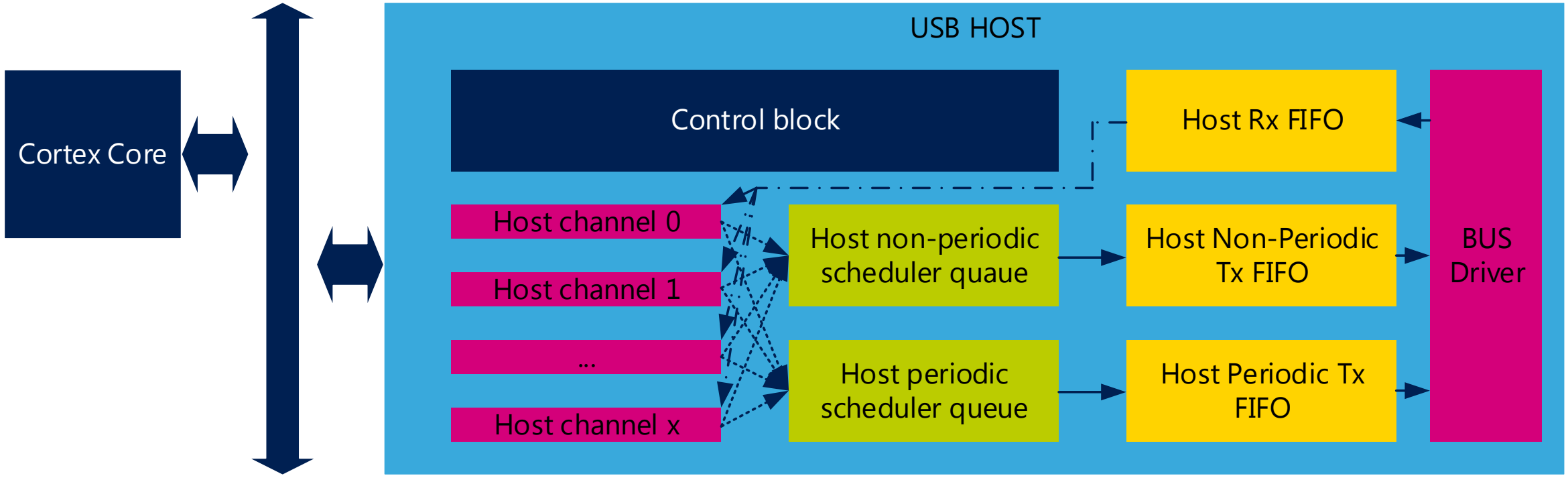
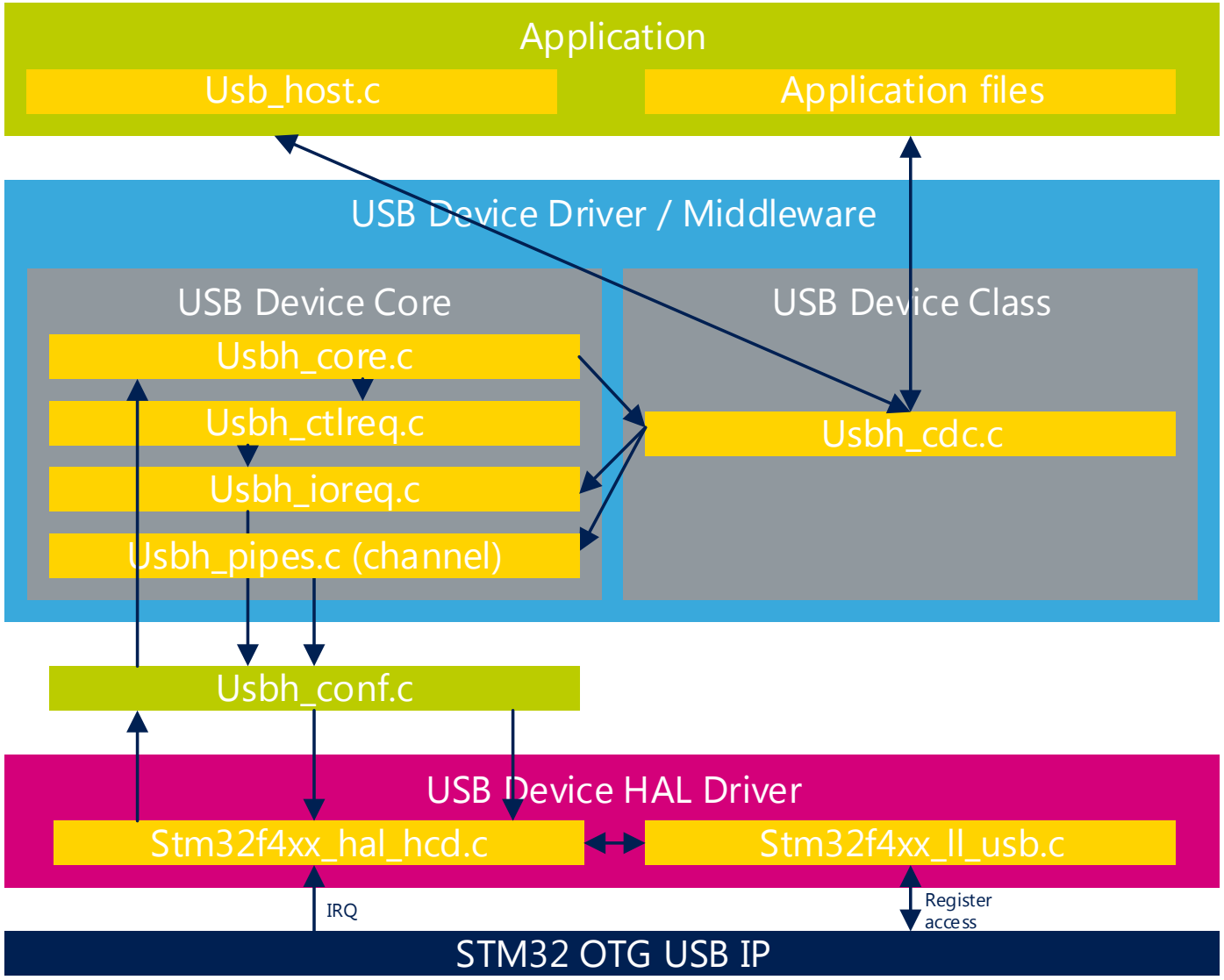
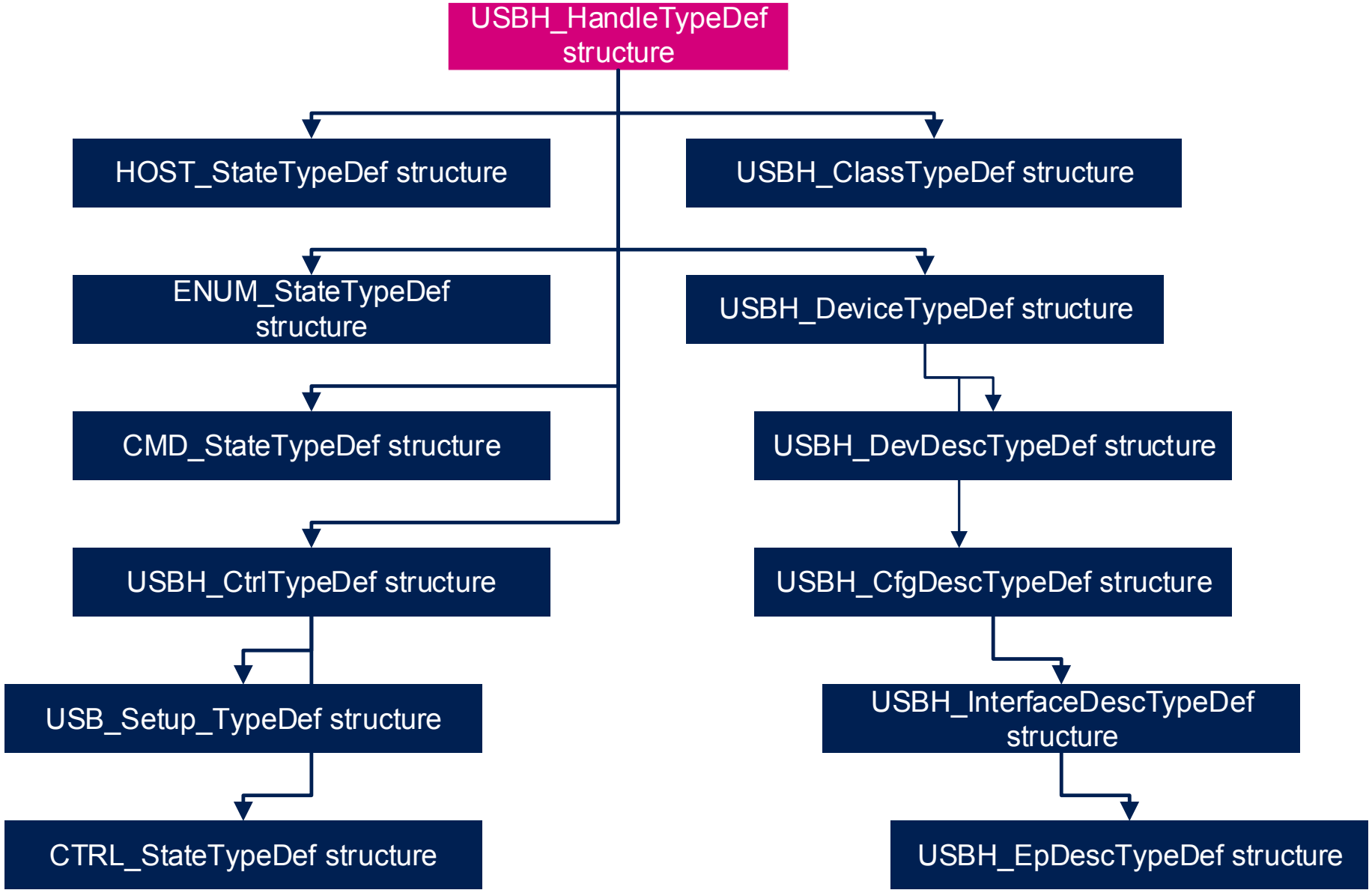
USB Host HW



Host Library



USB Host stack structure



USB Host stack structure


```
/* USB Host handle structure */
typedef struct _USBH_HandleTypeDef
{
  __IO HOST_StateTypeDef     gState;        /*  Host State Machine Value */
  ENUM_StateTypeDef      EnumState;    /* Enumeration state Machine */
  CMD_StateTypeDef       RequestState;
  USBH_CtrlTypeDef       Control;
  USBH_DeviceTypeDef     device;
  USBH_ClassTypeDef*     pClass[USBH_MAX_NUM_SUPPORTED_CLASS];
  USBH_ClassTypeDef*     pActiveClass;
  uint32_t               ClassNumber;
  uint32_t               Pipes[15];
  __IO uint32_t          Timer;
  uint8_t                id;
  void*                  pData;
  void                  (* pUser )(struct _USBH_HandleTypeDef *pHandle, uint8_t id);
#if (USBH_USE_OS == 1)
  osMessageQId           os_event;
  osThreadId             thread;
#endif
} USBH_HandleTypeDef;
```

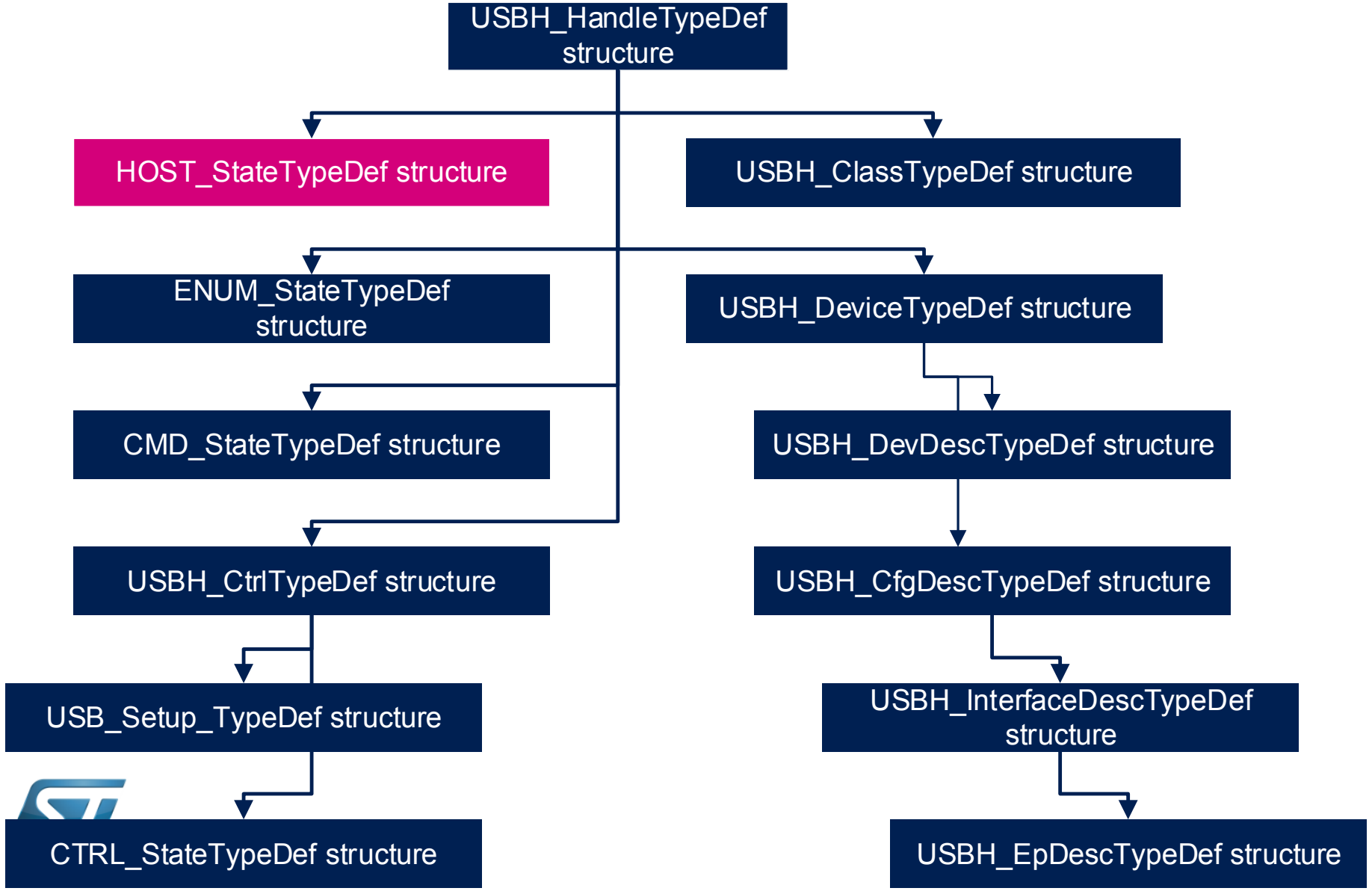
List of available classes

Class of connected device, must be in pClass list

Give information about pipes/channel usage (used or free)

Pointer on HAL structure HCD_HandleTypeDef

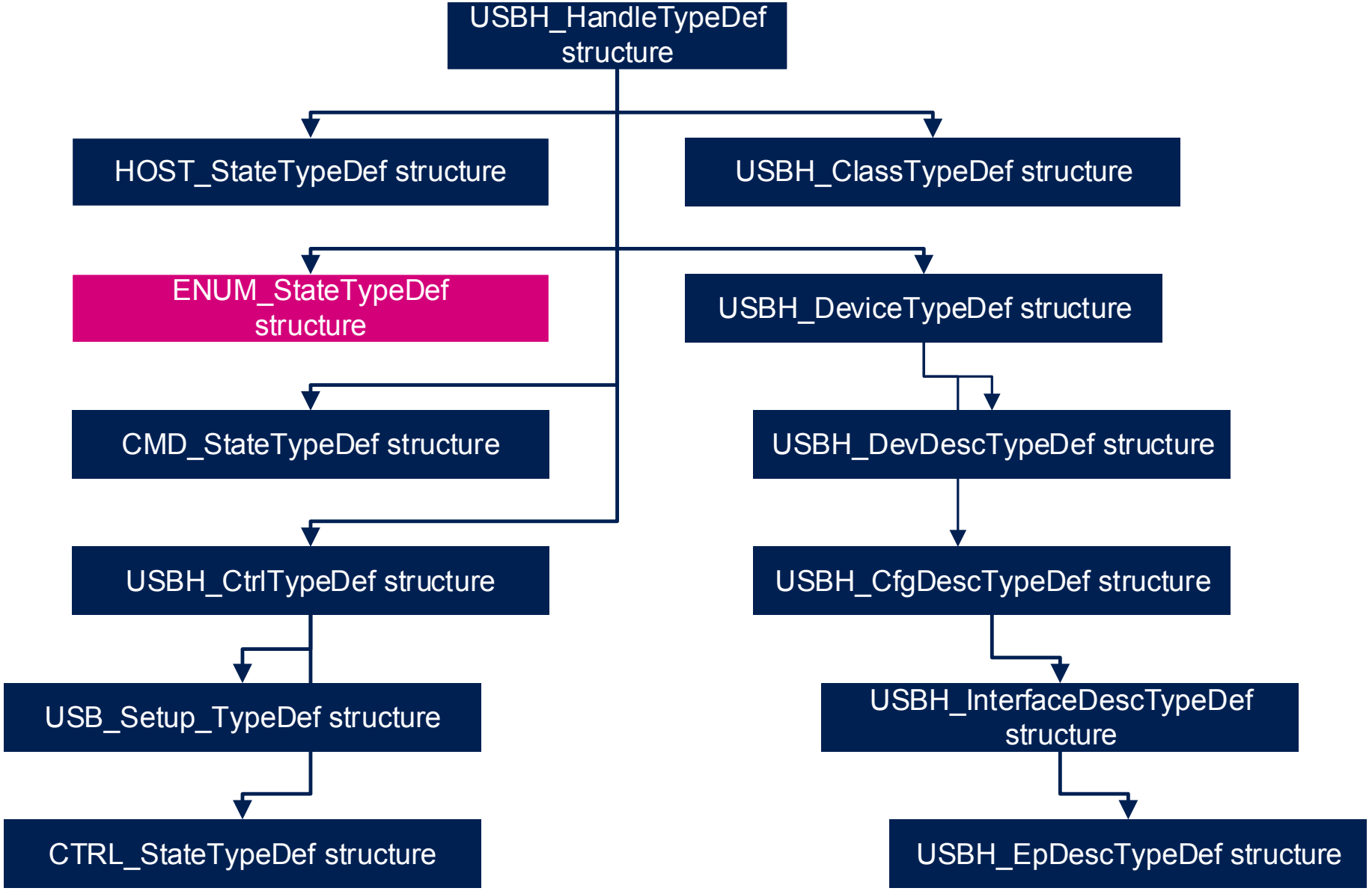
Host state


USBH_HandleTypeDef structure
HOST_StateTypeDef structure
USBH_ClassTypeDef structure
ENUM_StateTypeDef structure
USBH_DeviceTypeDef structure
CMD_StateTypeDef structure
USBH_DevDescTypeDef structure
USBH_CtrlTypeDef structure
USBH_CfgDescTypeDef structure
USB_Setup_TypeDef structure
USBH_InterfaceDescTypeDef structure
CTRL_StateTypeDef structure
USBH_EpDescTypeDef structure

Host state


```
/* Following states are used for gState */
typedef enum
{
  HOST_IDLE =0,
  HOST_DEV_WAIT_FOR_ATTACHMENT,
  HOST_DEV_ATTACHED,
  HOST_DEV_DISCONNECTED,
  HOST_DETECT_DEVICE_SPEED,
  HOST_ENUMERATION,
  HOST_CLASS_REQUEST,
  HOST_INPUT,
  HOST_SET_CONFIGURATION,
  HOST_CHECK_CLASS,
  HOST_CLASS,
  HOST_SUSPENDED,
  HOST_ABORT_STATE,
}HOST_StateTypeDef;
```

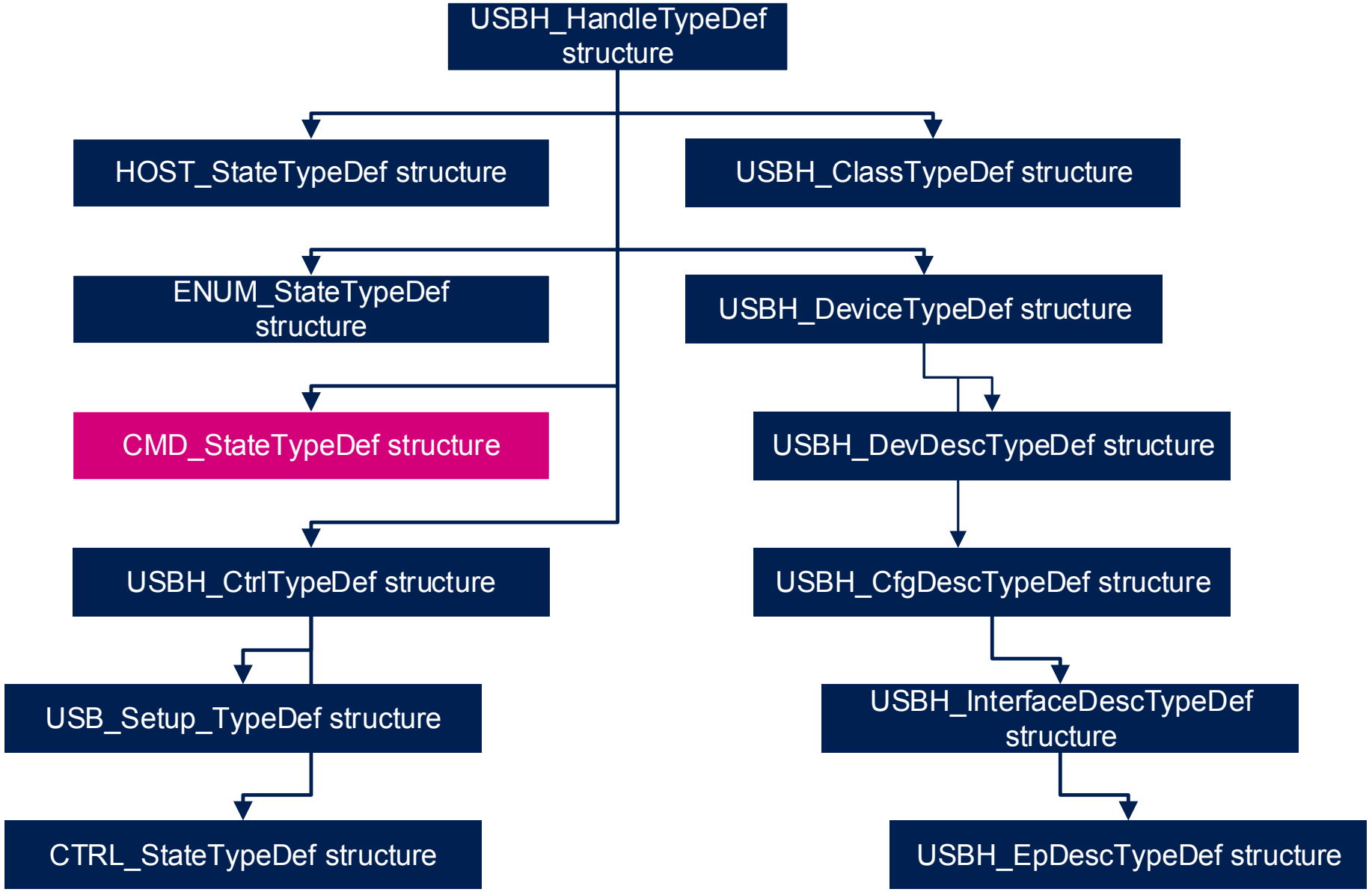
Host enumeration state



Host enumeration state


```
/* Following states are used for
EnumerationState */
typedef enum
{
  ENUM_IDLE = 0,
  ENUM_GET_FULL_DEV_DESC,
  ENUM_SET_ADDR,
  ENUM_GET_CFG_DESC,
  ENUM_GET_FULL_CFG_DESC,
  ENUM_GET_MFC_STRING_DESC,
  ENUM_GET_PRODUCT_STRING_DESC,
  ENUM_GET_SERIALNUM_STRING_DESC,
} ENUM_StateTypeDef;
```

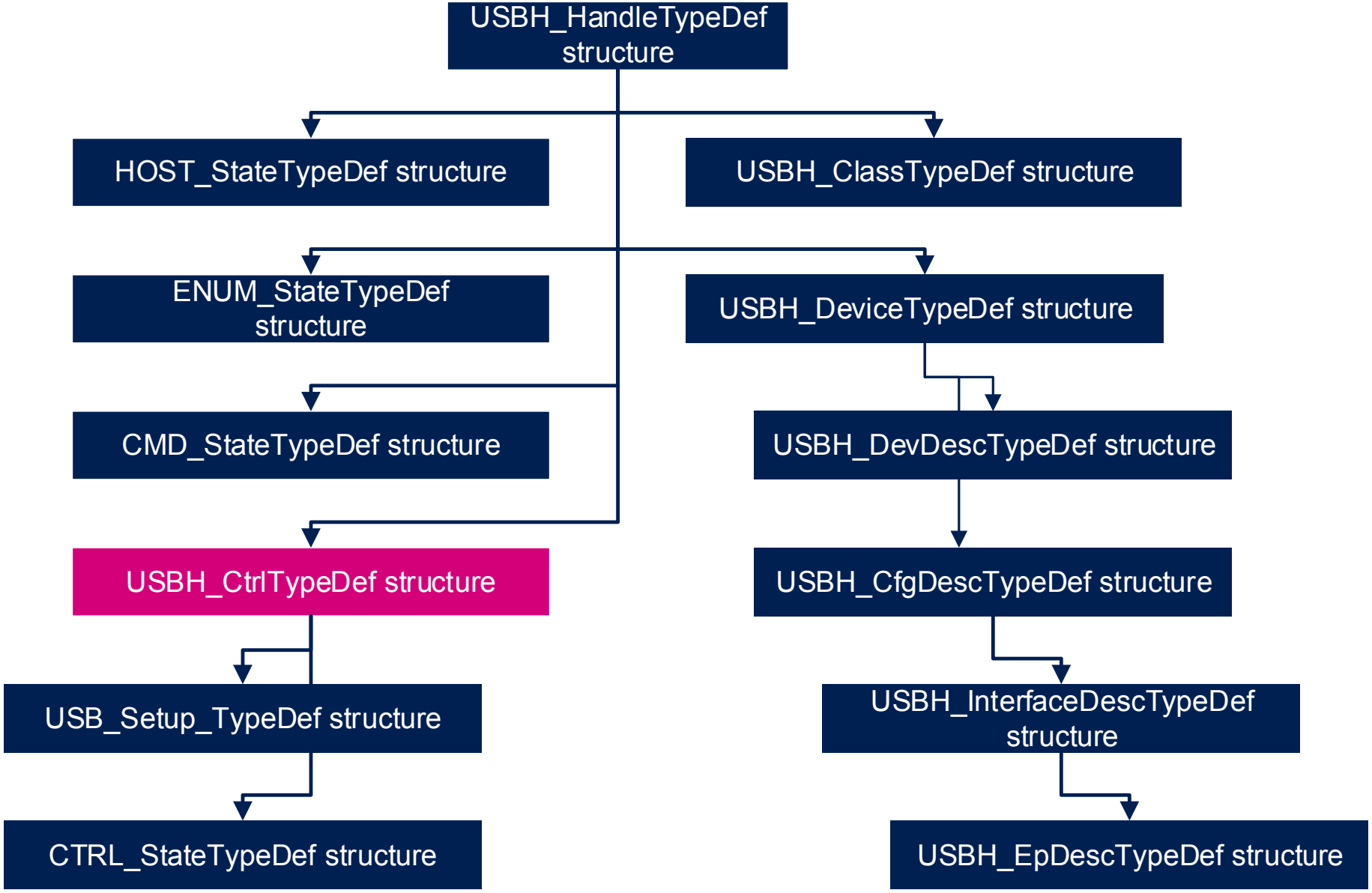
Host request state



Host request state


```
/* Following states are used for
RequestState */
typedef enum
{
  CMD_IDLE =0,
  CMD_SEND,
  CMD_WAIT
} CMD_StateTypeDef;
```

Host control request structure

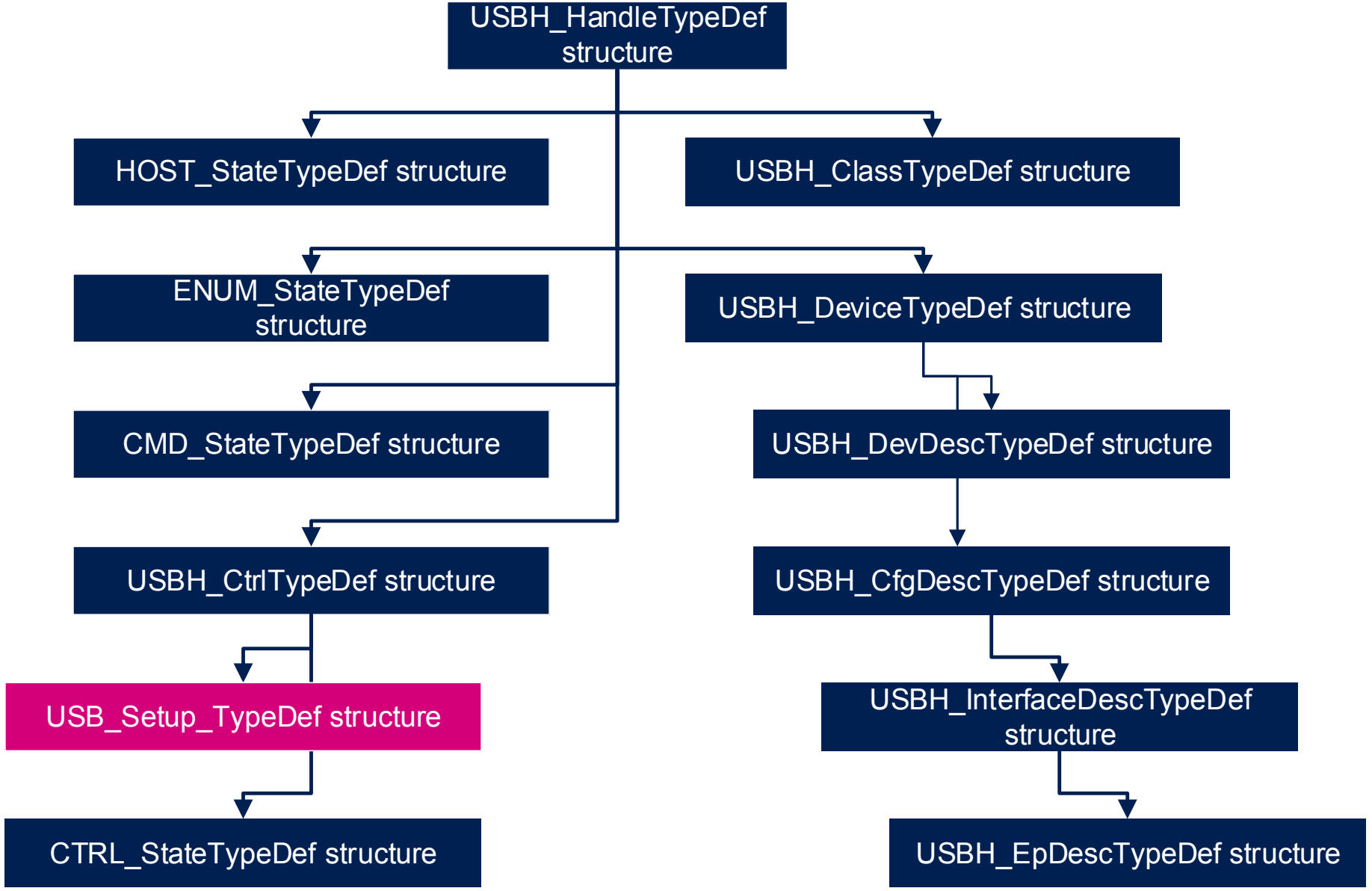


Host control request structure


```
/* Control request structure */
typedef struct
{
  uint8_t                pipe_in;
  uint8_t                pipe_out;
  uint8_t                pipe_size;
  uint8_t                *buff;
  uint16_t               length;
  uint16_t               timer;
  USB_Setup_TypeDef      setup;
  CTRL_StateTypeDef      state;
  uint8_t                errorcount;

} USBH_CtrlTypeDef;
```

Host setup packet structure

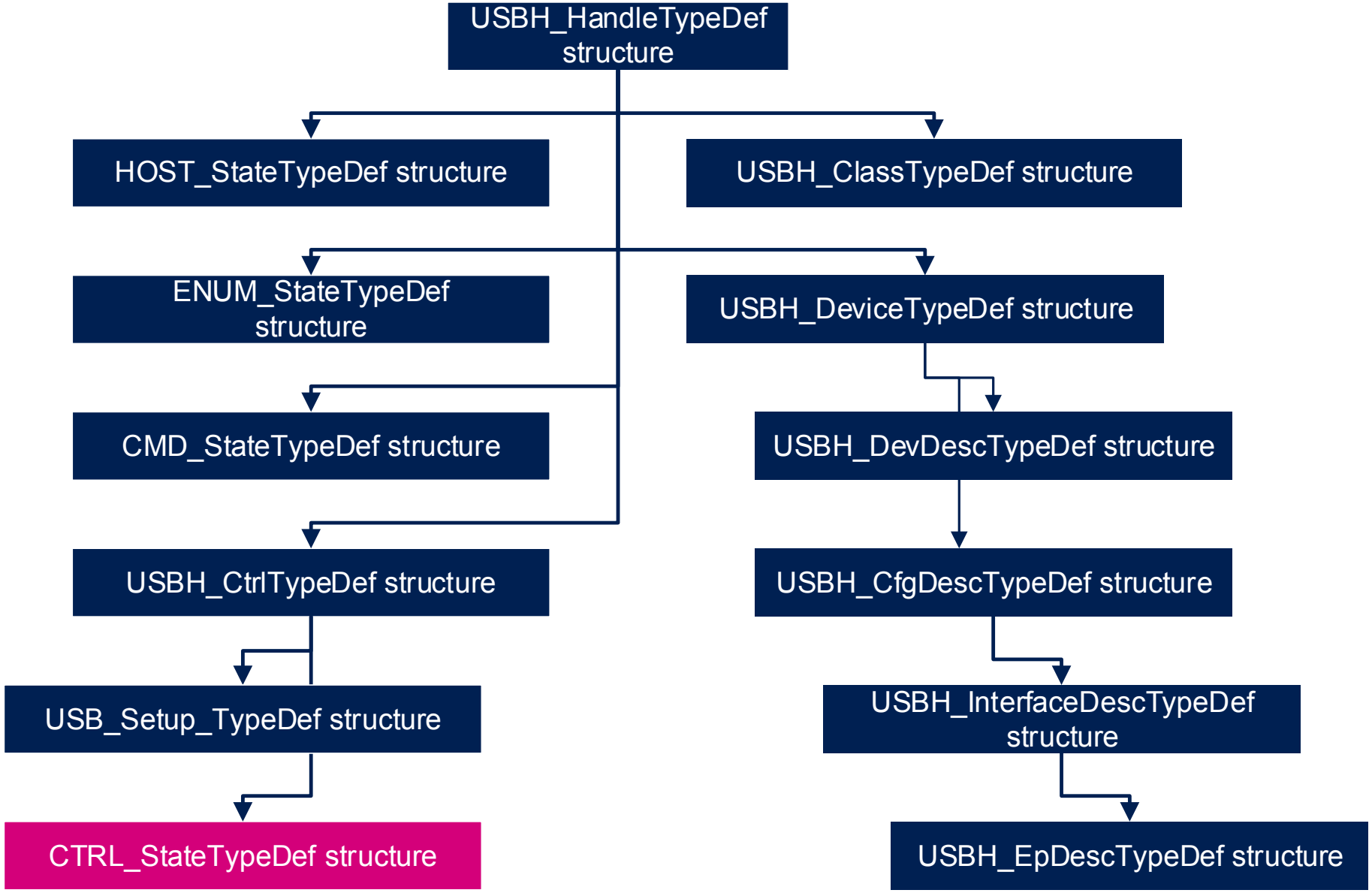


Host setup packet structure


```
typedef union _USB_Setup
{
  uint32_t d8[2];

  struct _SetupPkt_Struc
  {
    uint8_t           bmRequestType;
    uint8_t           bRequest;
    uint16_t_uint8_t  wValue;
    uint16_t_uint8_t  wIndex;
    uint16_t_uint8_t  wLength;
  } b;
}
USB_Setup_TypeDef;
```

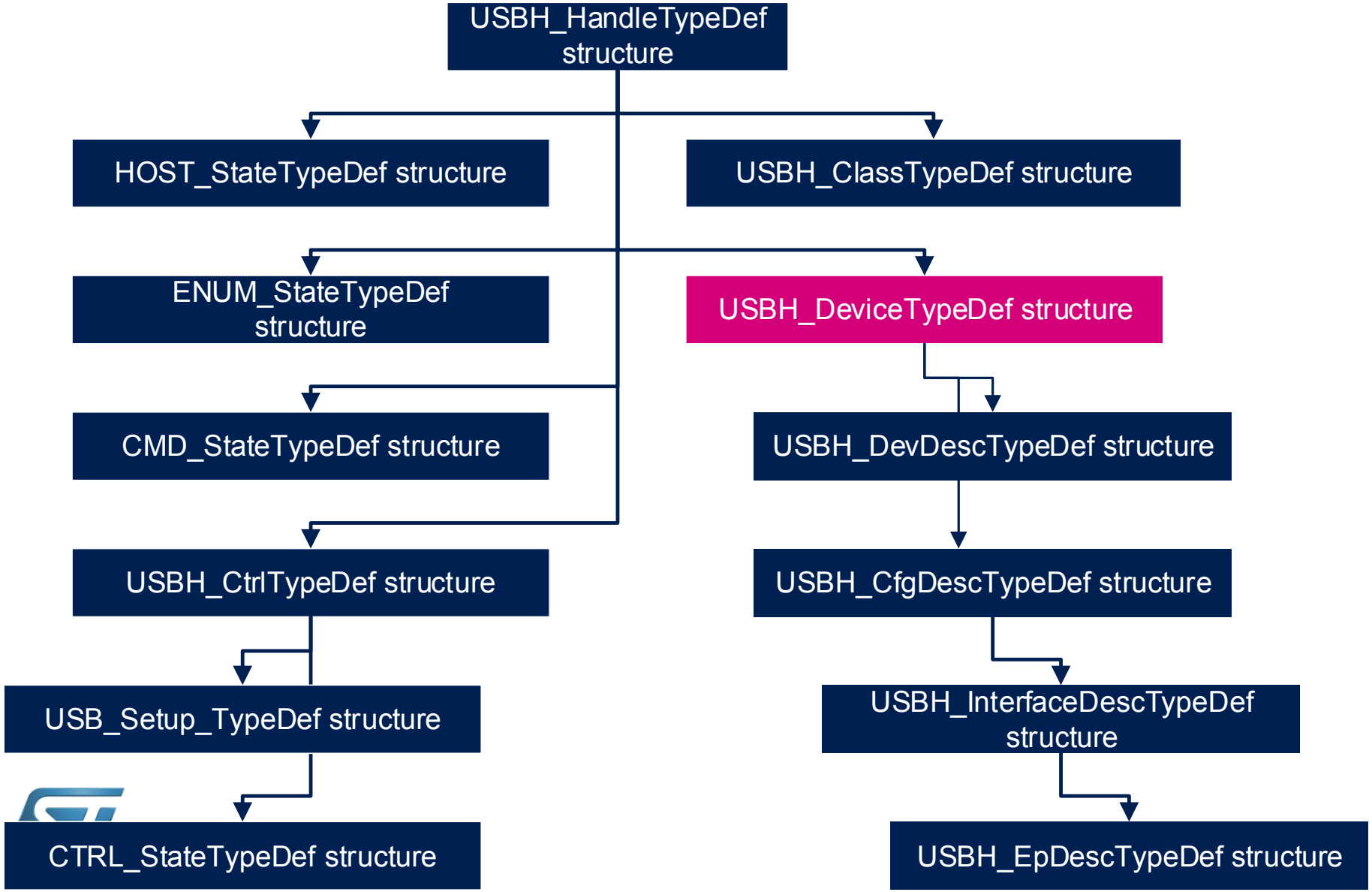
Host control request state



Host control request state


```
/* Following states are used for
CtrlXferStateMachine */
typedef enum
{
  CTRL_IDLE =0,
  CTRL_SETUP,
  CTRL_SETUP_WAIT,
  CTRL_DATA_IN,
  CTRL_DATA_IN_WAIT,
  CTRL_DATA_OUT,
  CTRL_DATA_OUT_WAIT,
  CTRL_STATUS_IN,
  CTRL_STATUS_IN_WAIT,
  CTRL_STATUS_OUT,
  CTRL_STATUS_OUT_WAIT,
  CTRL_ERROR,
  CTRL_STALLED,
  CTRL_COMPLETE
}CTRL_StateTypeDef;
```

Host attached device structure

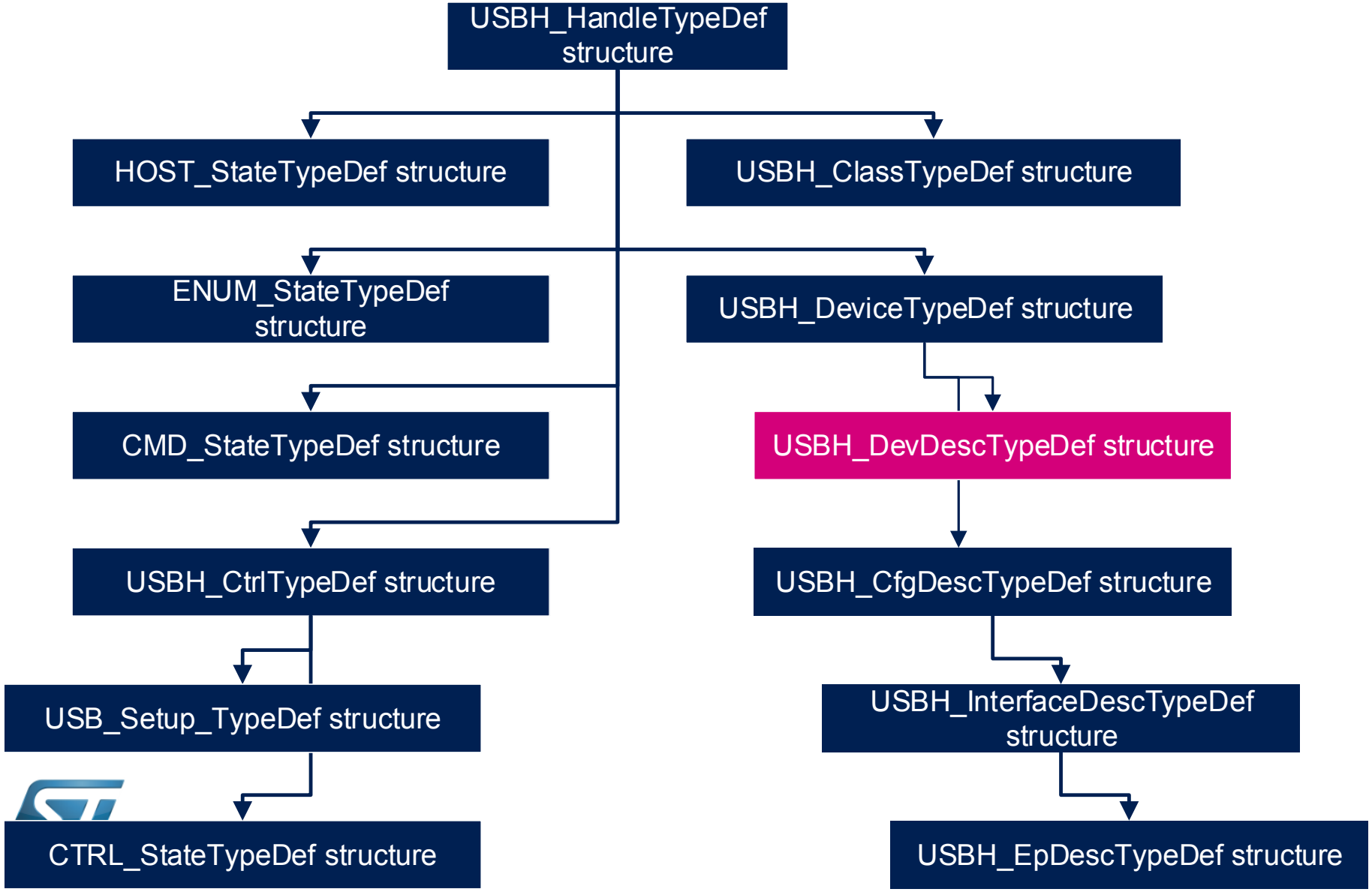


Host attached device structure


```
/* Attached device structure */
typedef struct
{
#if (USBH_KEEP_CFG_DESCRIPTOR == 1)
  uint8_t                         CfgDesc_Raw[USBH_MAX_SIZE_CONFIGURATION];
#endif
  uint8_t                         Data[USBH_MAX_DATA_BUFFER];
  uint8_t                         address;
  uint8_t                         speed;
  __IO uint8_t                    is_connected;
  uint8_t                         current_interface;
  USBH_DevDescTypeDef             DevDesc;
  USBH_CfgDescTypeDef             CfgDesc;

}USBH_DeviceTypeDef;
```

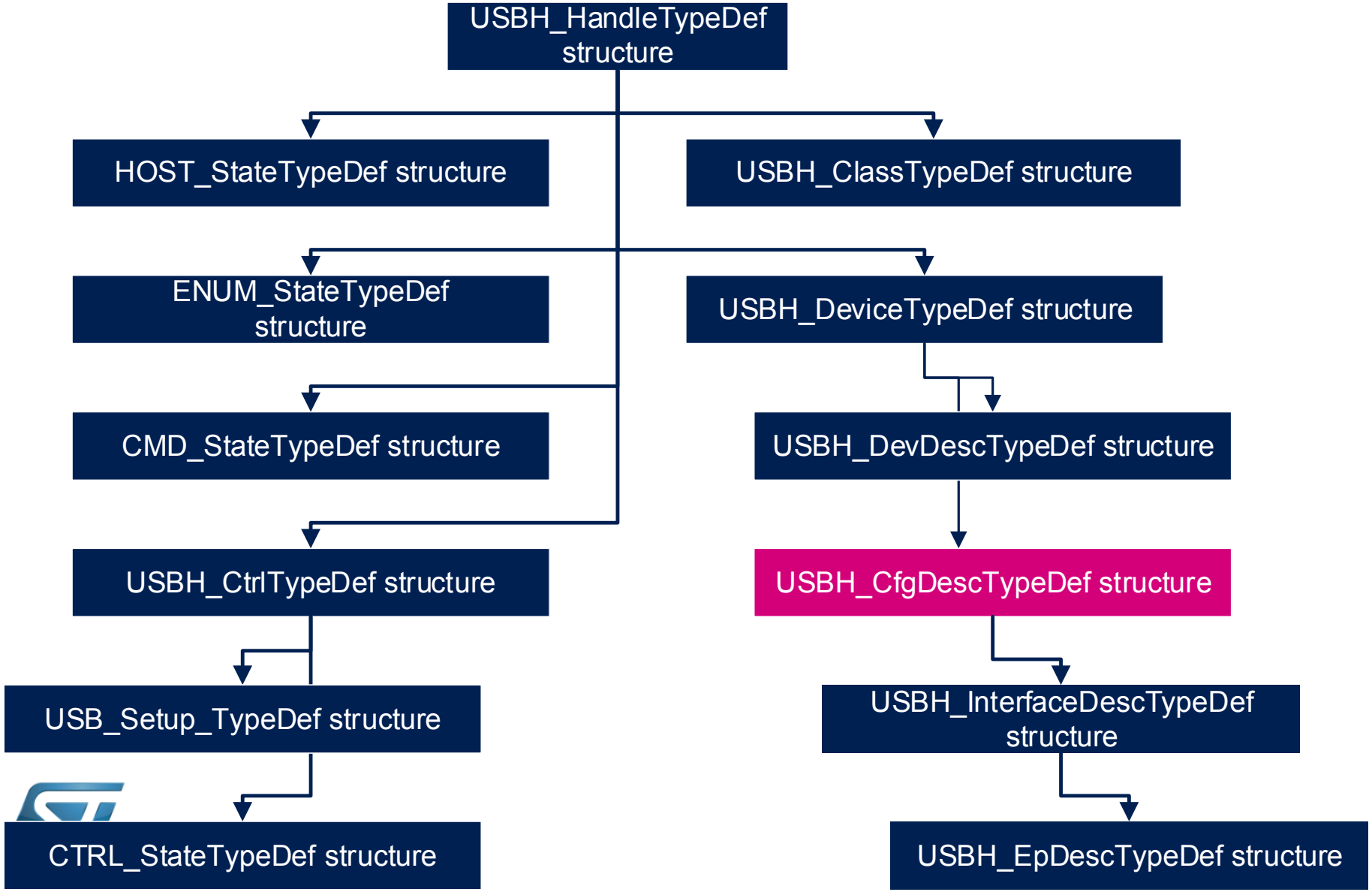
Device descriptor structure



Device descriptor structure


```
typedef struct _DeviceDescriptor
{
  uint8_t   bLength;
  uint8_t   bDescriptorType;
  uint16_t  bcdUSB;         /* USB Specification Number which device complies too */
  uint8_t   bDeviceClass;
  uint8_t   bDeviceSubClass;
  uint8_t   bDeviceProtocol;
  /* If equal to Zero, each interface specifies its own class
  code if equal to 0xFF, the class code is vendor specified.
  Otherwise field is valid Class Code.*/
  uint8_t   bMaxPacketSize;
  uint16_t  idVendor;       /* Vendor ID (Assigned by USB Org) */
  uint16_t  idProduct;      /* Product ID (Assigned by Manufacturer) */
  uint16_t  bcdDevice;      /* Device Release Number */
  uint8_t   iManufacturer;  /* Index of Manufacturer String Descriptor */
  uint8_t   iProduct;       /* Index of Product String Descriptor */
  uint8_t   iSerialNumber;  /* Index of Serial Number String Descriptor */
  uint8_t   bNumConfigurations; /* Number of Possible Configurations */
}
USBH_DevDescTypeDef;
```

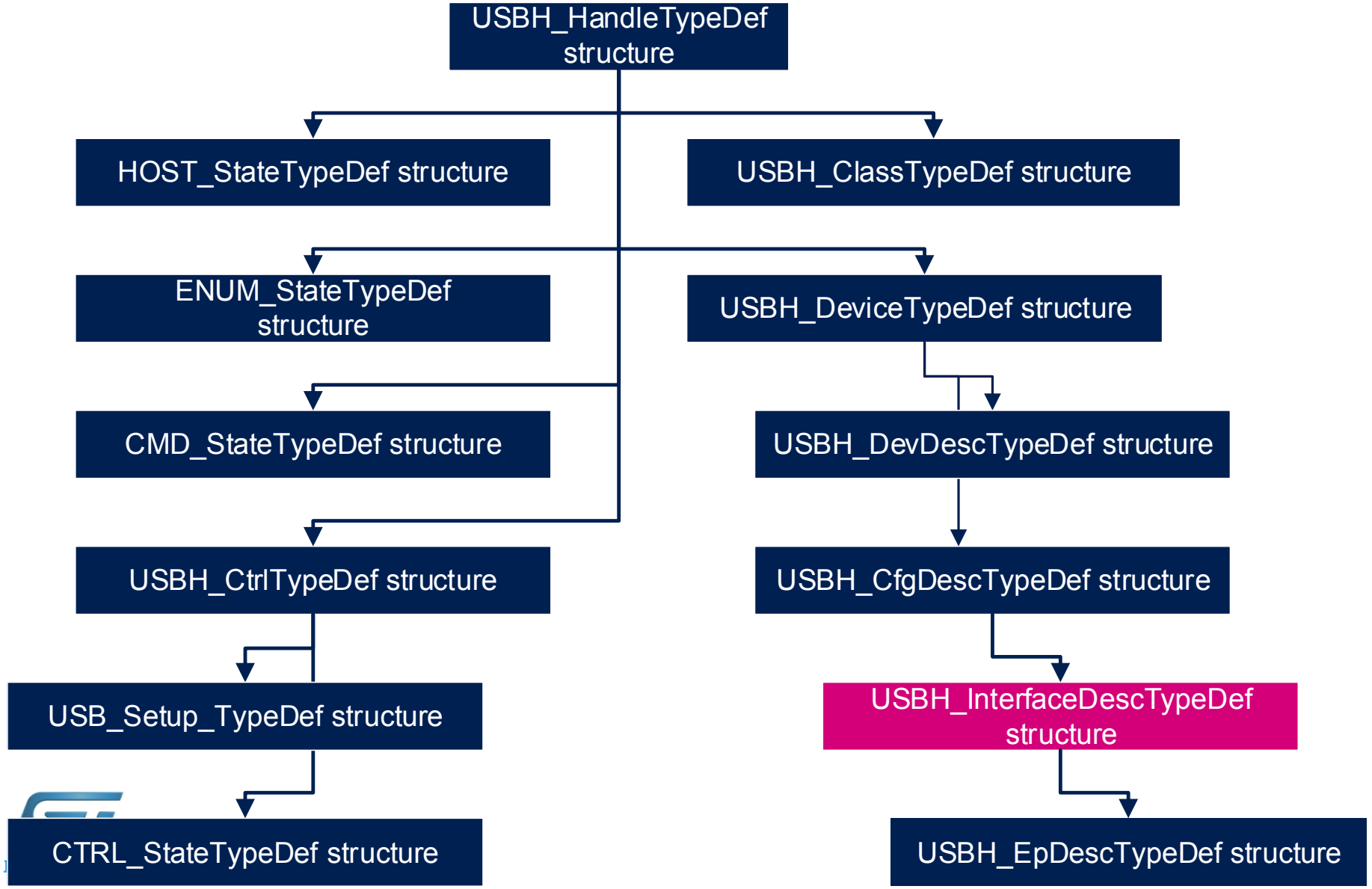
Device configuration descriptor structure



Device configuration descriptor structure


```
typedef struct _ConfigurationDescriptor
{
  uint8_t   bLength;
  uint8_t   bDescriptorType;
  uint16_t  wTotalLength;         /* Total Length of Data Returned */
  uint8_t   bNumInterfaces;        /* Number of Interfaces */
  uint8_t   bConfigurationValue;  /* Value to use as an argument to select this
configuration*/
  uint8_t   iConfiguration;        /*Index of String Descriptor Describing this
configuration */
  uint8_t   bmAttributes;          /* D7 Bus Powered , D6 Self Powered, D5 Remote Wakeup ,
D4..0 Reserved (0)*/
  uint8_t   bMaxPower;             /*Maximum Power Consumption */
  USBH_InterfaceDescTypeDef        Itf_Desc[USBH_MAX_NUM_INTERFACES];
}
USBH_CfgDescTypeDef;
```

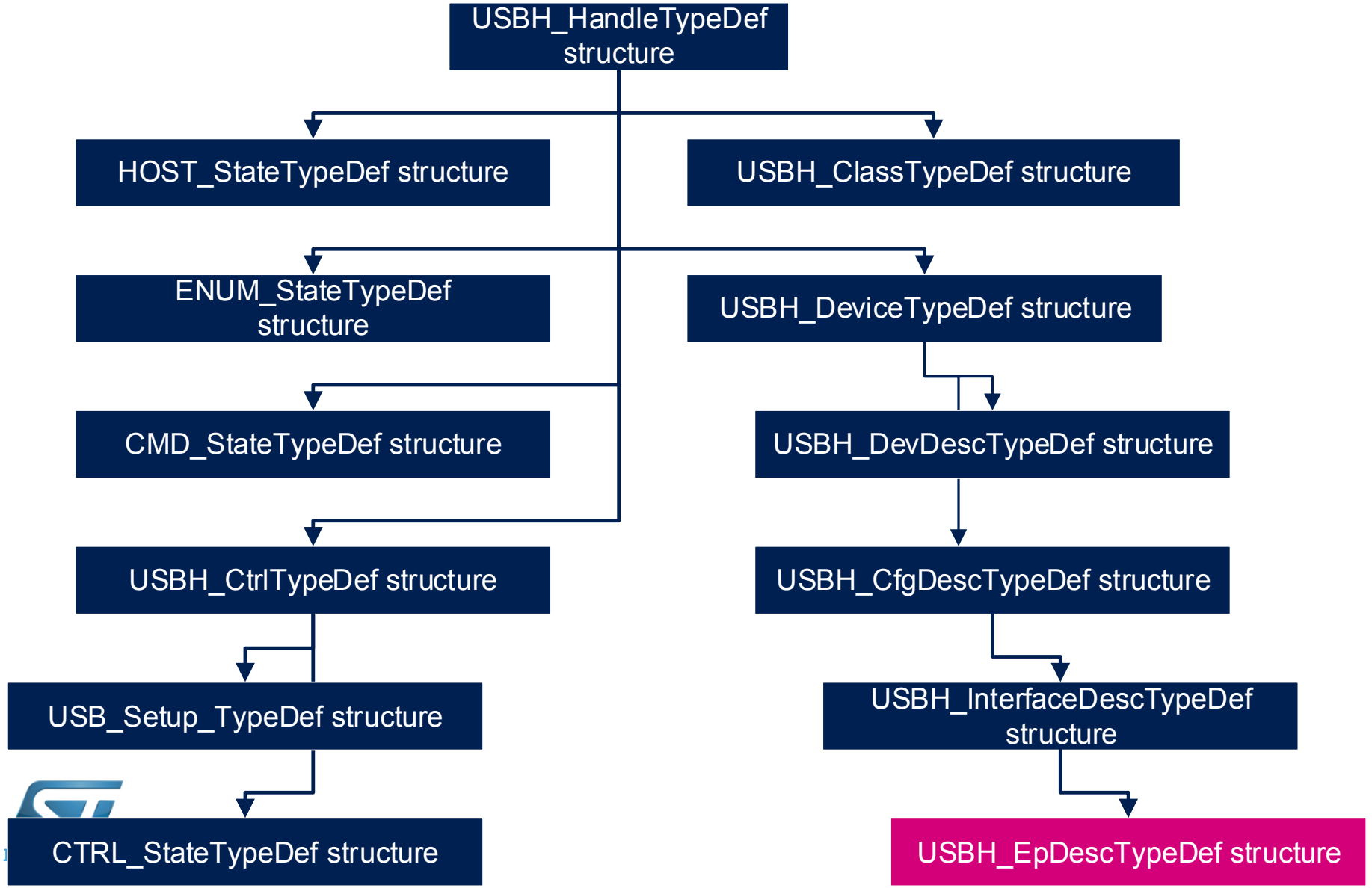
Device interface descriptor structure



Device interface descriptor structure


```
typedef struct _InterfaceDescriptor
{
  uint8_t bLength;
  uint8_t bDescriptorType;
  uint8_t bInterfaceNumber;
  uint8_t bAlternateSetting;     /* Value used to select alternative setting */
  uint8_t bNumEndpoints;         /* Number of Endpoints used for this interface */
  uint8_t bInterfaceClass;       /* Class Code (Assigned by USB Org) */
  uint8_t bInterfaceSubClass;    /* Subclass Code (Assigned by USB Org) */
  uint8_t bInterfaceProtocol;    /* Protocol Code */
  uint8_t iInterface;            /* Index of String Descriptor Describing this interface */
  USBH_EpDescTypeDef                Ep_Desc[USBH_MAX_NUM_ENDPOINTS];
}
USBH_InterfaceDescTypeDef;
```

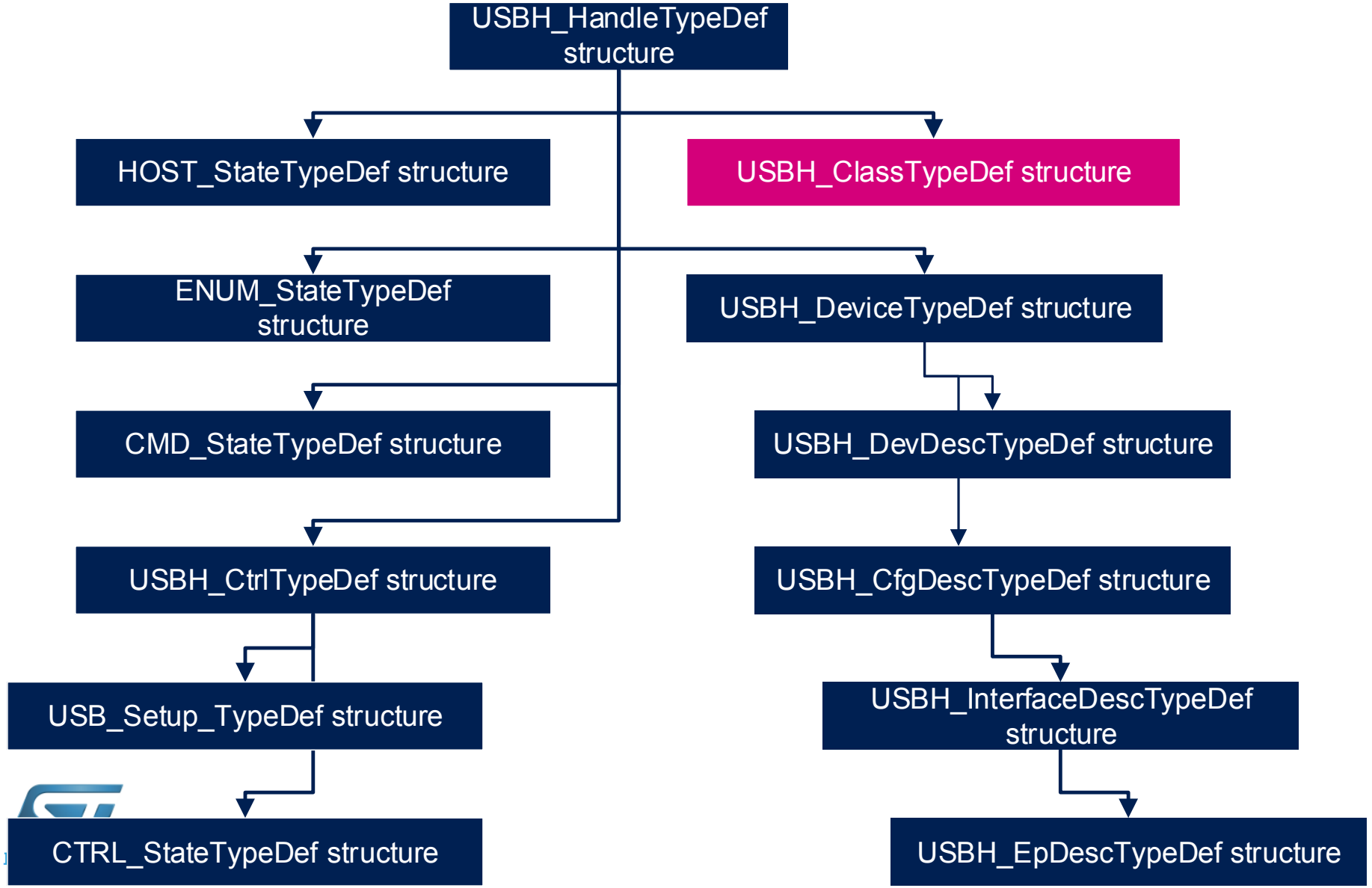
Device endpoint interface structure



Device endpoint interface structure


```
typedef struct _EndpointDescriptor
{
  uint8_t   bLength;
  uint8_t   bDescriptorType;
  uint8_t   bEndpointAddress;   /* indicates what endpoint this descriptor is describing
*/
  uint8_t   bmAttributes;       /* specifies the transfer type. */
  uint16_t  wMaxPacketSize;    /* Maximum Packet Size this endpoint is capable of sending
or receiving */
  uint8_t   bInterval;          /* is used to specify the polling interval of certain
transfers. */
}
USBH_EpDescTypeDef;
```

USB Host class structure

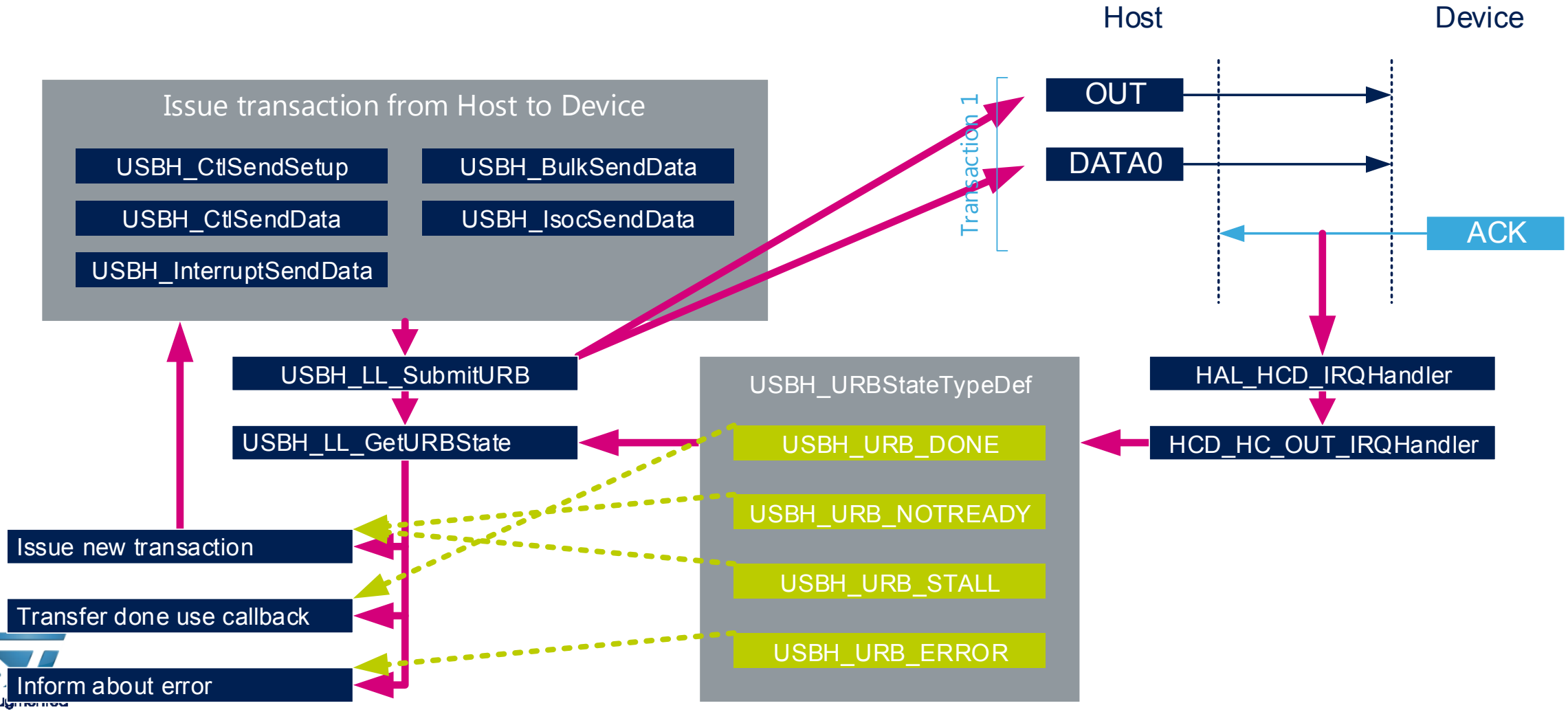


USB Host class structure

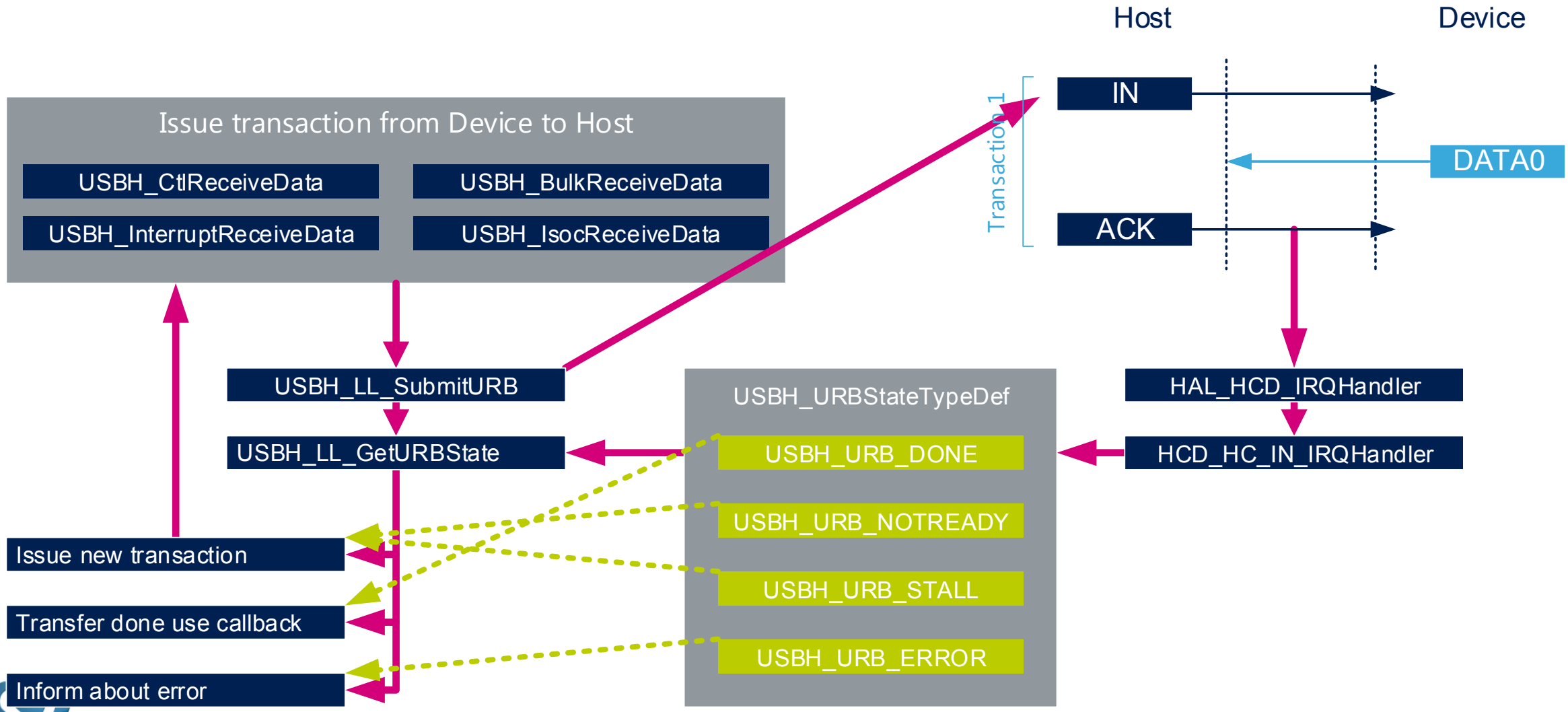

```
/* USB Host Class structure */
typedef struct
{
  const char           *Name;
  uint8_t               ClassCode;
  USBH_StatusTypeDef  (*Init)        (struct _USBH_HandleTypeDef *phost);
  USBH_StatusTypeDef  (*DeInit)      (struct _USBH_HandleTypeDef *phost);
  USBH_StatusTypeDef  (*Requests)    (struct _USBH_HandleTypeDef *phost);
  USBH_StatusTypeDef  (*BgndProcess) (struct _USBH_HandleTypeDef *phost);
  USBH_StatusTypeDef  (*SOFProcess) (struct _USBH_HandleTypeDef *phost);
  void*                 pData;
} USBH_ClassTypeDef;
```

Host Tx

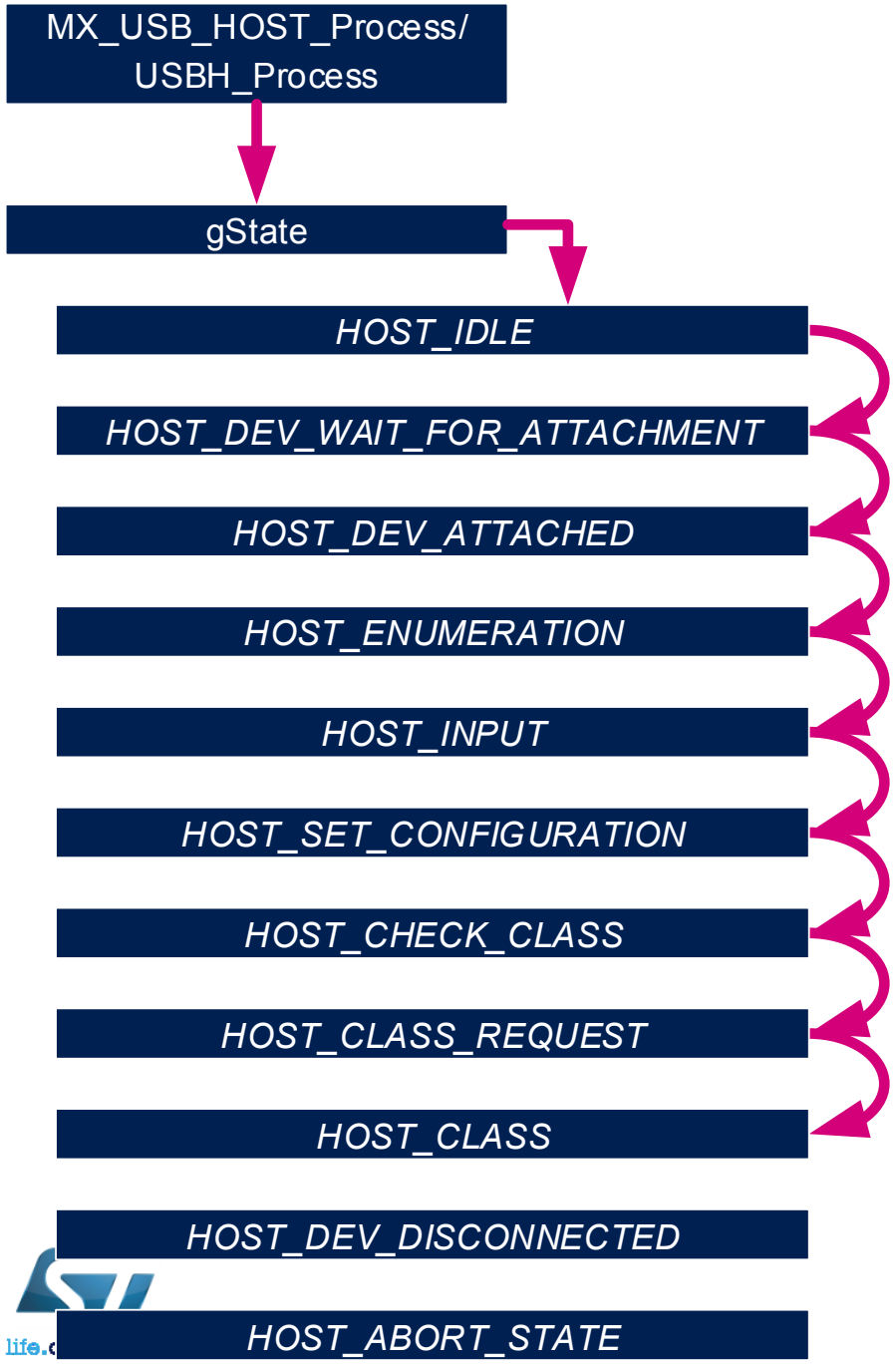
- Used in class or during enumeration



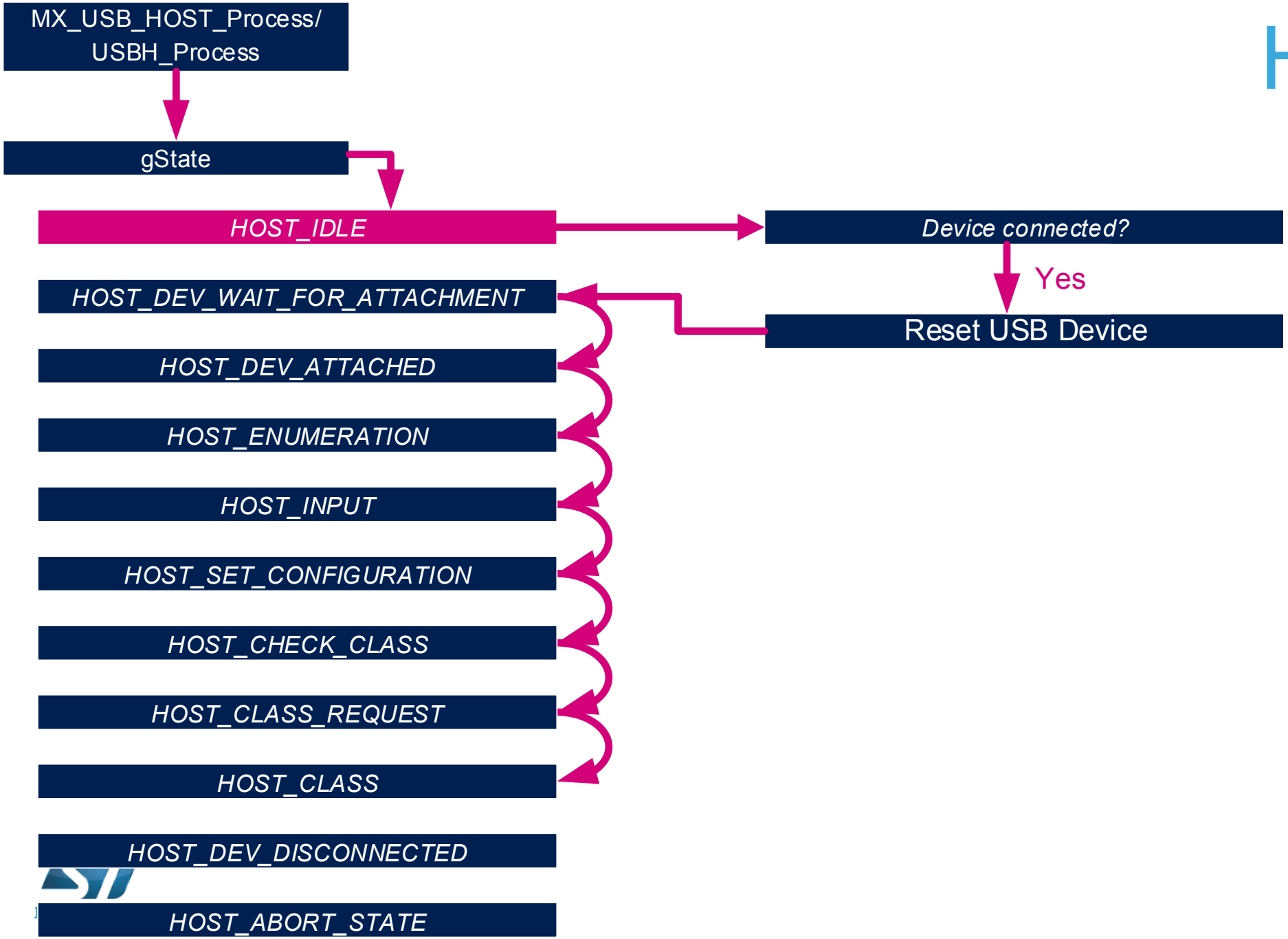
Host Rx



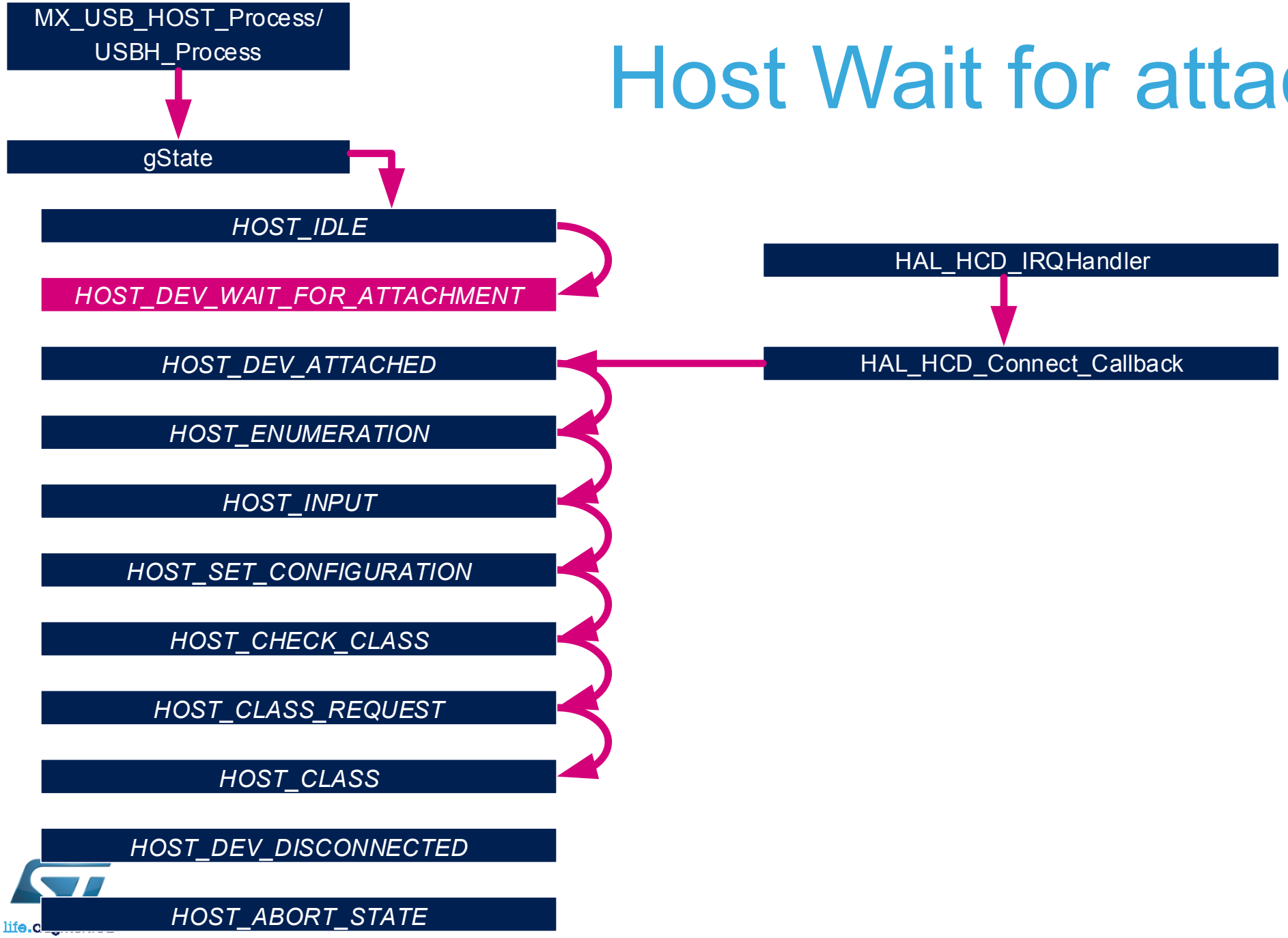
Periodic process



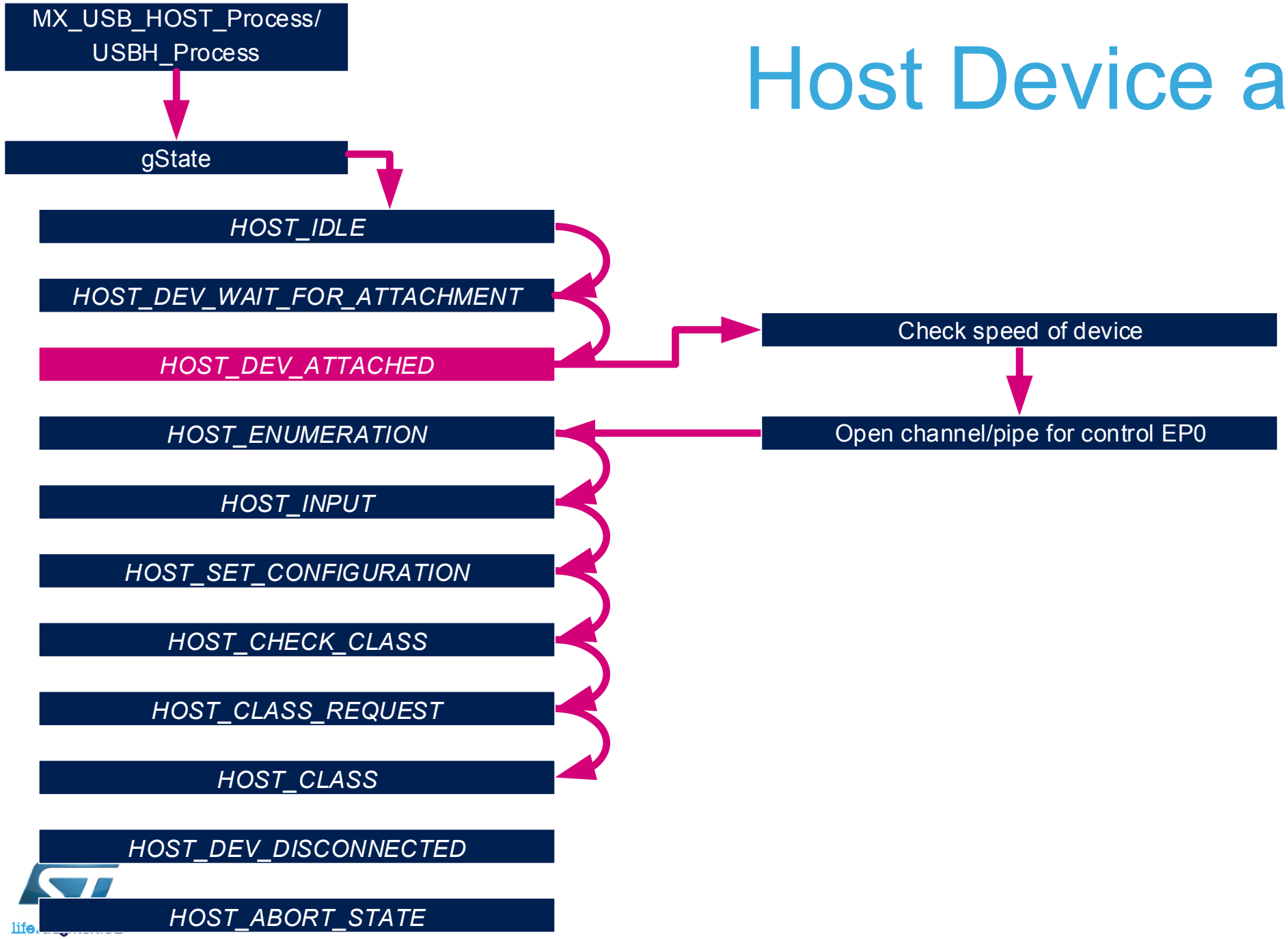
Host Idle


MX_USB_HOST_Process/
USBH_Process
gState
HOST_IDLE
Device connected?
Yes
Reset USB Device
HOST_DEV_WAIT_FOR_ATTACHMENT
HOST_DEV_ATTACHED
HOST_ENUMERATION
HOST_INPUT
HOST_SET_CONFIGURATION
HOST_CHECK_CLASS
HOST_CLASS_REQUEST
HOST_CLASS
HOST_DEV_DISCONNECTED
HOST_ABORT_STATE

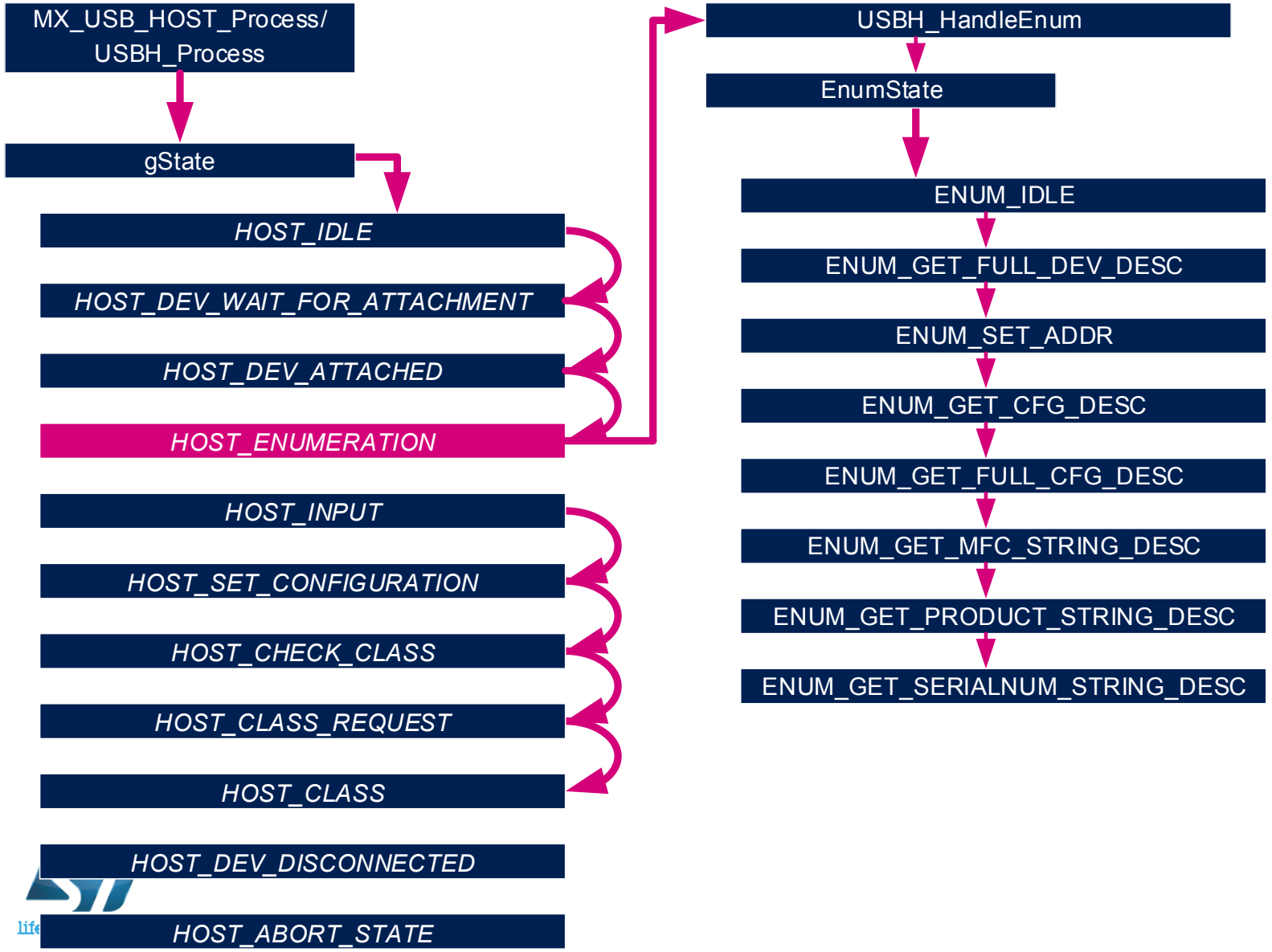
Host Wait for attachment



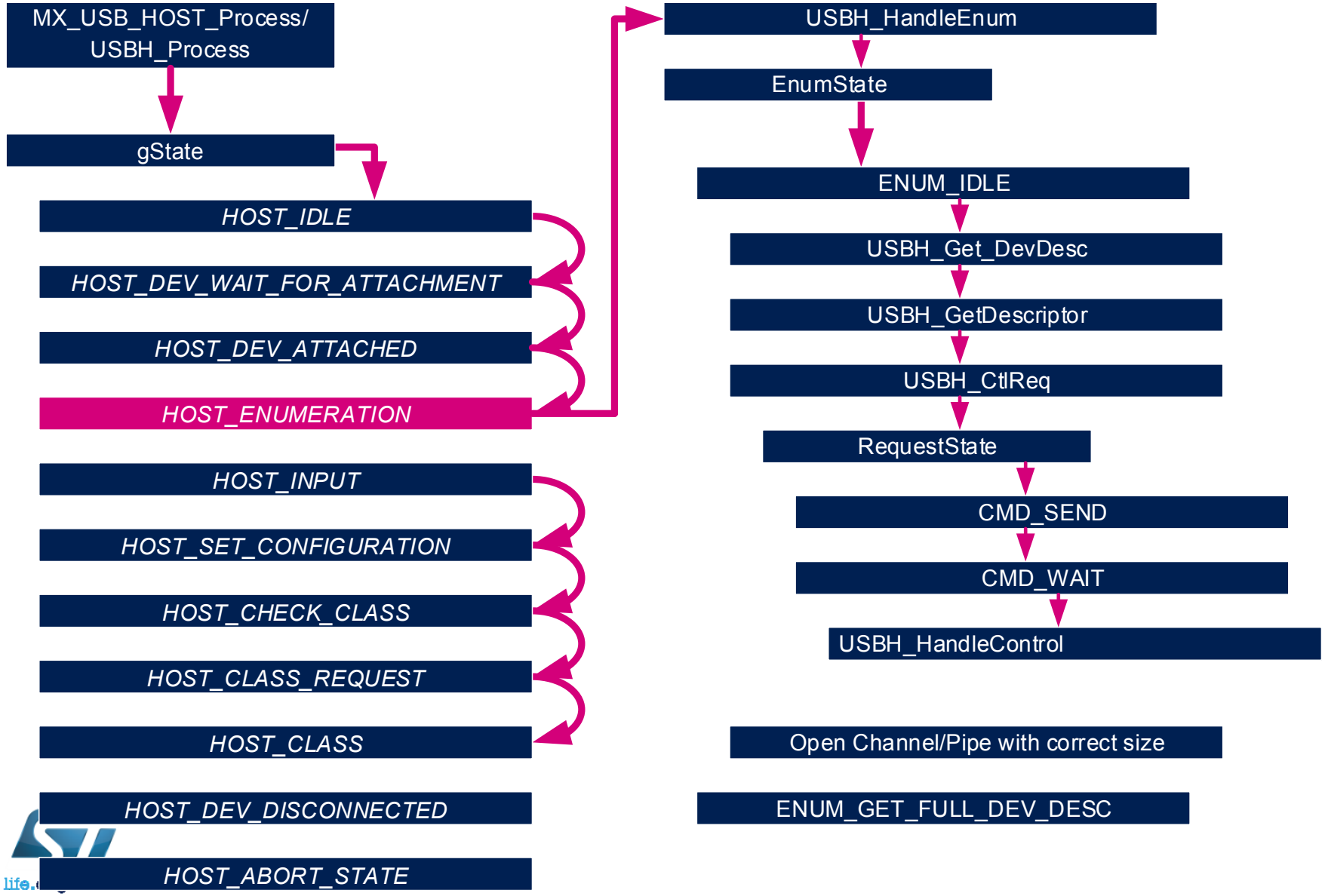
Host Device attached



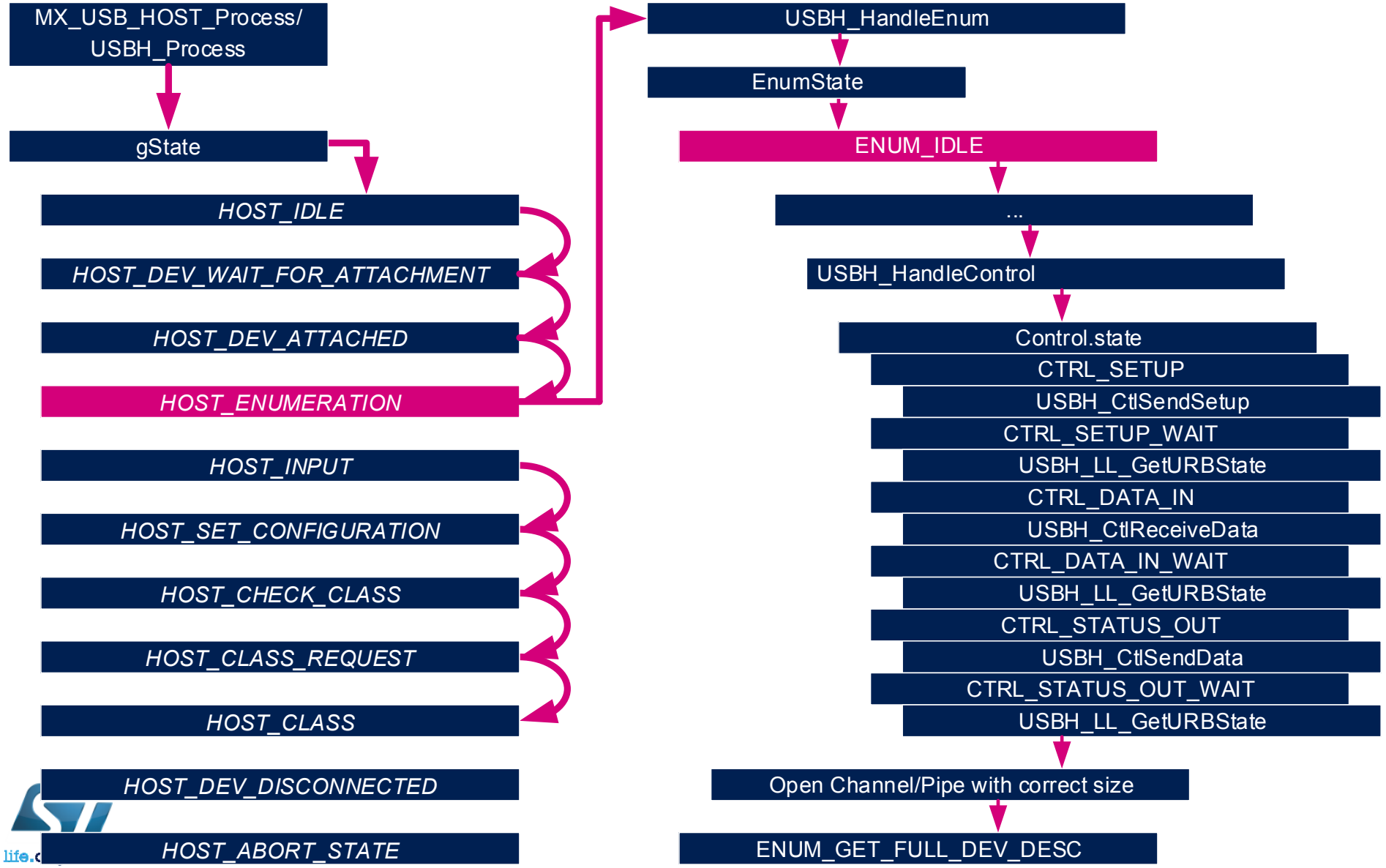
Host Enumeration



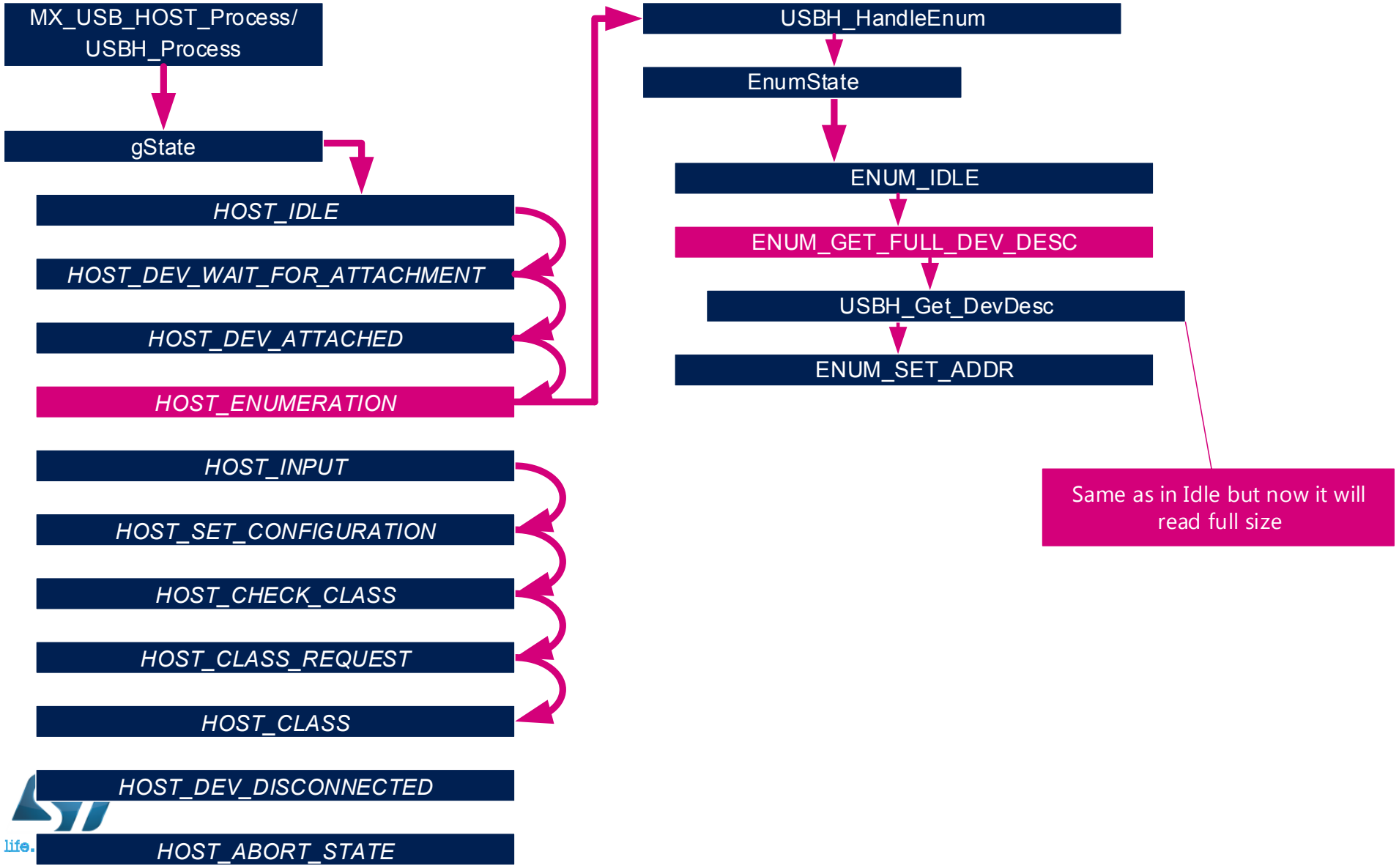
Host enumeration – Get Descriptor



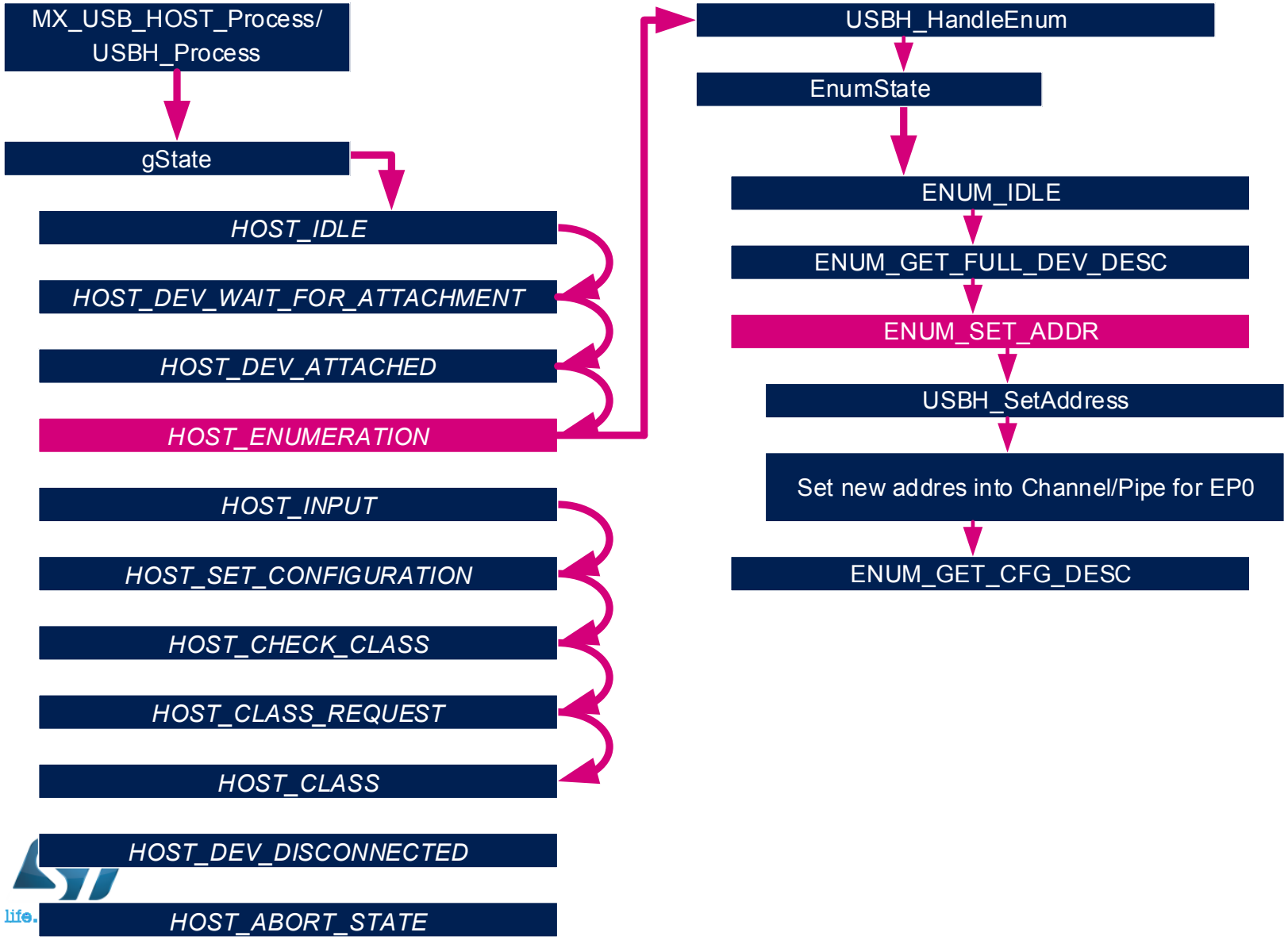
Host - Get Descriptor – Control transfer



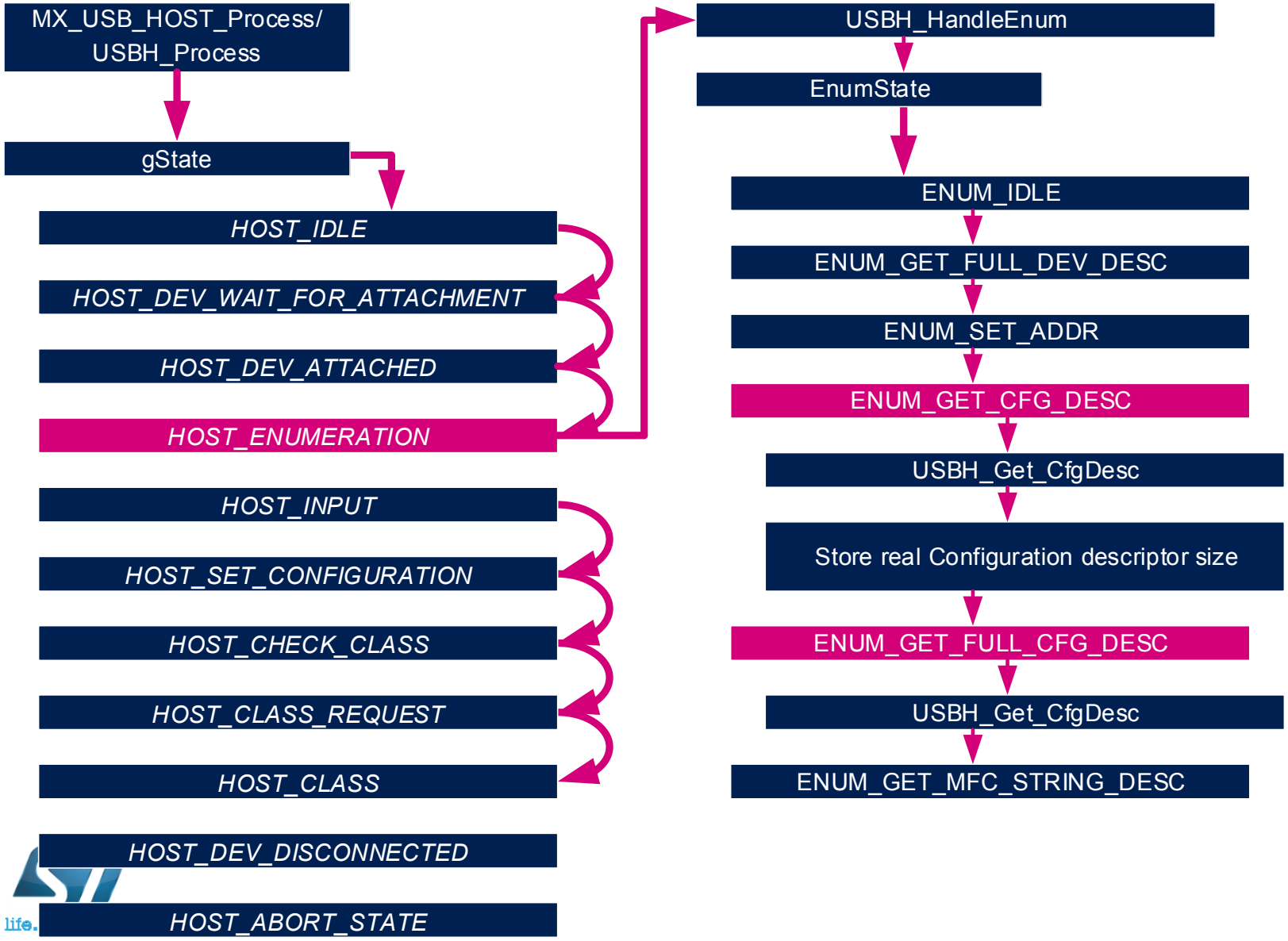
Host - Get full Device descriptor



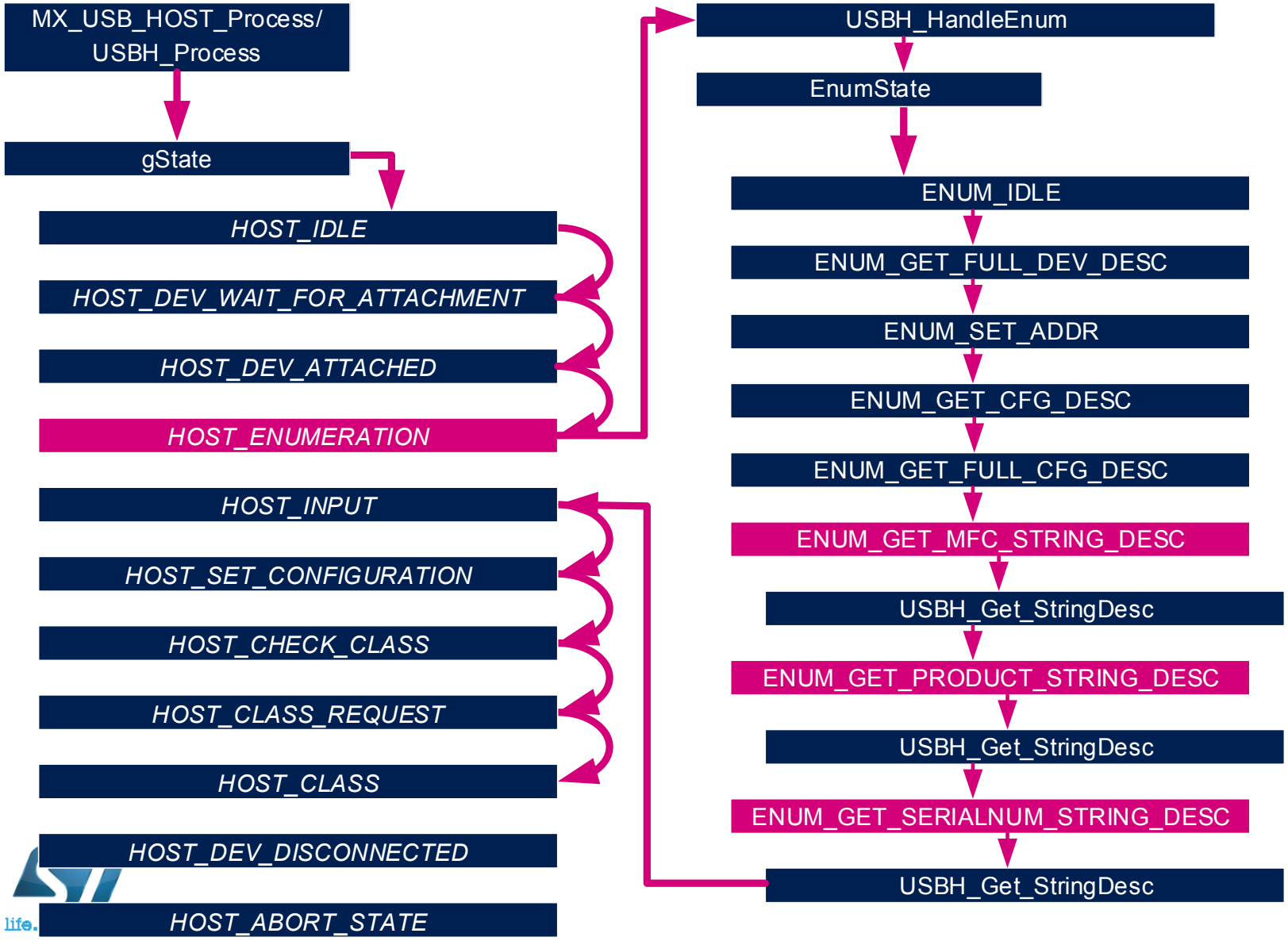
Host - Set device address



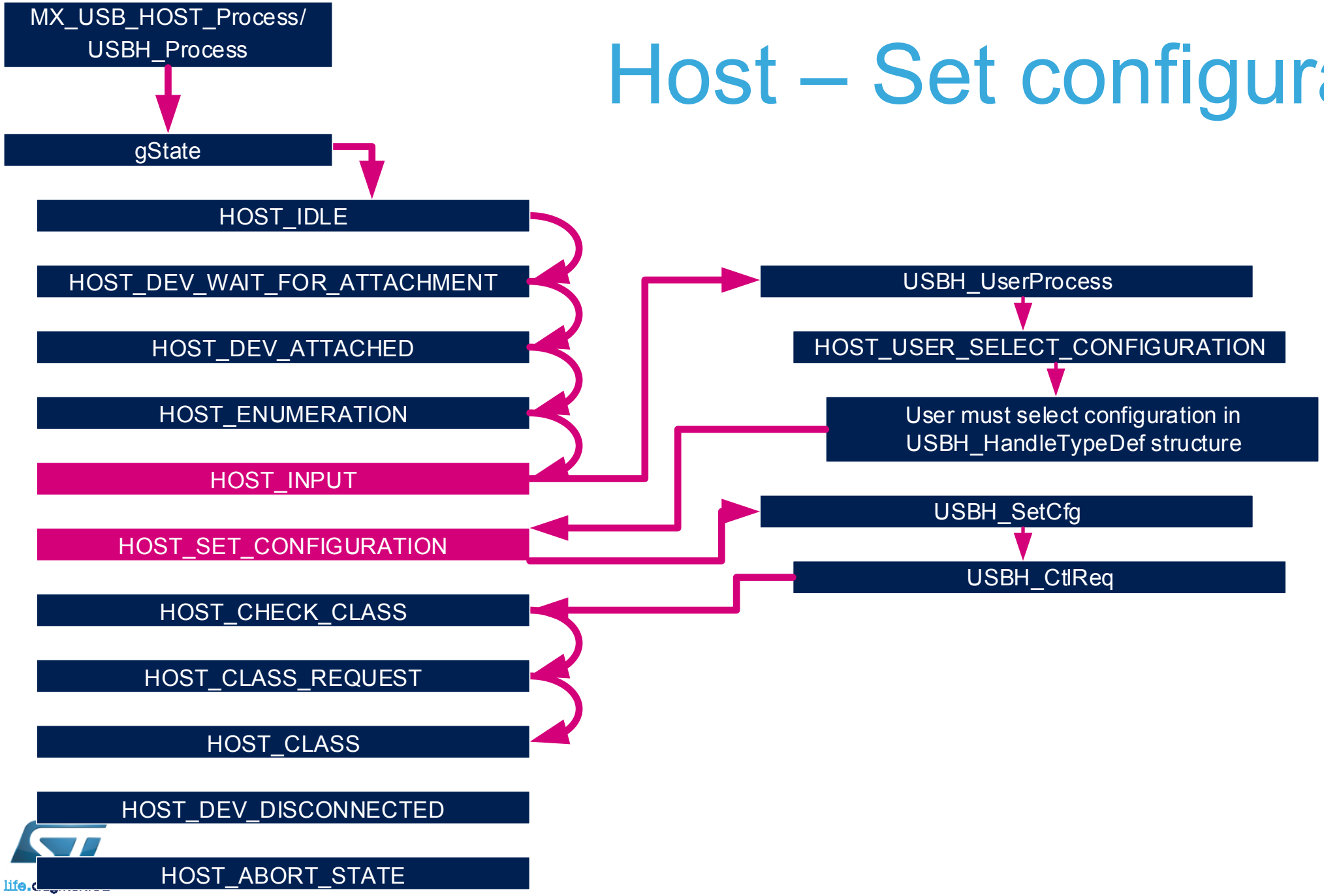
Host – Get configuration descriptor



Host Get string descriptors



Host – Set configuration



Host – Init class and Periodic class process