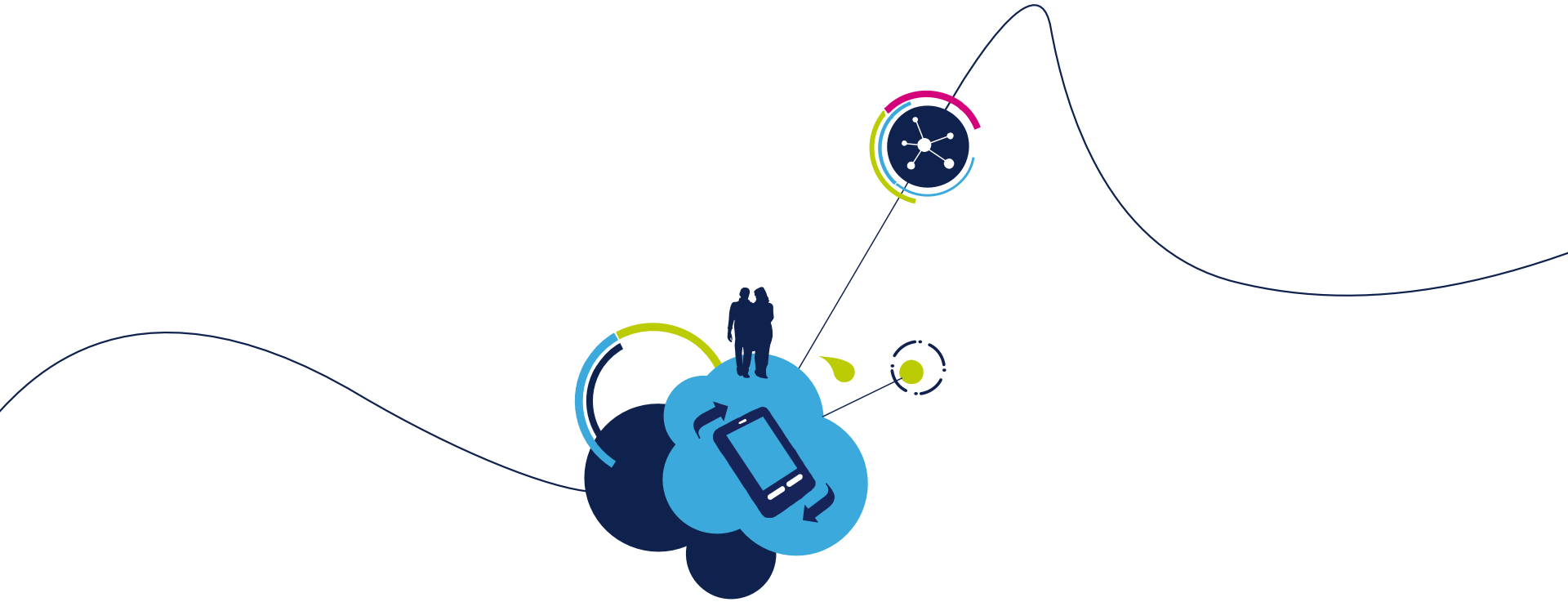


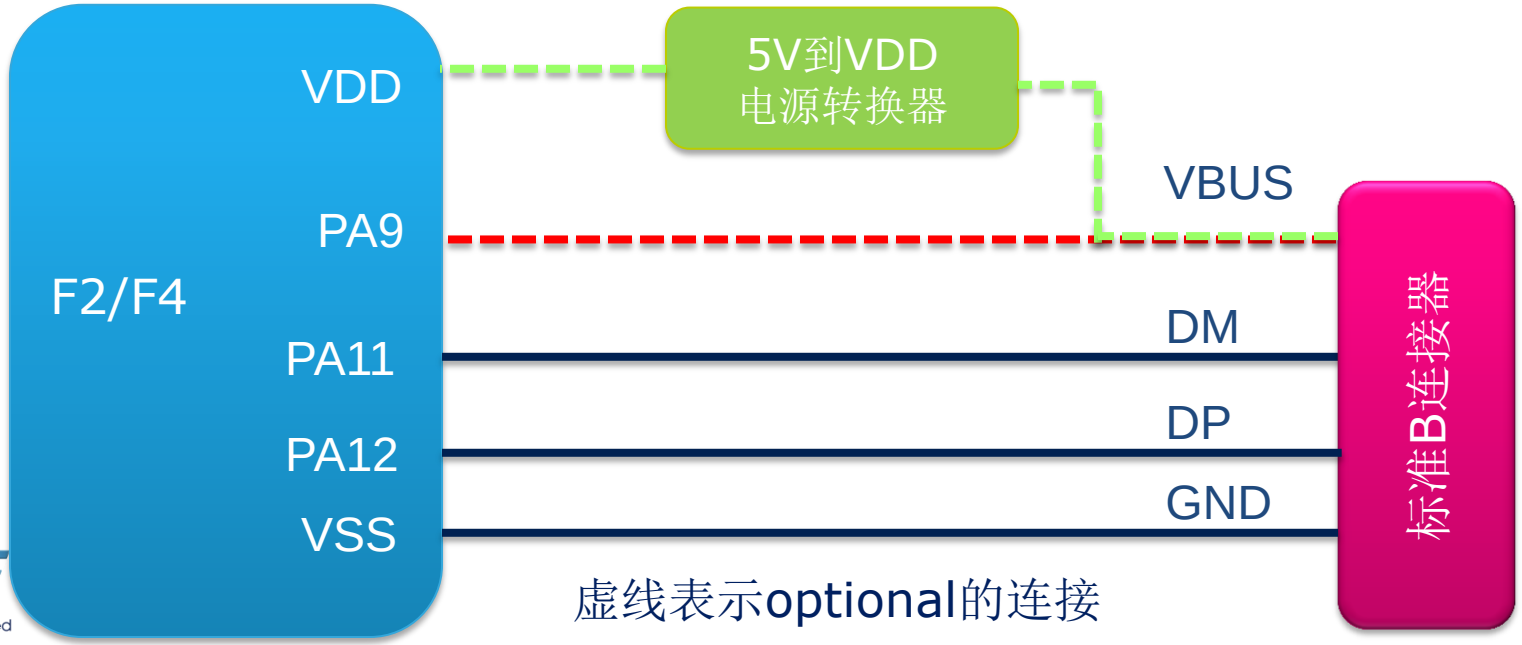
OTG模块作为USB设备

作为USB设备的四种情形

- OTG B设备
  - OTG B器件（连接的是B-side电缆）的默认状态，即插上B-side电缆时的状态
- OTG A设备
  - OTG A器件（连接的是A-side电缆）在HNP之后把角色切换成USB设备
- B器件
  - 连接的是B-side电缆，HNPCAP位被清零（角色不会变动了）
- 仅作为USB设备
  - FDMOD被置位，强制处于USB设备角色
  - 此时是否有ID信号都无所谓，被忽略

OTG_FS模块作为”device only”的连接

- PA9可以监测Vbus的供电，用于检测设备是否和主机断开
  - 消该监控时(NOVBUSSENS=1)，PA9可用作普通I/O口，此时VBUS被默认永远有效(5V)
- D+/D-线上无需串行电阻
- 芯片内的专用PLL给USB设备模块提供48MHz

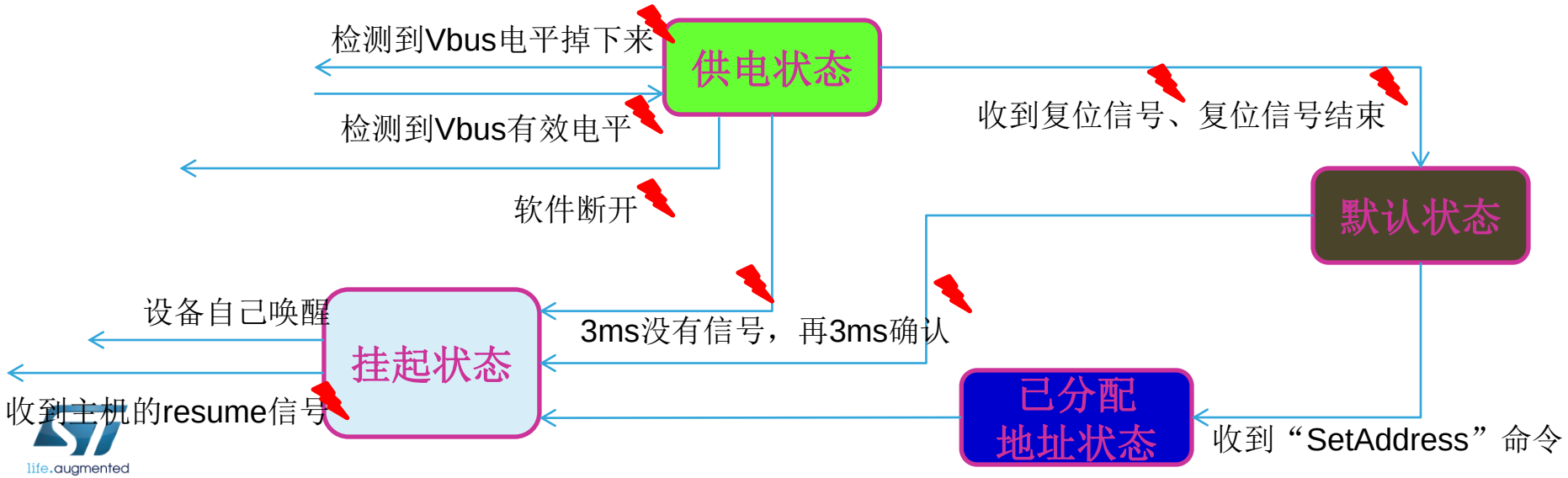


USB设备下的若干状态

- 供电状态
  - Vbus输入检测到总线上有效电平后就进入此状供电状态
  - 模块会自动使能D+上的上拉电阻，告知主机“设备连接”并产生“会话请求”中断
  - USB操作过程中若Vbus电压掉下来，模块会自动关闭D+上的上拉，并产生“会话结束”中断
  - 期望收到来自主机的USB复位信号
    - 收到复位信号，产生USBRST中断
    - 复位信号结束后，产生ENUMDNE中断
- 默认状态
  - 设备期望收到SET_ADDRESS命令
  - 一旦收到，就把新地址写入设备配置寄存器的地址域DAD@OTG_FS_DCFG
  - 之后，设备就可以以新配置的地址被主机发起的后续通信事务寻址了
- 软断开状态
  - 通过SDIS@OTG_FS_DCTL来移除D+上的上拉电阻
  - 主机端会触发“设备断开连接”中断，即时电缆还是连着的

USB设备下的若干状态(2)

- 挂起状态
  - 检测到3ms的空闲状态，触发早期挂起中断（ESUSP@OTG_FS_GINTSTS）
  - 并在3ms后，由挂起中断来确认（USBSUSP@OTG_FS_GINTSTS）
    - 标志位SUSPSTS@OTG_FS_DSTS置位
    - 模块进入挂起状态
  - 挂起状态的退出
    - 设备自己的应用置位WKUPINT@OTG_FS_DCTL 1到15ms再清零之，以来产生远程唤醒信号
    - 若是收到来自主机的继续信号则触发中断（RWUSIG@OTG_FS_GINTSTS），且设备的挂起位被自动清零。



设备对连入主机和从主机拔出的检测

- 原理：PA9检测总线电压Vbus有效否
  - Vbus sensing功能使能的情况下
  - “会话请求”中断
  - “会话结束”中断
- 寄存器
  - SRQINT@OTG_FS_GINTSTS
  - SEDET@OTG_FS_GOTGINT

HAL_PCD_IRQHandler @ <stm32f4xx_hal_pcd.c>

```
/* Handle Connection event Interrupt */
  if(__HAL_PCD_GET_FLAG(hpcd, USB_OTG_GINTSTS_SRQINT))
  {
    HAL_PCD_ConnectCallback(hpcd);
    __HAL_PCD_CLEAR_FLAG(hpcd, USB_OTG_GINTSTS_SRQINT);
  }

/* Handle Disconnection event Interrupt */
  if(__HAL_PCD_GET_FLAG(hpcd, USB_OTG_GINTSTS_OTGINT))
  {
    temp = hpcd->Instance->GOTGINT;
    if((temp & USB_OTG_GOTGINT_SEDET) == USB_OTG_GOTGINT_SEDET)
    {
      HAL_PCD_DisconnectCallback(hpcd);
    }
    hpcd->Instance->GOTGINT |= temp;
  }
```

USB设备的端点资源

- 双向控制端点：EP0
  - IN和OUT transaction由单独的寄存器处理
  - 控制寄存器：OTG_FS_DIEP**CTL0**、OTG_FS_DOEP**CTL0**
  - 传输配置寄存器：OTG_FS_DIEP**TSIZ0**、 OTG_FS_DOEP**TSIZ0**
  - 中断状态寄存器： OTG_FS_DIEP**INTx(x=0)**、 OTG_FS_DOEP**INTx(x=0)**
- 3个IN方向的EP
  - 支持同步、批量、中断等传输类型、支持“未完成的同步IN传输”中断
  - 控制寄存器：OTG_FS_DIEP**CTLx** (x=1~3)
  - 传输配置寄存器：OTG_FS_DIEP**TSIZx** (x=1~3)
  - 中断状态及屏蔽寄存器： OTG_FS_DIEP**INTx** / OTG_FS_DIEPMSK (x=1~3)
- 3个OUT方向的EP
  - 支持同步、批量、中断等传输类型、支持“未完成的同步OUT传输”中断
  - 控制寄存器：OTG_FS_DOEP**CTLx** (x=1~3)
  - 传输配置寄存器：OTG_FS_DOEP**TSIZx** (x=1~3)
  - 中断状态及屏蔽寄存器： OTG_FS_DOEP**INTx** / OTG_FS_DOEPMSK (x=1~3)

端点控制寄存器：DIEPCTL/ DOEPCTL

位	域	描述	EP0_IN	EP0_OUT	EPx_IN	Epx_OUT
31	EPENA	端点使能				
30	EPDIS	端点关闭				
29	SODDFRM	仅适用于同步端点：set odd frame	X	X		
28	SD0PID/ SEVNFRM	仅适用于中断/批量端点：set DATA0 PID 仅适用于同步端点：set even frame	X	X		
27	SNAK	控制该端点回复NAK握手信号				
26	CNAK	清除该端点的NAK控制位				
25:22	TXFNUM	仅用于IN端点：该端点对应的TX FIFO编号		X		X
21	STALL	控制该端点回复STALL握手信号				
20	SNPM	Snoop mode	X		X	
19:18	EPTYP	端点上支持的传输类型				
17	NAKSTS	依据什么回复NAK握手				
16	EONUM/ DPID	仅适用于同步IN端点：even/odd frame 仅适用于中断/批量类型的IN端点	X	X		
15	USBAEP	说明该端点在当前配置和接口是否使用				
→→	MPSIZ	最大包长（单位：字节）	1:0		10:0	

“X”表示不适用

端点传输寄存器：DIEPTSIZ/ DOEPTSIZ

- DIEPTSIZx/ DOEPTSIZx
  - 应用必须在使能端点之前配置该端点上的传输长度
    - 传输包含的字节数目
    - 传输包含的数据包数目
  - 应用还可读取传输状态

位	域	描述	EP0_IN	EP0_OUT	EPx_IN	Epx_OUT
30:29	Reserved.	保留	Y			
	STUPCNT	端点能够收到多少连续的SETUP数据包		Y		
	MCNT	**仅用于周期性IN端点**：每个frame必须发出的数据包个数 (1, 2, 3)			Y	
	RXDPID/ STUPCNT	**仅用于同步类型的OUT端点**：上一个收到的数据包的PID (DATA0, DATA1, DATA2, MDATA) **仅用于控制类型的OUT端点**：能够收到多少连续的SETUP数据包 (1, 2, 3)				Y
→→	PKTCNT	Transfer中数据包的个数(最大包或短包)	20:19	19	28:19	
→→	XFRSIZ	Transfer中字节的个数，完成后触发中断；可以设置成最大包长，这样每完成一个包都有中断	6:0		18:0	

端点中断状态寄存器

DIEPINTx (x=0…7)

位	域名		
7	TXFE	该端点的发送FIFO空	应用需要先读取OTG_FS_DAINT来知晓发送中断的是哪个端点；再到对应的DIEPINTX来查看具体中断事件
6	INEPNE	该端点上的NAK位生效了	
4	ITTXFE	发送FIFO空，却收到了IN令牌	
3	TOC	**控制类型IN端点上**检测到超时(从上一个收到的IN令牌开始，发送超时了)	
1	EPDISD	端点被应用关闭	
0	**XFRC**	**该端点上的传输成功结束**	

DOEPINTx (x=0…7)

位	域名		
6	B2BSTUP	**EP0的OUT端点上**收到3个以上的背靠背SETUP包	应用需要先读取OTG_FS_DAINT来知晓发送中断的是哪个端点；再到对应的DIEPINTX来查看具体中断事件
4	OTEPDIS	**EP0的OUT端点上**收到OUT令牌，但是端点还未使能	
3	STUP	**控制类型OUT端点上**的SETUP阶段结束了，且没有背靠背的SETUP包，应用可以解码SETUP中的数据内容了	
1	EPDISD	端点被应用关闭	
0	**XFRC**	**该端点上的传输成功结束**	

OTG_FS core interrupt register (OTG_FS_GINTSTS)

内核中断寄存器

OTG_FS interrupt mask register (OTG_FS_GINTMSK)

OTG事件中断

查看详细事件信息@ OTG_FS_GOTGINT

OUT端点中断

IN端点中断

查看具体端点号@ OTG_FS_DAINT

OTG_FS all endpoints interrupt mask register (OTG_FS_DAINTMSK)

查看具体中断源 @ OTG_FS_DIEPINTx

OTG_FS device IN endpoint common interrupt mask (OTG_FS_DIEPMSK)

查看具体中断源 @ OTG_FS_DOEPINTx

OTG_FS device OUT endpoint common interrupt mas (OTG_FS_DOEPMSK)

主机通道中断

主机端口中断

查看具体中断源@ OTG_FS_HPRT

查看具体通道号@ OTG_FS_HAINT

查看具体中断源@ OTG_FS_HCINTx

life.augmented

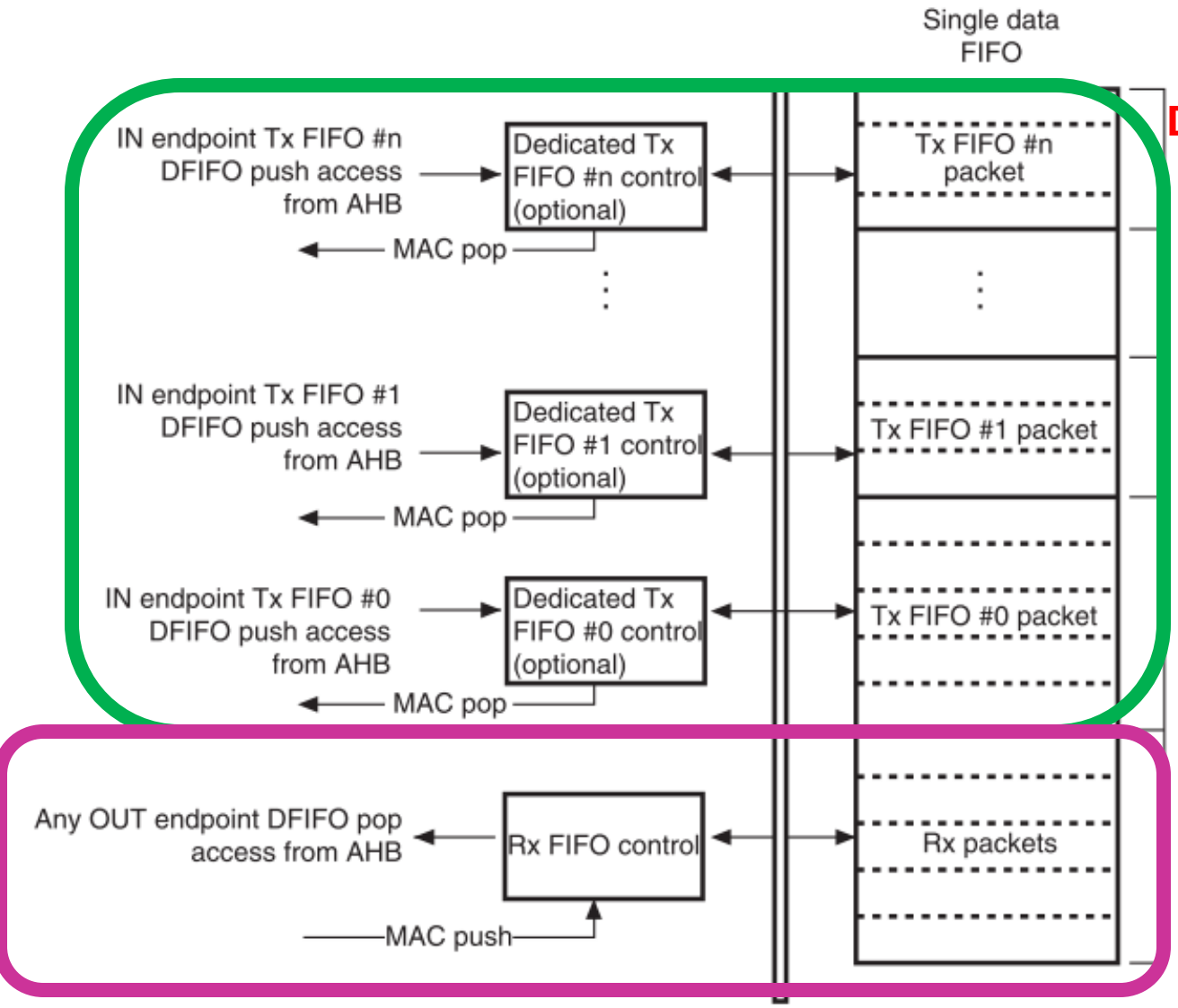
GINTSTS中适用于设备的中断

位	域名	描述	适用于的角色
31	WKUPINT	检测到Resume信号（。。。）	【D】（【H】）
30	SRQINT	Vbus电压有效（。。。）	【D】（【H】）
21	INCOMPISOOUT	周期传输未完成（。。。）	【D】（【H】）
20	IISOIXFR	未完成的同步OUT传输（。。。）	【D】（【H】）
19	OEPINT	OUT端点中断	【D】继续查看OTG_FS_**DAINT**和OTG_FS_**DOEPINTx**来确定哪个端点上的具体什么中断事件
18	IEPINT	IN端点中断	【D】继续查看OTG_FS_**DAINT**和OTG_FS_**DIEPINTx**来确定哪个端点上的具体什么中断事件
15	EOPF	一帧中用于周期传输的时间结束了	【D】时间门限在OTG_FS_DCFG.PFIVL中设置
14	ISOODRP	由于RxFIFO空间不够造成同步OUT包丢包	【D】
13	ENUMDNE	设备速度枚举完成	【D】读取OTG_FS_DSTS获得枚举的速度
12	USBRST	检测到USB复位	【D】
11	USBSUSP	检测到USB挂起状态	【D】
10	ESUSP	检测到早期USB挂起状态	【D】

GINTSTS中适用于设备的中断(2)

位	域名		
7	GONAKEFF	“设置全局OUT NAK”位生效	【D】
6	GINAKEFF	“设置全局非周期IN NAK”位生效	【D】
4	RXFLVL	接收FIFO非空，至少有一个数据包可读	【D】（【H】）
3	SOF	检测到SOF信号（。。。）	【D】（【H】）
1	MMIS	角色不匹配中断	【D】（【H】）
0	CMOD	当前角色	【D】（【H】）

作为USB设备时的FIFO架构



FIFO深度
DIEPTXF3[31:16]
FIFO始址
DIEPTXF3[15:0]

DIEPTXF2

DIEPTXF1

DTXFSTSx
(x-0~3)

FIFO深度
DIEPTXF0[31:16]
FIFO始址
DIEPTXF0[15:0]

FIFO深度
GRXFSIZ[15:0]

GRXSTSP

A1 = 0 (Rx start address fixed to 0)

接收FIFO起始地址固定=0

设备FIFO的使用（1）

	FIFO始址和大小的配置	FIFO当前状态的查看	FIFO的使用备注
端点共享的接收FIFO	**GRXFSIZ** >> 接收FIFO始址固定是0	**GRXSTSP** >> 收到的数据包状态 >> 收到的数据包PID >> 收到的数据包字节数 >> 收到的数据包所属端点号	收到的数据包一个挨一个的存放； Packet的状态信息和data payload一起存放； 只要RX FIFO中有数据就不断产生非空中断RXFLVL@GINTSTS RX FIFO没有空间时主机被NAK
每个IN端点的发送FIFO	**DIEPTXFx (x=1~3)** >> FIFO起始地址 >> FIFO大小 **DIEPTXF0** >> 和主机非周期发送FIFO尺寸寄存器复用 >> FIFO起始地址 >> FIFO大小	**DTXFSTSx (x=0~3)** >> FIFO空闲空间 0 → 全满 X → X个字可用 (x!=0)	该缓存区用于存放发送packet中的数据负载； 只要TX FIFO半空或全空就会触发TX FIFO空中断：TXFE@DIEPINTx； 只有TX FIFO有足够空闲空间，应用才可提前把要发送的数据和请求放进去
共性	宽度为32位字，最小容量64B；最大容量1KB		

作为USB设备时的FIFO分配

- 一共1.25KB，即1280B，以32-bit为单位 → 320个表项
- RX FIFO
  - 为SETUP包预留空间
    - 至少10个表项
  - 全局OUT NAK
    - 一个表项
  - 为数据包预留空间
    - 最小size要是(最大的MPS/4+1)，因为包的状态信息也一起存入FIFO
    - 如果有多个ISO EP，最小size要是[2*(最大MPS/4)+1]个表项
  - 每个transfer complete status
    - 一个表项
- TX FIFO
  - 最小size要是最大的那个IN EP的MPZ
  - 当然越大性能越好

CDC Device Demo.设置FIFO大小

- USBD_LL_Init() ← USBD_Init() ← main()

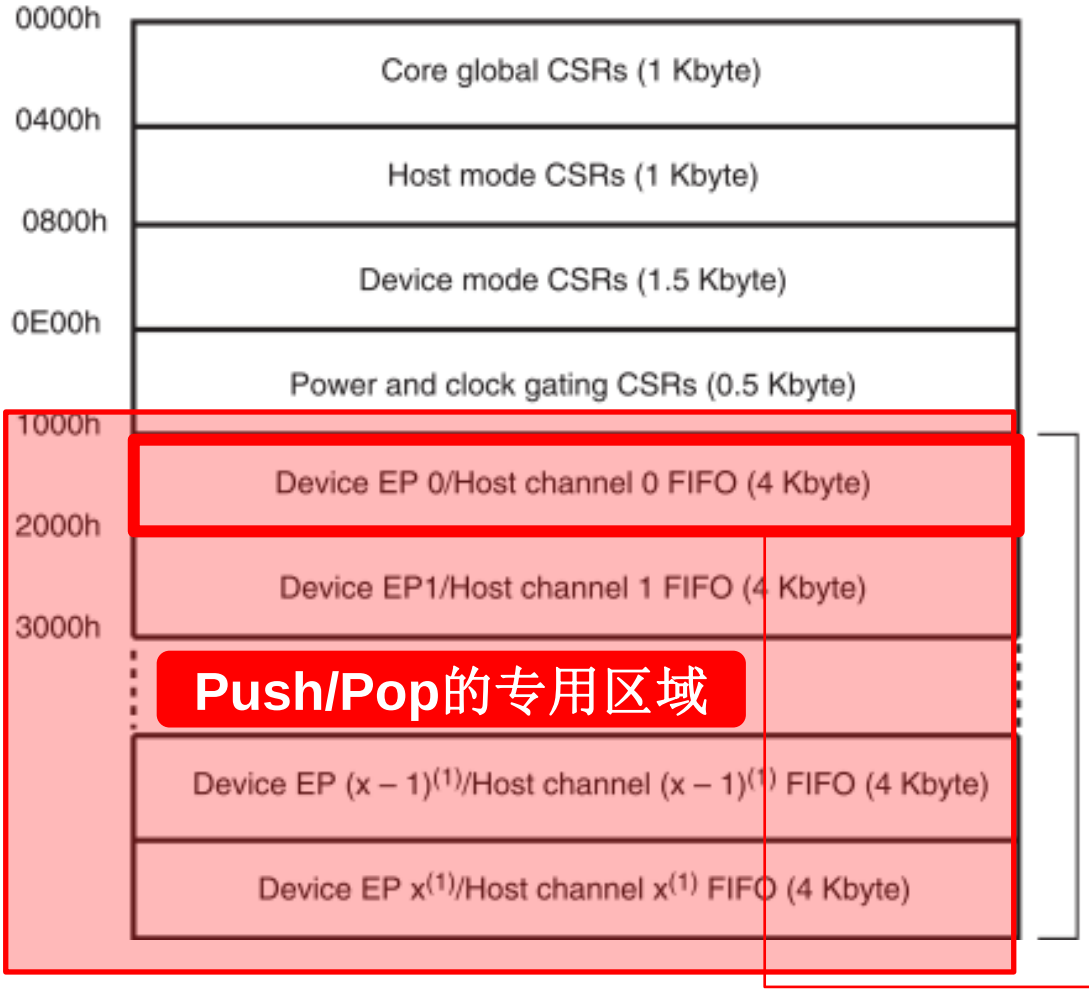
	RX FIFO	始址	EP0TXF	始址	EP1TXF
FS OTG	**128-word** 512B	0x80	**64-word** 384B	0xE0	**128-word** 256B
HS OTG	512-word	0x200	128-word	0x280	-372-word

USBD_LL_Init() @ <usbd_conf.c>

```
#ifdef USE_USB_FS
  HAL_PCDEx_SetRxFiFo(&hpcd, 0x80);
  HAL_PCDEx_SetTxFiFo(&hpcd, 0, 0x40);
  HAL_PCDEx_SetTxFiFo(&hpcd, 1, 0x80);

#ifdef USE_USB_HS
 HAL_PCDEx_SetRxFiFo(&hpcd, 0x200);
  HAL_PCDEx_SetTxFiFo(&hpcd, 0, 0x80);
  HAL_PCDEx_SetTxFiFo(&hpcd, 1, 0x174);
```

设备FIFO的使用（2）
读取端点的RX FIFO/写端点的TX FIFO



>> 这些寄存器，在设备和主机模式下都可访问

>> 用于对特定端点或者通道的DFIFO的读或者写访问

>> 若该通道是IN或者该端点是OUT，则只能对该FIFO执行写操作

>>若该通道是OUT或者该端点是IN，则只能对该FIFO执行读操作

例如：
设备IN EP0/ 主机OUT通道0 的写地址
也是
设备OUT EP0/ 主机IN 通道0 的读地址

life.augmented

```
void *USB_ReadPacket (*USBx, uint8_t *dest, uint16_t len)
{
  uint32_t i=0;
  uint32_t count32b = (len + 3) / 4;

  for ( i = 0; i < count32b; i++, dest += 4 )
  {
    *(__packed uint32_t *)dest = USBx_DFIFO(0);
  }
  return ((void *)dest);
}
```

为何是固定0？因为接收FIFO是共享的！

```
                                        0x1000                                  0x1000
#define USBx_DFIFO(i) (USBx + USB_OTG_FIFO_BASE + (i) * USB_OTG_FIFO_SIZE)
```

```
USB_WritePacket (*USBx, uint8_t *src, uint8_t ch_ep_num, uint16_t len, uint8_t dma)
{
  uint32_t count32b= 0 , i= 0;
  if (dma == 0)
  {
    count32b =  (len + 3) / 4;
    for (i = 0; i < count32b; i++, src += 4)
    {
      USBx_DFIFO(ch_ep_num) = *((__packed uint32_t *)src);
    }
  }
  return HAL_OK;
}
```

```
                                        0x1000                                  0x1000
#define USBx_DFIFO(i) (USBx + USB_OTG_FIFO_BASE + (i) * USB_OTG_FIFO_SIZE)
```

life.augmented

作为USB设备的编程模型

- 结合USB库例程代码讲解…
- 请移步后续视频获得更多详细信息

OTG作为设备的使用小结

- 典型硬件连接
- 设备状态
  - 供电有效，软件断开、默认状态，挂起…
- 设备端点及相关寄存器
  - 端点静态属性配置，端点动态传输配置
- 设备FIFO的使用
- 设备中断
- 设备编程模型

CONFIDENTIALITY OBLIGATIONS:

THIS DOCUMENT CONTAINS SENSITIVE INFORMATION.

IT IS CLASSIFIED “MICROCONTROLLERS, MEMORIES & SECURE MCUs (MMS) RESTRICTED AND ITS DISTRIBUTION IS SUBMITTED TO ST/MMS AUTHORIZATION

AT ALL TIMES YOU SHOULD COMPLY WITH THE FOLLOWING SECURITY RULES:

- DO NOT COPY OR REPRODUCE ALL OR PART OF THIS DOCUMENT
- KEEP THIS DOCUMENT LOCKED AWAY
- FURTHER COPIES CAN BE PROVIDED ON A “NEED TO KNOW BASIS”, PLEASE CONTACT YOUR LOCAL ST SALES OFFICE OR DOCUMENT WRITTER.