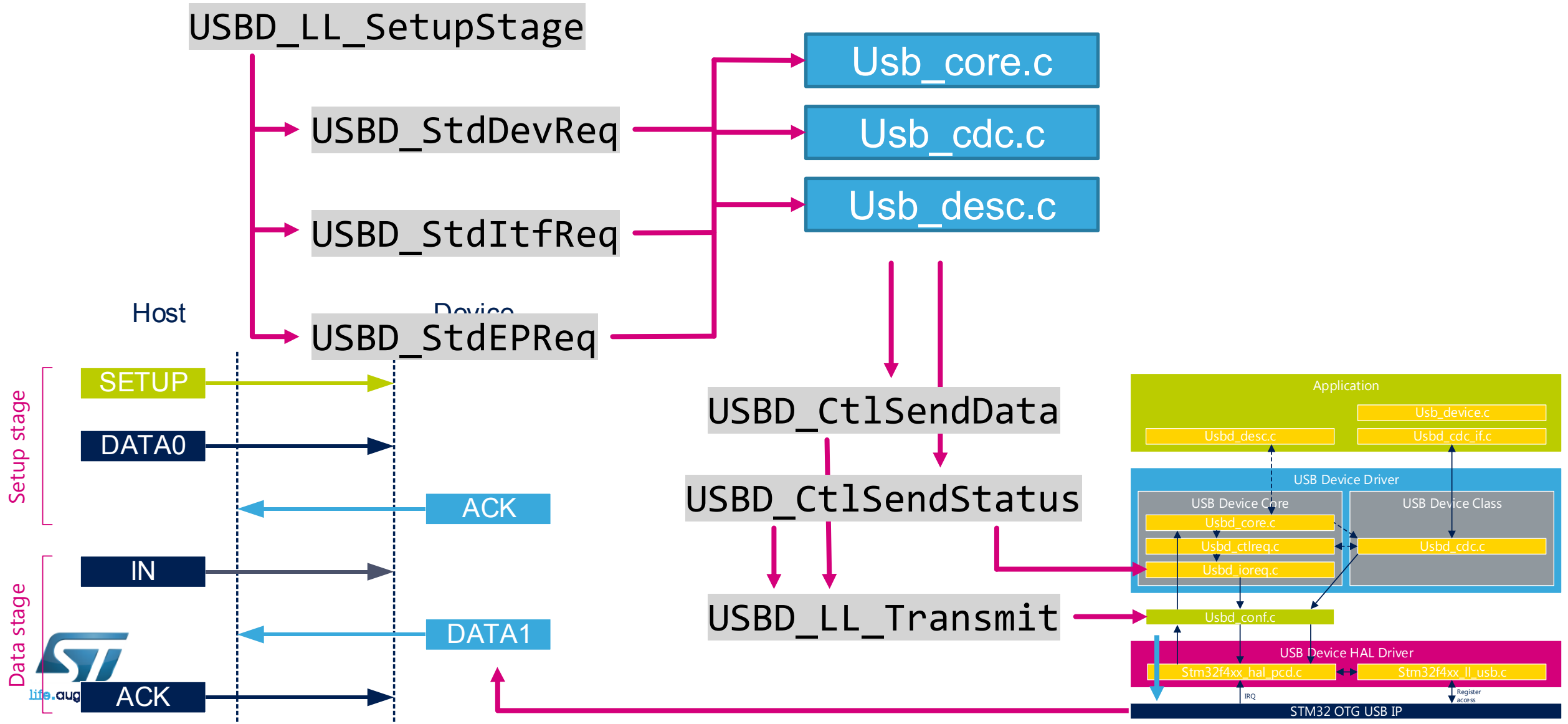
Control packet handling


```
USBD_LL_SetupStage
  → USBD_StdDevReq
  → USBD_StdItfReq
  → USBD_StdEPReq
  → USBD_LL_StallEP
```

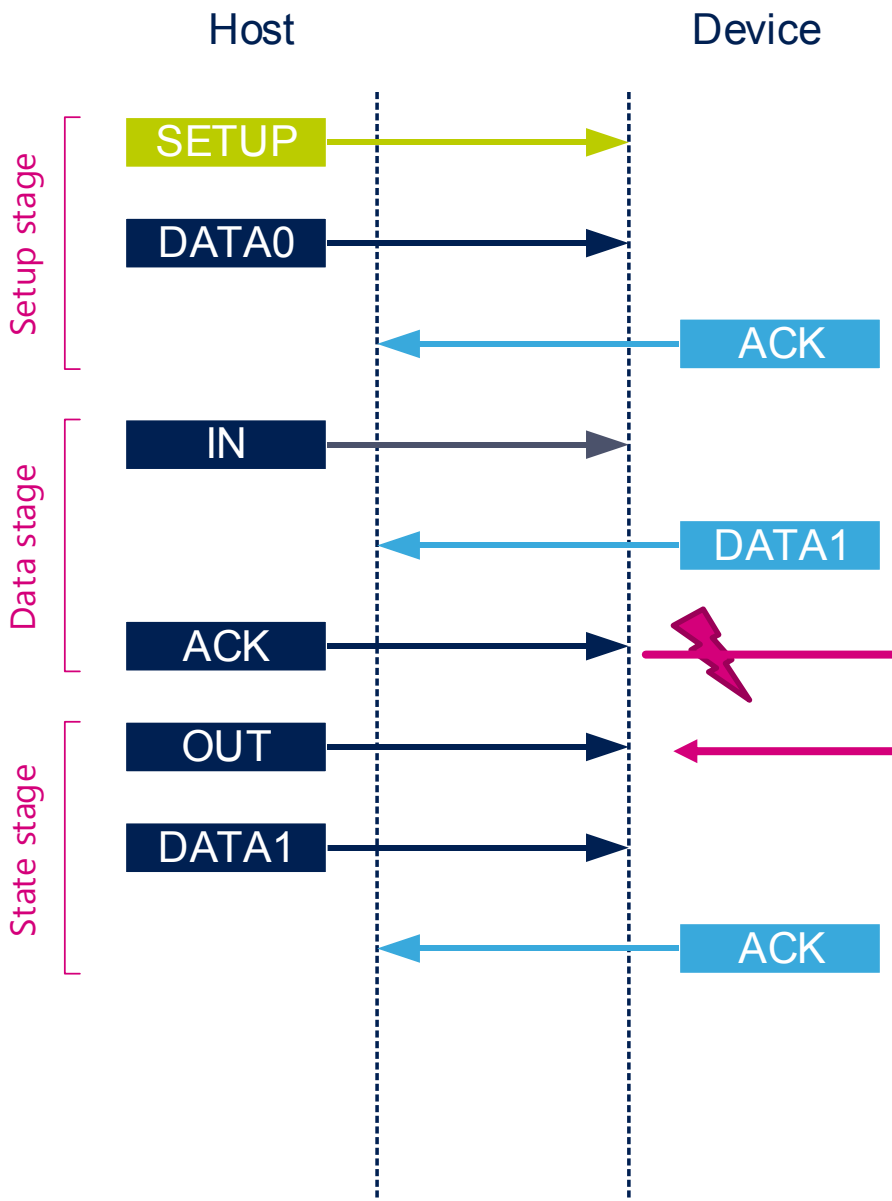
Setup request not recognized device answer with STALL



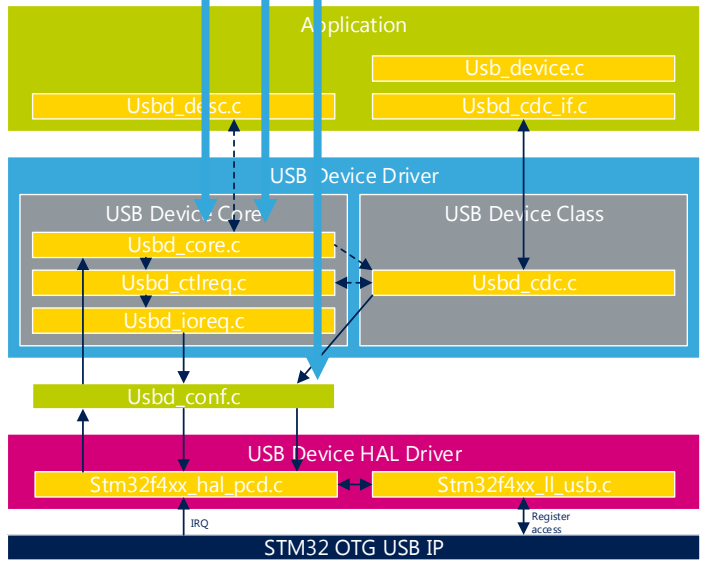
Control packet handling



Control packet handling

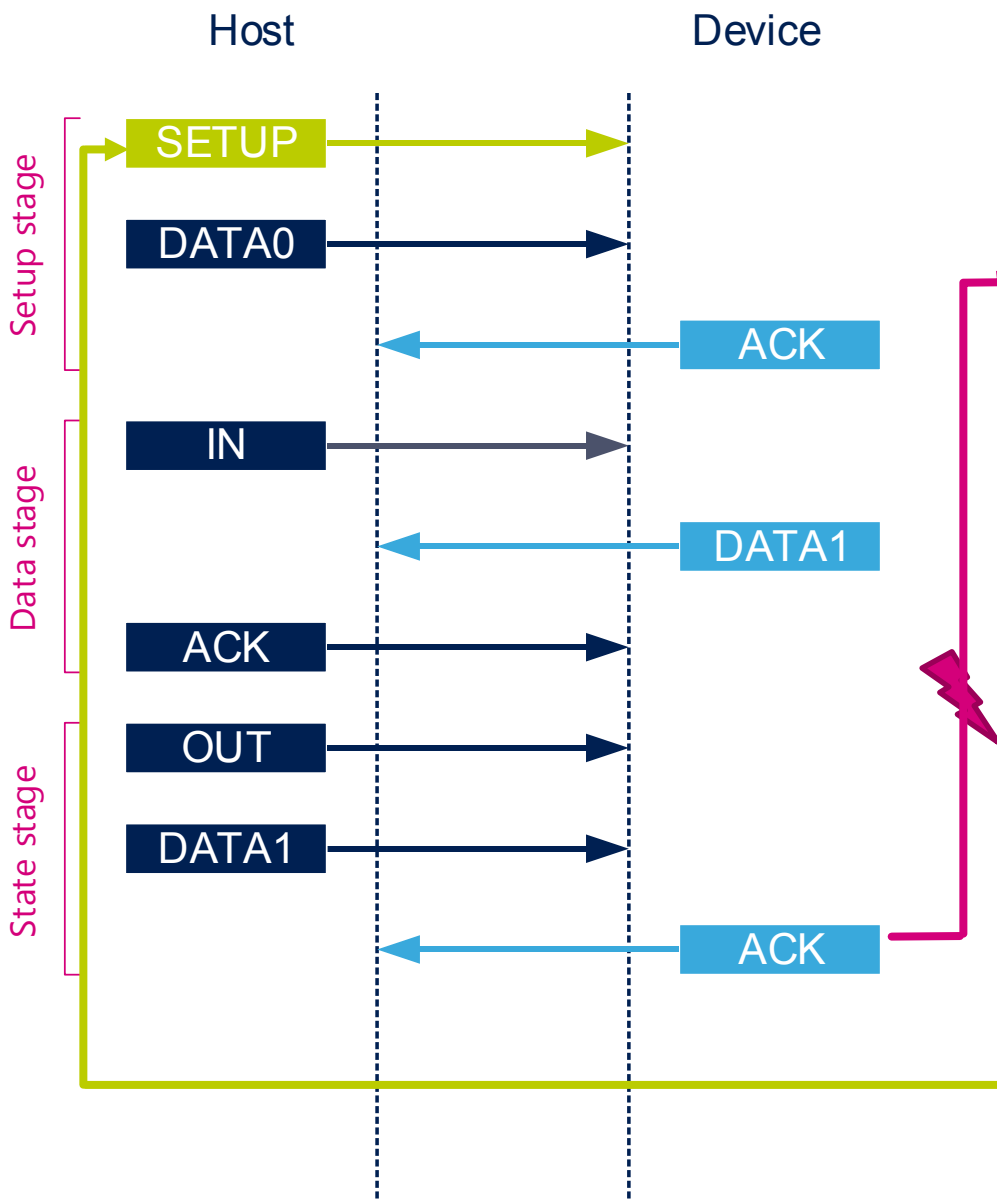



```
HAL_PCD_DataInStageCallback
        ↓
USBD_LL_DataInStage
        ↓
USBD_LL_PrepareReceive
```

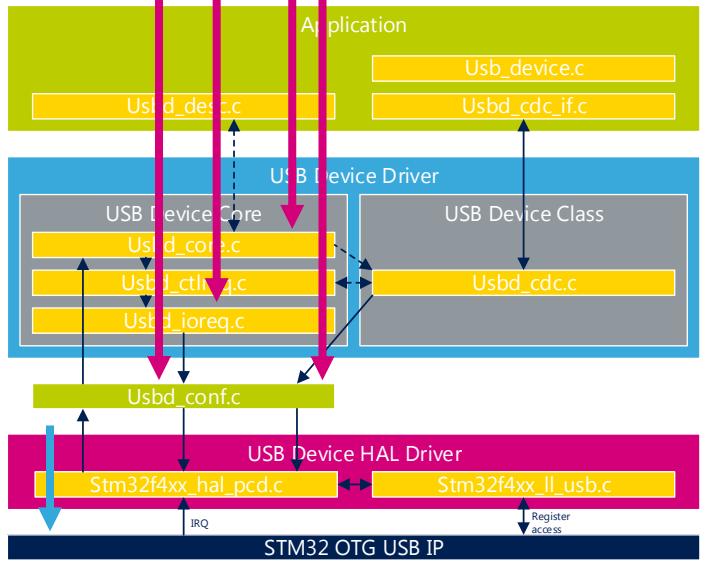


life.augmented

Control packet handling




```
HAL_PCD_DataOutStageCallback
        ↓
USBD_LL_DataOutStage
        ↓
USBD_CtlContinueRx
        ↓
USBD_LL_PrepareReceive
```



life.augmented

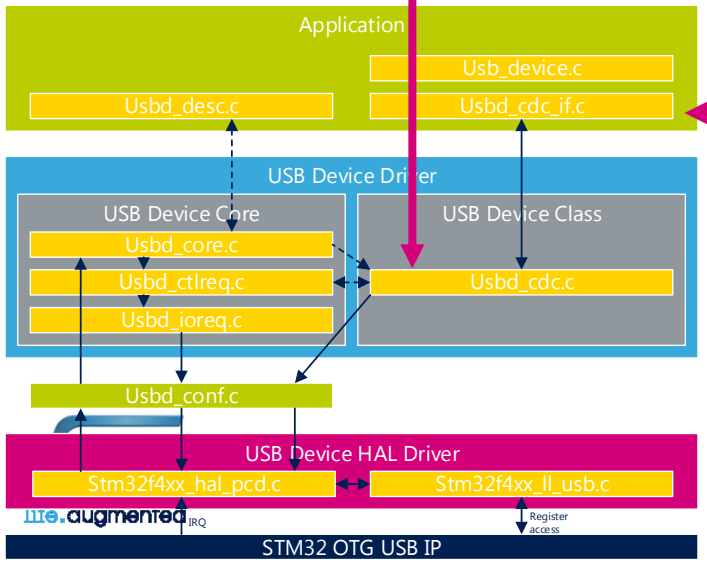
Device IF structures

This is name is used to call the pointer function

((USBD_CDC_ItfTypeDef *) pdev->pUserData)->Init();

```
typedef struct _USBD_C
{
  int8_t (* Init)
  int8_t (* DeInit)
  int8_t (* Control)
  int8_t (* Receive)
}USBD_CDC_ItfTypeDef;
```

```
USBD_CDC_ItfTypeDef
USBD_Interface_fops_FS =
{
  CDC_Init_FS,
  CDC_DeInit_FS,
  CDC_Control_FS,
  CDC_Receive_FS
};
```



Definition of function pointers which will be used in usbd_cdc.c

Real function implementation In usbd_cdc_if.c

Device Class structures

This is name is used to call the pointer function
pdev->pClass->Init(pdev, cfgidx)

```
typedef struct _Device_cb
{
  uint8_t  (*Init)            (struct _USBD
  uint8_t  (*DeInit)          (struct _USBD
 /* Control Endpoints*/
  uint8_t  (*Setup)           (struct _USBD
  uint8_t  (*EP0_TxSent)      (struct _USBD
  uint8_t  (*EP0_RxReady)     (struct _USBD
  /* Class Specific Endpoints*/
  uint8_t  (*DataIn)          (struct _USBD
  uint8_t  (*DataOut)         (struct _USBD
  uint8_t  (*SOF)             (struct _USBD
  uint8_t  (*IsoINIncomplete) (struct _USBD
  uint8_t  (*IsoOUTIncomplete)(struct _USBD

  uint8_t  *(*GetHSConfigDescriptor)(uint16_
  uint8_t  *(*GetFSConfigDescriptor)(uint16_
  uint8_t  *(*GetOtherSpeedConfigDescriptor)
  uint8_t  *(*GetDeviceQualifierDescriptor)(
#if (USBD_SUPPORT_USER_STRING == 1)
  uint8_t  *(*GetUsrStrDescriptor)(struct _USBD_HandleTypeDef *pdev ,uint8_t Index,  uint
#endif
} USBD_ClassTypeDef;
```

```
USBD_ClassTypeDef  USBD_CDC =
{
  USBD_CDC_Init,
  USBD_CDC_DeInit,

  USBD_CDC_Setup,
  NULL,                    /* EP0_TxSent, */
  USBD_CDC_EP0_RxReady,

  USBD_CDC_DataIn,
  USBD_CDC_DataOut,
  NULL,
  NULL,
  NULL,

  USBD_CDC_GetHSCfgDesc,
  USBD_CDC_GetFSCfgDesc,
  USBD_CDC_GetOtherSpeedCfgDesc,
  USBD_CDC_GetDeviceQualifierDescriptor,
};
```

Defined in usbd_cdc.c

Used in usbd_core.c

Usbd structure


```
/* USB Device handle structure */
typedef struct _USBD_HandleTypeDef
{
  uint8_t                 id;
  uint32_t                dev_config;
  uint32_t                dev_default_config;
  uint32_t                dev_config_status;
  USBD_SpeedTypeDef       dev_speed;
  USBD_EndpointTypeDef    ep_in[15];
  USBD_EndpointTypeDef    ep_out[15];
  uint32_t                ep0_state;
  uint32_t                ep0_data_len;
  uint8_t                 dev_state;
  uint8_t                 dev_old_state;
  uint8_t                 dev_address;
  uint8_t                 dev_connection_status;
  uint8_t                 dev_test_mode;
  uint32_t                dev_remote_wakeup;

  USBD_SetupReqTypedef    request;
  USBD_DescriptorsTypeDef *pDesc;
  USBD_ClassTypeDef       *pClass;
  void                    *pClassData;
  void                    *pUserData;
  void                    *pData;
} USBD_HandleTypeDef;
```

USBD_ClassTypeDef

Defined in usbd_cdc.c

USBD_CDC_HandleTypeDef

Dynamically allocated in usbd_cdc.c

USBD_CDC_ItfTypeDef

Defined in usbd_cdc_if.c

Defined in usbd_conf.c.c

Handle for usb LL driver

PCD_HandleTypeDef