



# Developing USB applications using the STM32 ARM Cortex-M3 microcontroller

---

**Anis BEN ABDALLAH**  
**Embedded World 2011**



# Agenda

---

- Review of some important concepts in USB 2.0 standard
- USB device controller implementations in the STM32 microcontroller series
- Building blocks of a USB device application
- Overview about the STM32 USB device firmware library
- Overview about the USB Personal Healthcare Device Class (PHDC) and Continua™ ready ST stack
- ST PC software package for USB development

---

# Review of Important Concepts in USB 2.0 Standard





# USB Speeds and bus components

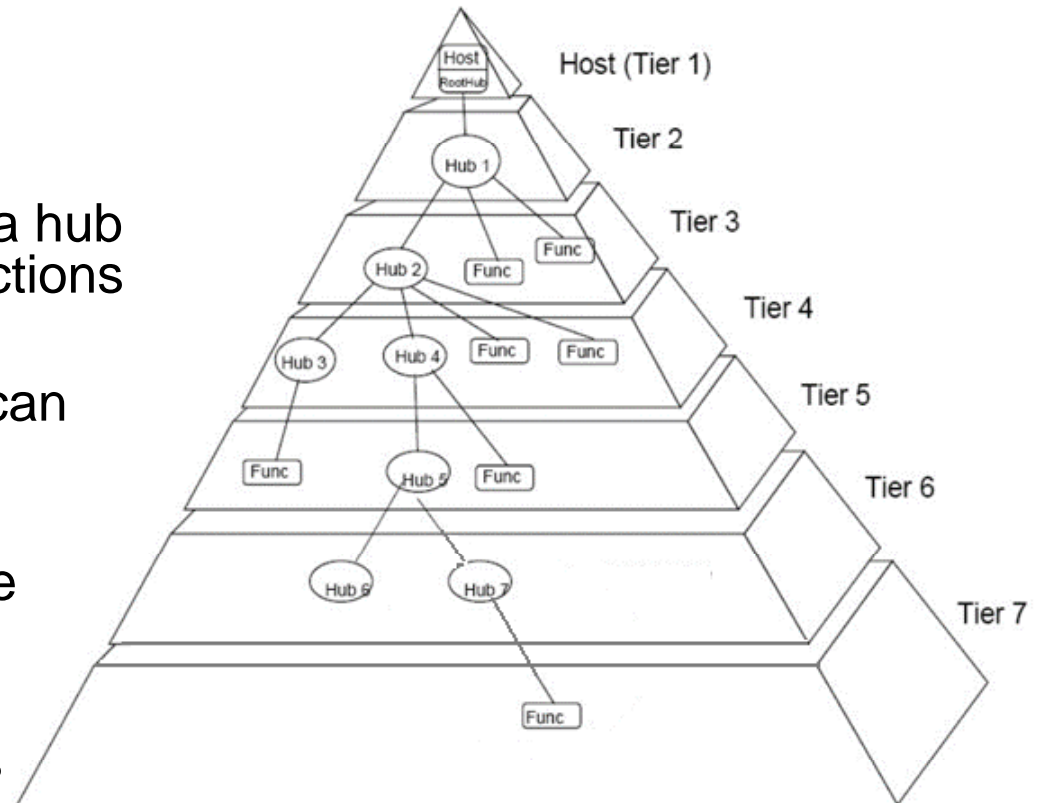
---

- USB 2.0 speeds
  - Low speed: 1.5 Mbits/s
  - Full speed : 12 Mbits/s
  - High speed: 480 Mbits/s
- USB keeps high compatibility at protocol level between all supported speeds
- Bus components
  - **USB host or Root hub:** initiates all the transaction on the bus
  - **USB function:** is a device with one or more interfaces that expose capabilities to the host (ex: mouse, keyboard,..)
  - **USB hub:** allows to connect multiple devices to the USB host. It has an upstream port for communication with the host and multiple downstream ports for direct connection to devices

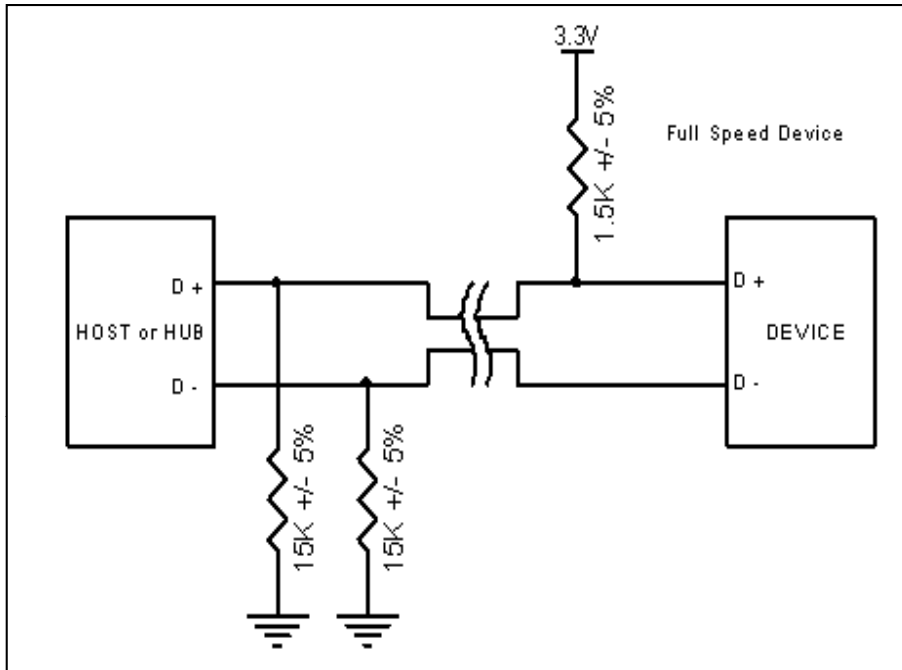
# USB Topology



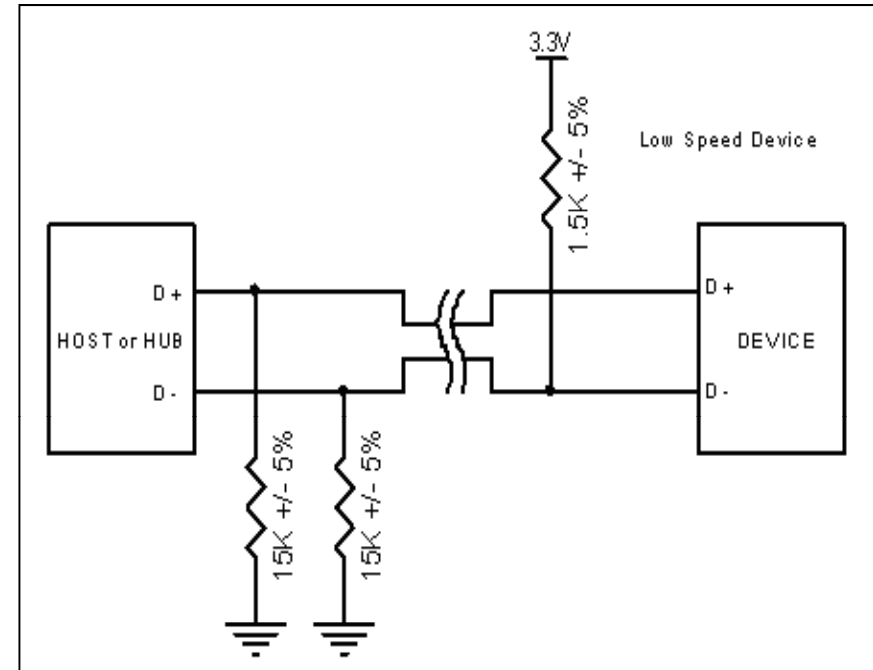
- USB bus has a Tiered Star topology
- At the center of each star is a hub with functions as end connections
- A maximum of 127 devices can be connected in the bus
- A maximum of 5 hubs can be connected in series
- the maximum cable length is 5meter



# USB Device attachment & speed detection



**Full/high Speed: Pull-up on D+**



**Low Speed: Pull-up on D-**

- The 1.5K pull-up allows the host to detect the device attachment and its supported speed
- High-speed device is detected first as full-speed device then high-speed capability is detected through bus handshake mechanism called “chirp sequence”



# USB Device Power

---

- Two possible power configurations
  - Self-powered: device power provided from external power-supply
  - Bus-powered: power provided from VBUS (5v)
- For bus-powered device, two options are possible:
  - **Low-power devices** :maximum power consumption is 100mA
  - **High-power devices** :maximum power consumption is 100mA during bus enumeration and 500mA after configuration
- During device enumeration, the device indicates to host its power configuration (self-powered/bus-powered) and its power consumption in the device configuration descriptor



# USB Suspend mode

---

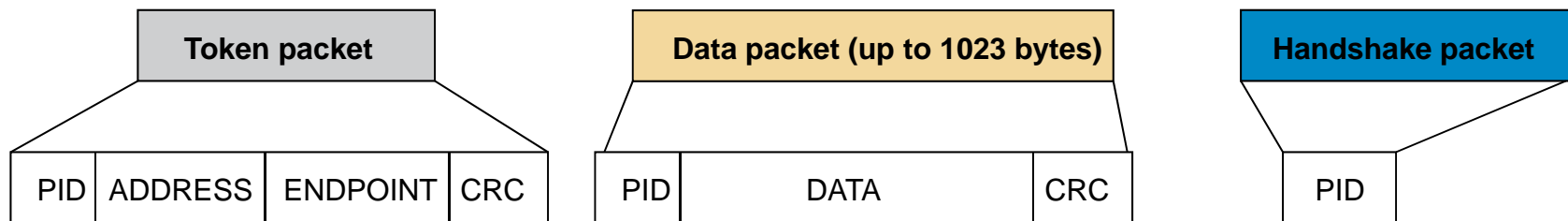
- USB device should enter in USB Suspend mode when the bus is in idle state for more than 3 ms
- In suspend mode, when the device is bus powered , the current drawn from VBUS power shouldn't exceed 2.5mA
- USB host prevents device from entering in suspend mode by periodically issuing Start of Frame (SOF) or Keep Alive for LS
  - For High-speed, SOF is sent every micro-frame: 125us +/- 65ns
  - For Full-speed, SOF is sent every frame: 1ms +/- 500ns
  - For Low-speed, Keep Alive (End of Packet) is sent every 1ms in absence of low-speed data
- Exist from Suspend mode can be
  - Initiated from host by issuing the resume signaling
  - Initiated from device by issuing the remote wakeup signaling



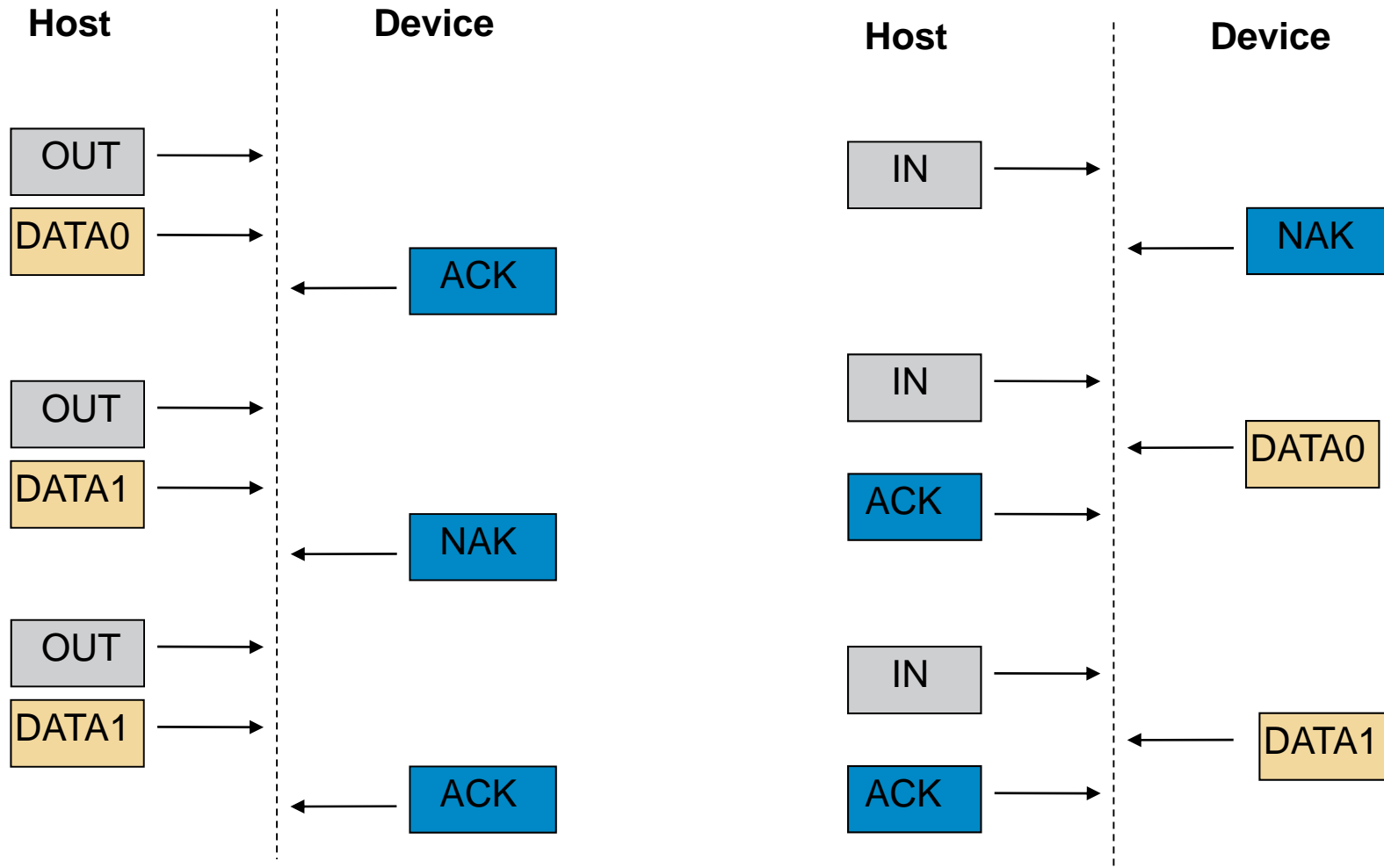
# USB Transaction



- **Token packet** (SETUP, IN, OUT) always issued by the host, includes:
  - PID (IN: Device to host data transaction or SETUP/OUT: host to device data transaction)
  - Target device address
  - Target endpoint number
  - CRC
- **Data packet** (DATA0, DATA1, DATA2, MDATA) includes:
  - PID: DATA0, DATA1, DATA2 or MDATA (DATA2 and MDATA are used only in HS mode)
  - Carries the data payload of a transaction sent by the host or device
  - DATA PID toggle used to synchronize HOST and DEVICE to avoid repeated packet transfer in case of corrupted or lost handshake
  - CRC
- **Handshake packet** (ACK, NAK, STALL, NYET)
  - ACK: packet reception acknowledged (sent from host or device)
  - NAK : packet reception not acknowledged (sent from device only)
  - STALL: control request not supported or endpoint halted (sent from device only)
  - NYET: device not ready to accept further packets (only for high-speed device)



# Examples of IN/OUT transactions





# USB Transfer

---

- A USB transfer is composed of one or multiple bus transactions
- Four types of USB transfers are defined:
  - **Control:** used for control and device configuration requests (ex: device enumeration)
  - **Bulk:** used for data transfers with no guaranteed delivery rate (ex: printer, mass-storage drive,..)
  - **Interrupt:** used for devices that need to be polled periodically for data transfers (ex: mouse, keyboard, joystick)
  - **Isochronous:** used for data streaming applications, that requires a guaranteed delivery rate, but no error checking (ex: audio, video devices)
- During each frame (in LS/FS) or micro-frame (in HS), the host will schedule the needed transfers with different bandwidth allocation for each transfer type



# USB Control Transfer

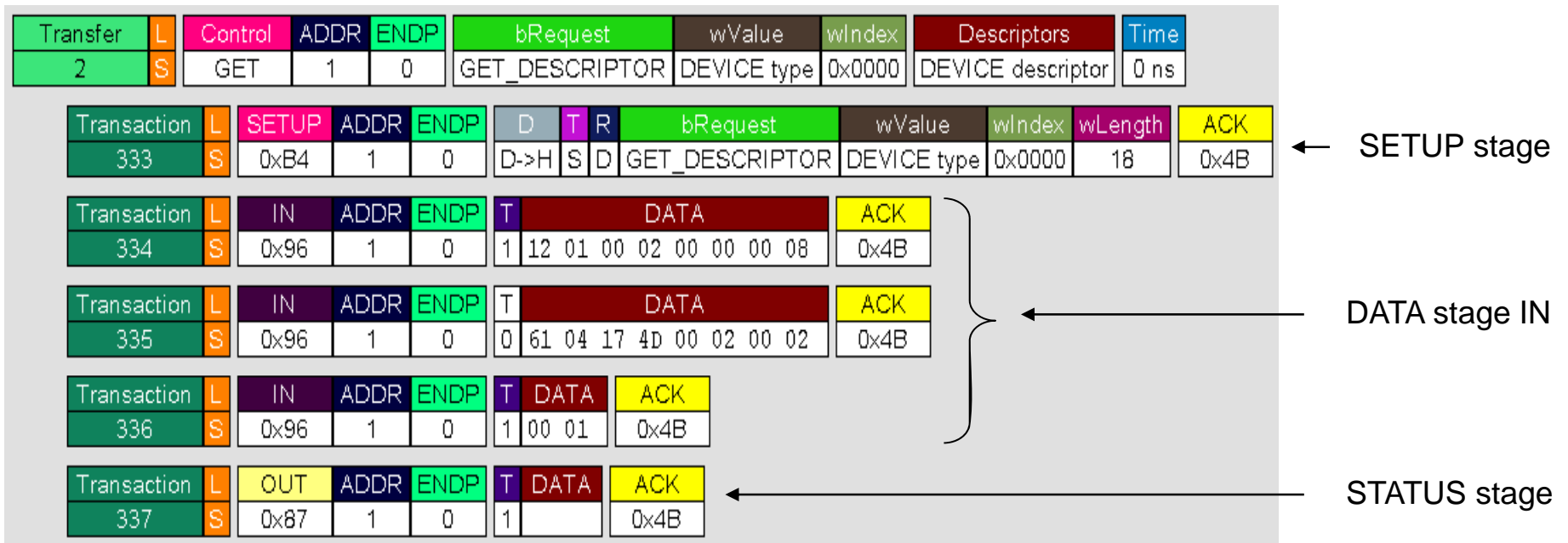
---

- Used for standard control requests during device enumeration process or during class operation
- All devices should support control transfer through endpoint 0 (bidirectional)
- It is given reserved bus bandwidth for **10% for FS/LS** and **20% for HS**
- Control transfer has 3 stages
  - **SETUP stage**: one SETUP transaction for issuing the control request (ex: Get Descriptor)
  - **Optional DATA stage IN or OUT**: one or multiple data transactions
  - **Status stage**: one IN or OUT transaction with a Zero Length data packet to check if control transfer request executed correctly or not.
- The maximum data packet size during the optional data stage is 8 bytes for LS and 64 bytes in FS/HS
- Transfer error management done through handshake packet and data PID toggle mechanism

# Example of a USB Control Transfer



- Get device descriptor standard request:





# USB Bulk Transfer

---

- Used to transfer large amount of data without guaranteed delivery rate (sending data to printer, drive,...)
- **Lowest priority transfer** with **no reserved bus bandwidth** but can occupy the full bandwidth if no other transfer on the bus
- Supported only by full-speed and high-speed devices
- Consist of one or more IN or OUT transactions during each frame/micro-frame (unidirectional)
- The max packet size is 64 bytes for FS and 512 bytes for HS
- Transfer error management done through handshake packet and data PID toggle mechanism



# USB Interrupt Transfer

---

- Interrupt transfers are used to poll devices to determine if they have data that needs to be transferred (mouse, keyboard,..)
- Interrupt IN or OUT data transfers are scheduled periodically within a maximum polling period negotiated during device enumeration but host is free to initiate more IN/OUT transactions if there is bandwidth available
- limited reserved bandwidth for Low/Full speed devices
  - For low-speed the packet max length is 8 bytes with a guaranteed maximum latency of up to 1 packet each 10 frames => 800 Bytes/s
  - For full-speed the packet max length is 64 bytes with a guaranteed maximum latency of up to 1 packet each frame => 62.5 KBytes/s
- High bandwidth with high-speed
  - For high-speed the packet max length is 1024 bytes with up to 3 packets each micro-frame
- Transfer error management done through handshake packet and data PID toggle mechanism



# USB Isochronous transfers

- Used mainly for **streaming real-time data** like audio and video
- Needs a **guaranteed bandwidth** with a **constant transfer rate** but there is **no error checking**
- The requested bandwidth is negotiated between host and device during enumeration
- Transfer is in one direction and can consist of one or more data OUT or IN transactions with no handshake packet
  - In FS, the max packet length is 1023 bytes with a maximum of one packet per frame
  - In HS, the max packet length is 1024 bytes with a maximums of 3 packets per micro-frame
- Devices that use isochronous transfer need in most of the cases to establish a **synchronous** connection (ex: speaker, microphone, video camera,...)
  - Minimal or no data buffering
- The synchronization between the data source (producer) and the sink (consumer) can be achieved by
  - Having the source and sink clocks synchronized to the SOF packet
  - Doing a clock adaptation either on the source using a dedicated feedback pipeline or on the sink clock based on the received data rate



# Host constraints for Interrupt & Isochronous Transfers

---



- The host may not be able to provide the requested bandwidth to device, in this case the host will try other possible configurations with lower bandwidth requirements (if provided by the device)
- If still no bandwidth available, the host will refuse device configuration
- Host software may have some latency for processing data and issuing transfer requests on time due to other processes taking CPU time
- In order to avoid multiple SW calls for handling data to be transmitted or received, large chunks of data transfers should be scheduled

# USB High-Speed mode specific features

## PING/NYET Protocol

---



- For control and bulk transfers, when a high-speed device is not ready to receive further data OUT packets, it can send the NYET handshake
- When the host receives the NYET handshake, it should send the PING packet periodically to check if device is ready or not to resume receiving data packets
- When ACK is received for a PING request, the host will resume sending data packet

# USB High-Speed mode specific features

## SPLIT Protocol

---



- The SPLIT protocol is used when the HS host need to communicate with a low/full speed device which is connected to a high-speed hub
- The host will do the data transaction with the HS hub in high-speed, then the hub as a host will initiate the same transaction in low/full speed with the device
- The data transaction done by the host with the HUB is preceded with the Start SPLIT (SSPLIT) token
- The host will later use CSPLIT token to retrieve the device response from the HUB

# USB controllers in the STM32 microcontroller series





# USB Device Controllers in STM32 series

---

- USB device controller is present in almost all STM32 ARM Cortex-M3 series
- Three hardware implementations are available
  - USB 2.0 full-speed device controller
  - USB 2.0 full-speed OTG dual role host/device controller
  - USB 2.0 high-speed OTG dual role host/device controller
- Selection of the controller that can fit the application needs will depend on
  - Needed USB transfer performance
  - Needed CPU performance
  - Available Flash and RAM memory size
  - Presence of other needed peripherals
  - Power consumption requirements
  - External components (BOM)

# USB Device Controller in STM32 series

---



## USB 2.0 Full-speed Device Controller



# USB 2.0 Full-speed Device Controller Features

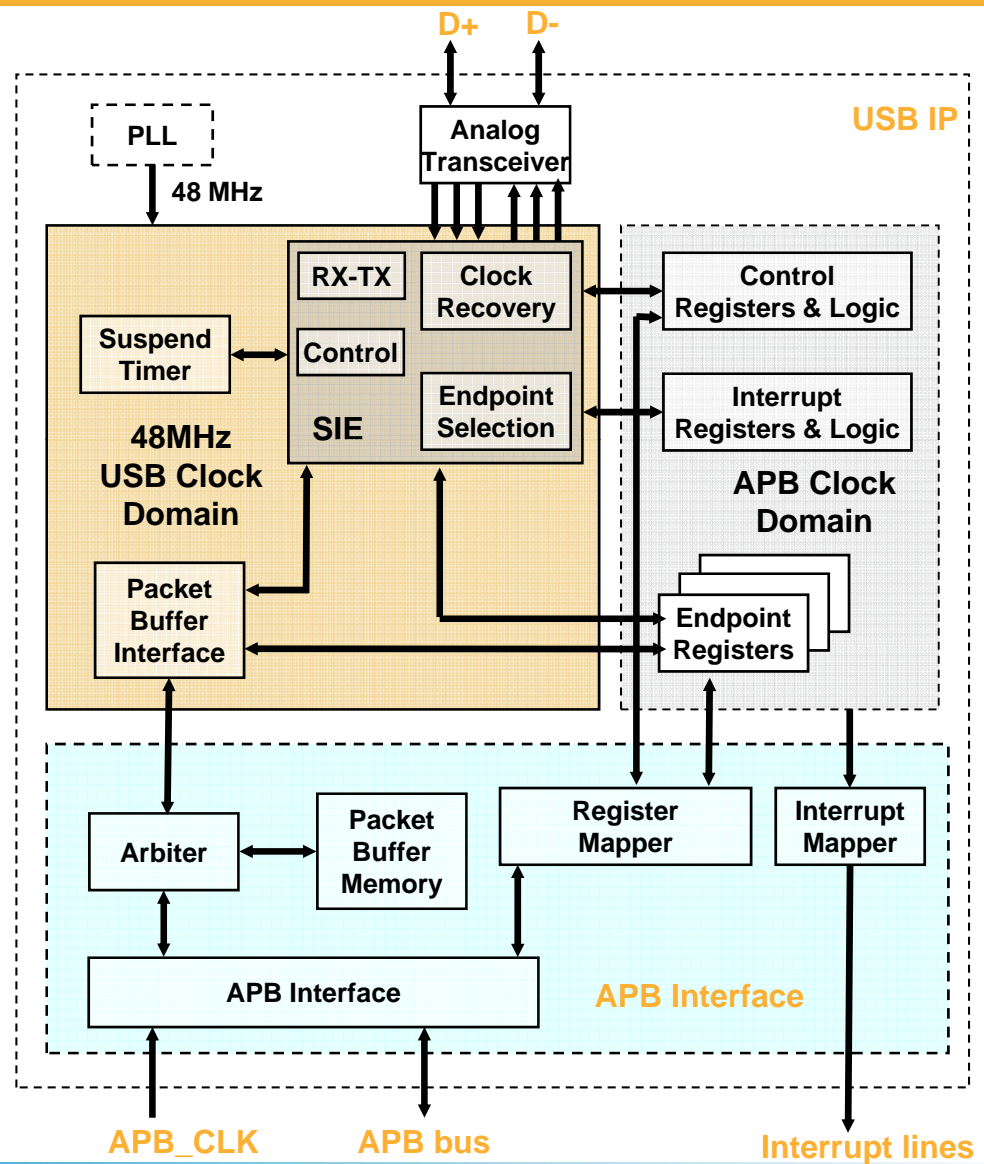


- Available on the following ARM Cortex-M3 platforms:
  - **STM32F102: USB access line** (48 MHz MCU, up to 16KB SRAM and 128KB of FLASH )
  - **STM32F103: Performance line** (72 MHz MCU, up to 96KB SRAM and 1MB FLASH)
  - **STM32L152: Ultra-low power series** (32 MHz MCU, up to 16KB SRAM and 128KB of FLASH)
- Main features
  - USB 2.0 full-speed compliant
  - Up to 8 bi-directional endpoints (or 16 unidirectional endpoints)
  - Embedded full-speed analog transceiver
  - Supports all transfer modes (control, bulk, interrupt and isochronous)
  - Dedicated SRAM area of 512 bytes as packet memory that can be shared among the needed endpoints
  - Double-buffering mechanism for isochronous and bulk transfers
  - USB Suspend/Resume with system entry/wakeup for low power mode

# USB 2.0 Full-speed Device Controller Block Diagram



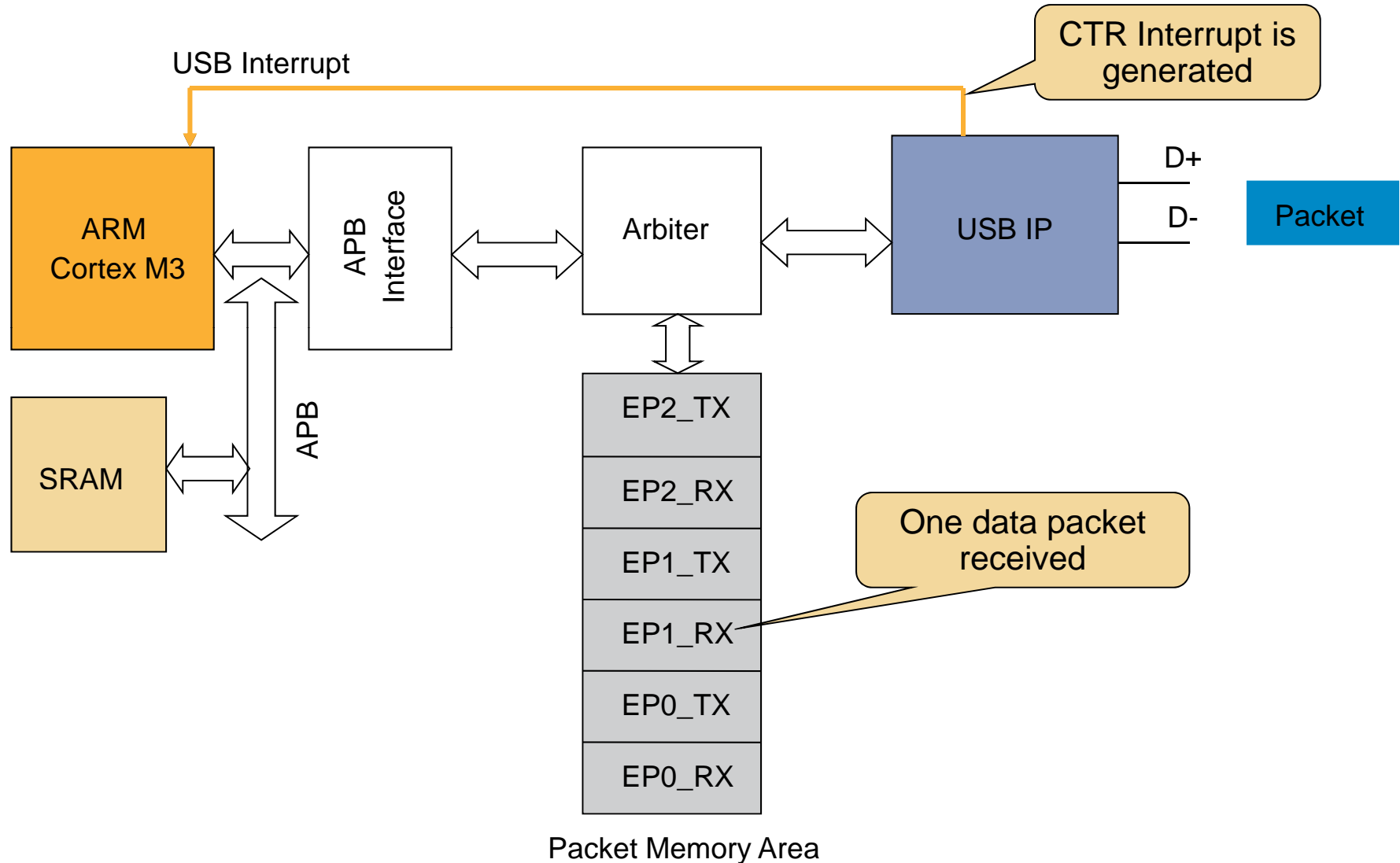
- **SIE (Serial Interface Engine)**
  - NRZI Encoding/Decoding
  - Synchronization & Pattern Recognition
  - Bit-stuffing and Handshake evaluation
  - PID & CRC generation and checking
  - Interrupt generation
- **Suspend Timer**
  - Generate the Suspend interrupt when no SOF is detected for 3ms
- **Packet Buffer Memory**
  - 512 bytes dedicated SRAM memory
  - The Arbiter allows dual access either from packet buffer interface or APB interface
- **3 interrupt vectors (lines)**
  - Low priority interrupt for managing all endpoints
  - High priority interrupt: can be used for managing isochronous/double-buffered endpoints only
  - Suspend/Resume interrupt





# USB 2.0 Full-speed Device Controller

## Operation overview



# USB 2.0 Full-speed Device Controller

## Transactional model handling

---



- After each successful transaction on any configured endpoint, an interrupt (correct transfer CTR) is raised
- The “Correct transfer” interrupt handler has to:
  - Check interrupt status bits to determine the endpoint on which the transaction has occurred
  - For **OUT/SETUP endpoints**: copy received data packet from packet memory area to application buffer for processing, then re-enable the endpoint to be able to receive next incoming packet
  - For **IN endpoints**: copy next data to be transferred from application buffer to packet memory area, then re-enable the endpoint to send the packet when the next IN token comes from host
- The hardware will automatically change the endpoint to NAK state after end of each transaction, so it is up to application to re enable endpoint for next transaction
- The Transactional model has simple FW handling, but does not allow multiple-packet transfer without CPU intervention after each transferred packet

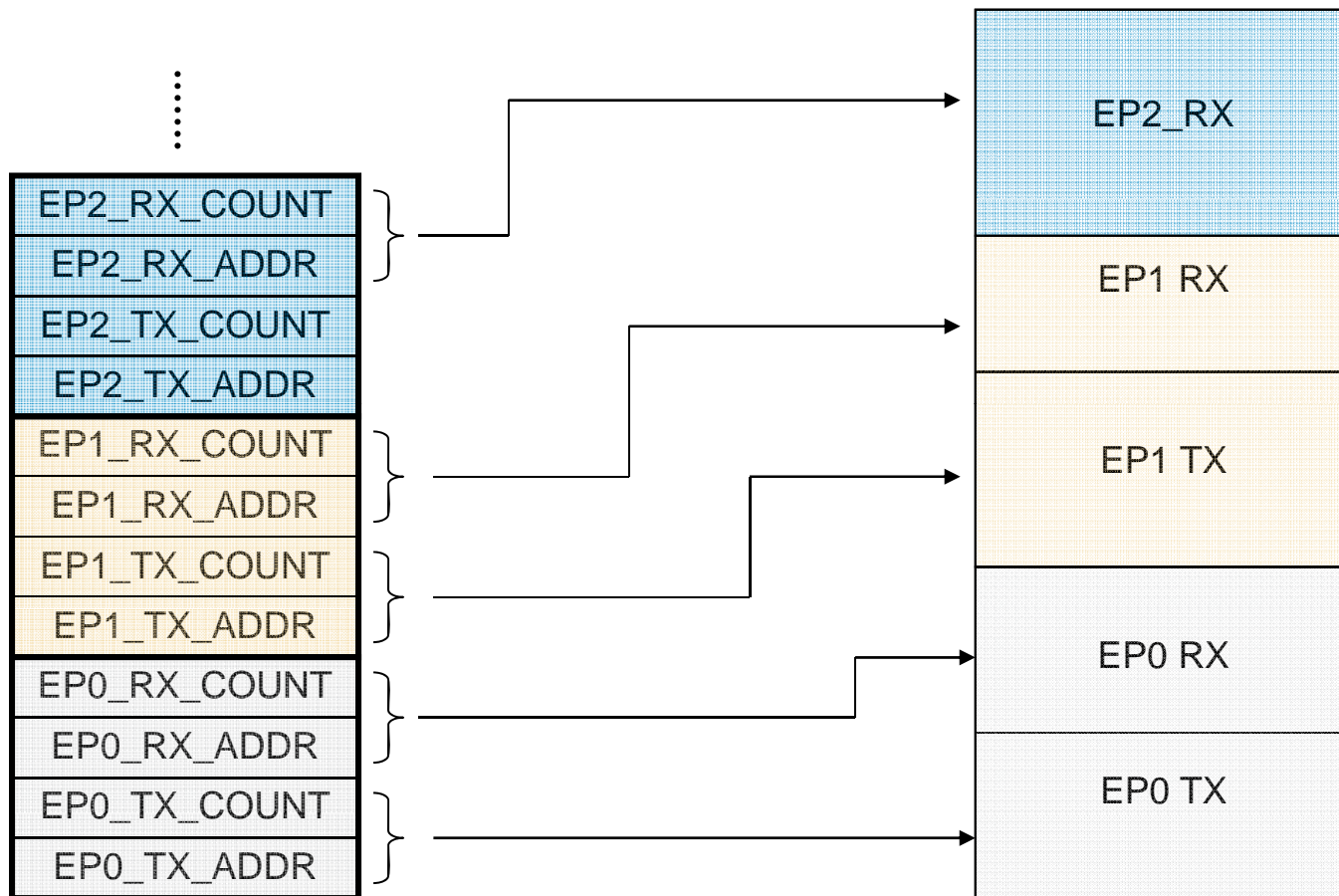
# USB 2.0 Full-speed Device Controller Endpoint Configuration/Enabling



- Before start of any transfer on one endpoint, the following configuration should be done:
  - Endpoint address (only lower four bits)
  - Endpoint transfer type (control, bulk, interrupt or isochronous)
  - Endpoint TX or RX packet start address location in the packet memory area
  - For OUT/SETUP endpoints the max receive packet size should be configured
  
- After the configuration, endpoint can be enabled for a transfer
  - **IN endpoint:**
    - Data can be copied from application buffer to endpoint PMA buffer
    - the TX transfer count should be updated (the maximum is one max packet size)
    - Endpoint status should be changed to “ACK” to allow data transfer when IN token arrives
  - **OUT/SETUP endpoint:**
    - Endpoint status should be changed to “ACK” to allow OUT/SETUP data packet reception on endpoint

# USB 2.0 Full-speed Device Controller

## Packet Memory Area



Buffer Description Table (BTABLE)

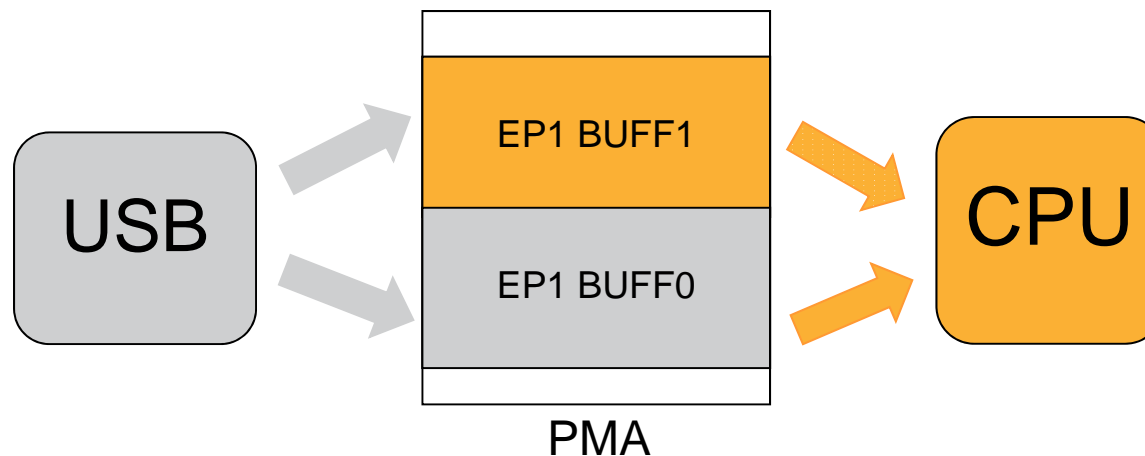
Packet Memory Area

# USB 2.0 Full-speed Device Controller

## Double-Buffering mechanism

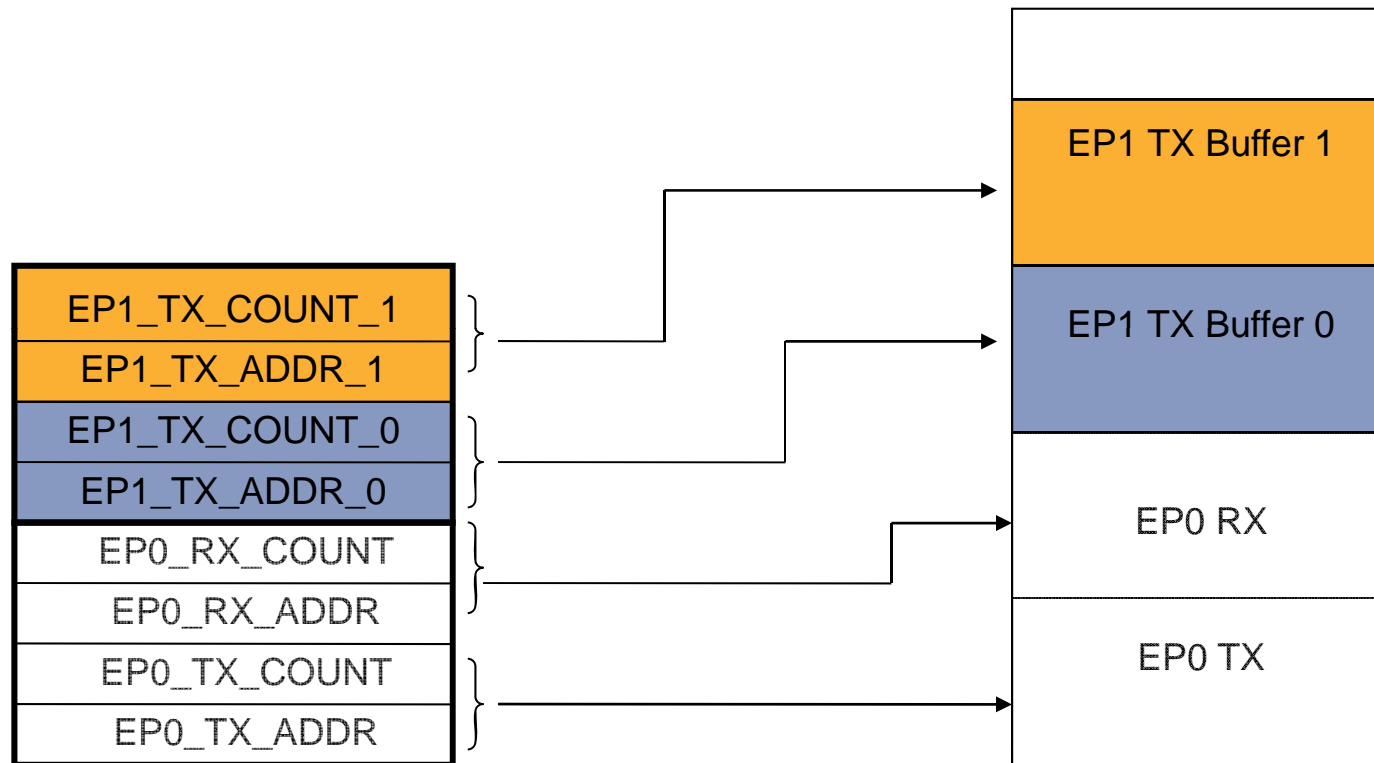


- Double buffering is used to improve the transfer performance for isochronous and bulk endpoints (in one direction only)
- Consists of using two buffers in PMA (buffer0 and buffer1), at any time CPU should be accessing one buffer (for R/W) while USB IP is accessing the other buffer
- USB swapping between buffer0 and buffer1 is done by hardware
- In double-buffered bulk transfer, If application (CPU) is too slow to give its buffer to USB, then NAK will be sent to host



# USB 2.0 Full-speed Device Controller

## Packet Memory Area with double-buffering



Buffer Description Table (BTABLE)

Packet Memory Area

# USB 2.0 Full-speed Device Controller

## Suspend/Resume Interrupt

---



- When no SOF is detected for 3 ms, a suspend interrupt is generated
- In the interrupt handler of the suspend interrupt, if bus powered device, the MCU should enter in low power mode in order to lower its power consumption
- In order to achieve the best low power consumption, the STM32 can enter in STOP mode (all peripherals and CPU clocks OFF)
- A host resume/reset signaling detection can wakeup the MCU from STOP mode
- The device can also initiate a bus resume or “remote wakeup” using external interrupt that can wakeup MCU from STOP mode

# USB Device Controller in STM32 series

---



## USB 2.0 Full-Speed OTG Host/Device Controller



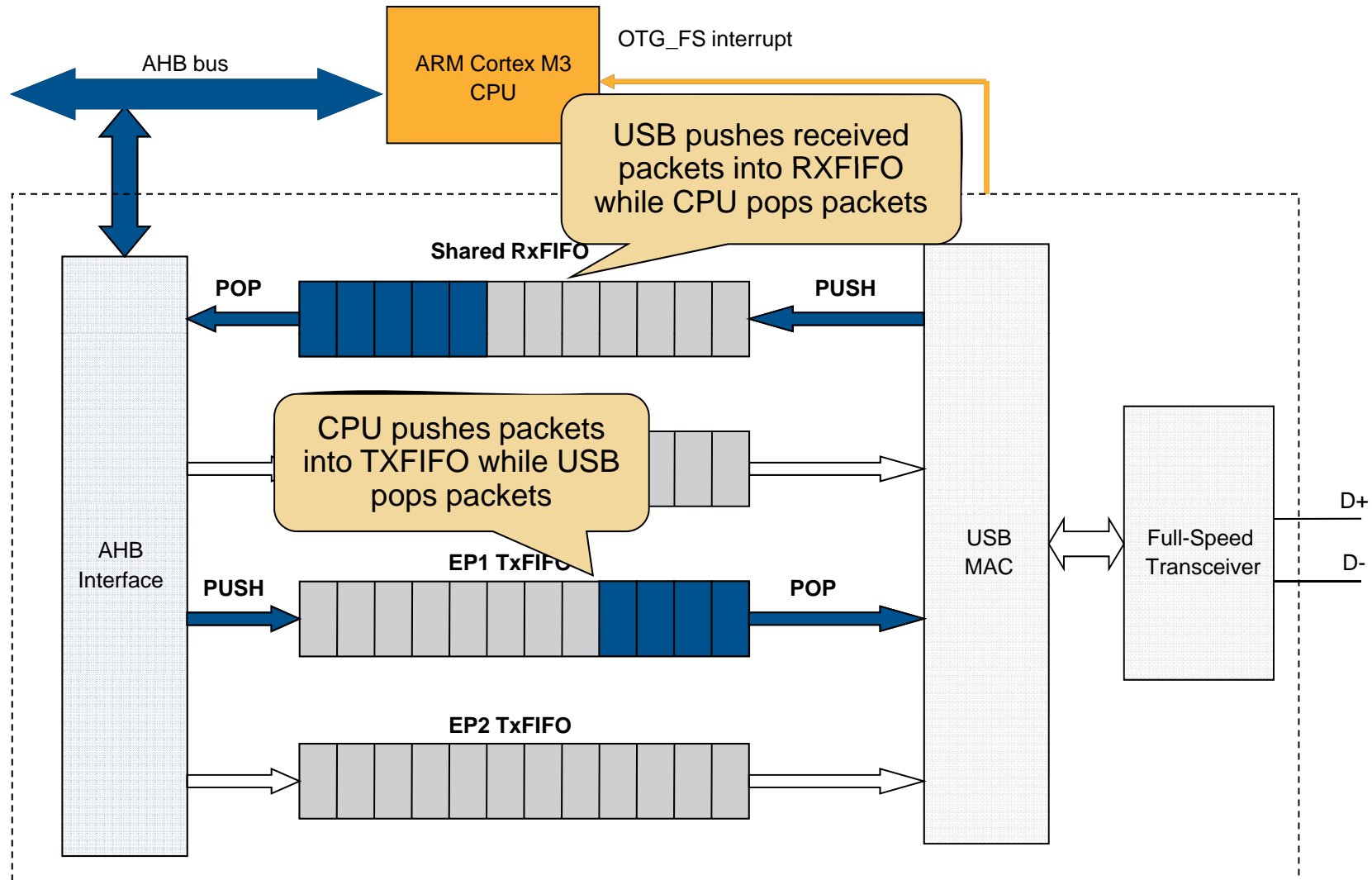


# USB 2.0 Full-Speed OTG Host/Device Controller Features



- Available on the STM32 connectivity line
  - **STM32F105/7** : 72 MHz cortex M3 MCU with up to 64KB SRAM and 256KB FLASH
  
- Main features
  - USB 2.0 Full-speed dual role Host/Device with OTG mode support
  - Can be configured as host-only or device-only controller
  - Integrated Full-speed PHY with OTG mode support
  - Dedicated packet memory of 1.25 Kbytes with advanced FIFO management and dynamic memory allocation
  - Device mode features
    - 1 bidirectional control endpoint0
    - Up to 3 IN and 3 OUT endpoints configurable to support Bulk, Interrupt or isochronous transfer
    - 1 shared FIFO for all OUT and control endpoints
    - Up to 3 dedicated TxFIFOs for IN and control endpoints

# USB 2.0 Full-Speed OTG Host/Device Controller FIFO operation



# USB 2.0 Full-Speed OTG Host/Device Controller

## FIFO Configuration & Transfer initialization

---



- **FIFOs configuration**
  - The size of TxFIFOs and the shared RxFIFO can be configured as needed by the application in a shared memory space of 1.25KBytes
  - Dynamic reconfiguration of the FIFOs sizes is possible
  
- **OUT/SETUP transfer initialization**
  - Software should configure the transfer size
  - Enable the endpoint for packet reception
  - Wait packets to be received
  
- **IN transfer initialization**
  - Software should configure the transfer size
  - Enable the endpoint
  - Start writing data to dedicated endpoint TxFIFO

# USB 2.0 Full-Speed OTG Host/Device Controller

## Interrupt handling

---



- Three important interrupts are used during transfer
  - **TxFIFO empty interrupt:** occurs when TxFIFO for endpoint n is empty or half empty, used to inform application that it can write more data to TxFIFO
  - **Shared RxFIFO Queue level interrupt:** raised when there is at least one received data packet inside the shared RxFIFO
  - **Correct Transfer Interrupt:** occurs when the full programmed transfer is finished on one endpoint



## USB 2.0 High-Speed OTG Host/Device Controller

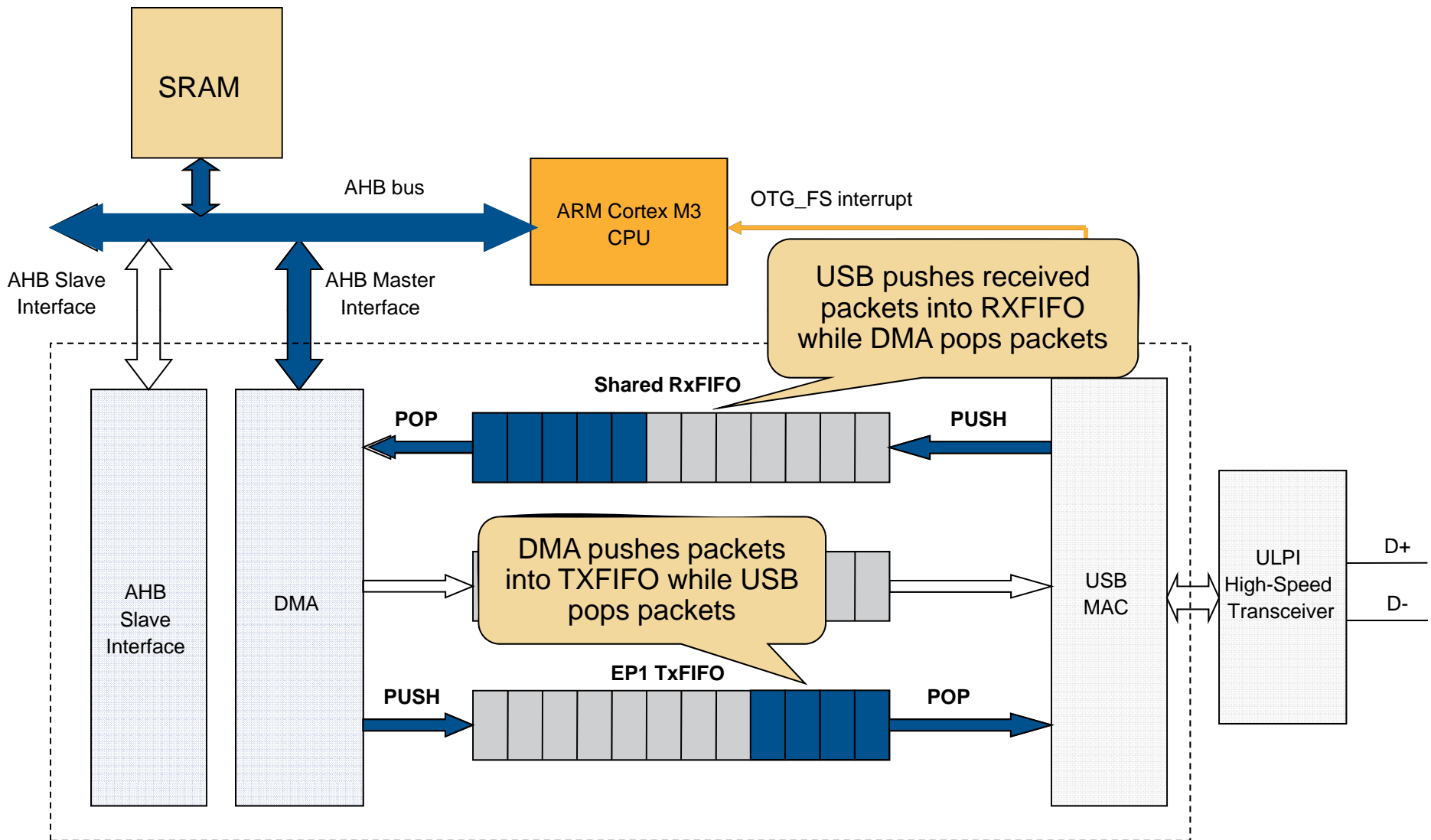


# USB 2.0 High-Speed OTG Host/Device Controller Features



- Available on the new STM32F2x Cortex-M3 ARM platform
  - Up to 120MHz MCU
  - Up to 128KBytes of SRAM and up to 1 Mbytes of FLASH
  - Up to two USB controllers
    - One Full-Speed USB dual role host/device OTG controller
    - One High-Speed USB dual role host/device OTG controller
  
- Device features
  - High-speed/Full-speed Device support
  - 1 bidirectional control endpoint0
  - Up to 5 IN and 5 OUT endpoints configurable to support Bulk, Interrupt or isochronous transfer
  - Dedicated DMA with access to internal SRAM or external memory bus
  - Dedicated packet memory of 4 Kbytes with advanced FIFO management and dynamic memory allocation
  - Internal analog transceiver for Full-Speed mode
  - Needs connection to external transceiver for High-speed mode through ULPI bus

# USB 2.0 High-Speed OTG Host/Device Controller FIFO operation



# USB 2.0 High-Speed OTG Host/Device Controller

## DMA operation

---



- DMA allows to manage a full transfer without CPU intervention after each transaction
  - CPU is informed of end of transfer using the Correct transfer interrupt
- A custom threshold FIFO level can be defined to trigger data transfer
  - **Transmit threshold TxFIFO trigger level:** free space in TxFIFO than can trigger an transfer from memory to FIFO
  - **Receive threshold RxFIFO trigger level:** a minimum receive data level in receive FIFO that can trigger a transfer to memory



# Summary Comparison Table



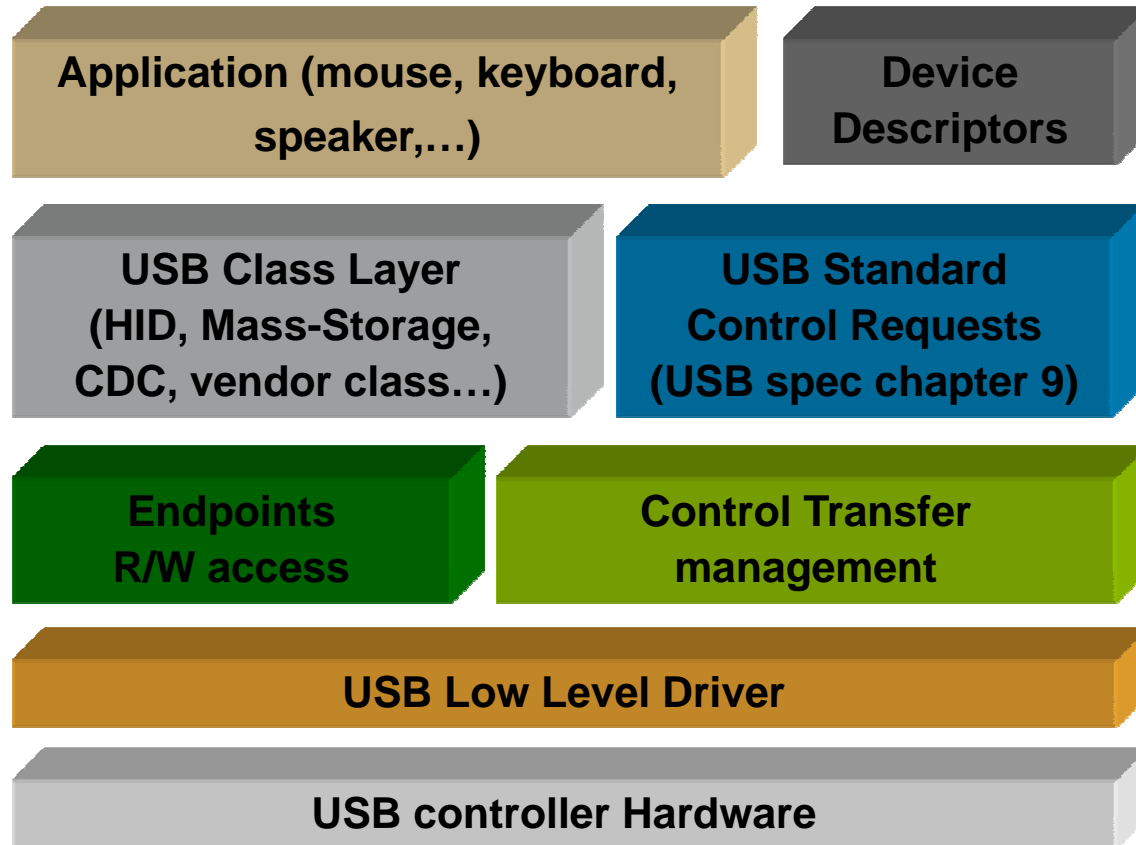
Controller	USB Full-Speed Device controller	USB OTG Full-speed dual role host/device	USB OTG High-speed dual role host/device
<b>Supported speeds</b>	Low/Full speed	Full speed only in device mode	High/Full speed
<b>Number of endpoints</b>	8 bidirectional	-1 bidirectional control - 3 IN - 3 OUT	-1 bidirectional control - 5 IN - 5 OUT
<b>CPU speed</b>	Up to 72MHz	Up to 72MHz	Up to 120 MHz
<b>FIFO support</b>	NO	YES	YES
<b>Packet Memory</b>	512 Bytes	1.25 KBytes	4 KBytes
<b>DMA support</b>	NO	NO	YES
<b>PHY</b>	Internal Full-Speed	Internal Full-Speed	-Internal Full-Speed -External High-speed (through ULPI bus)



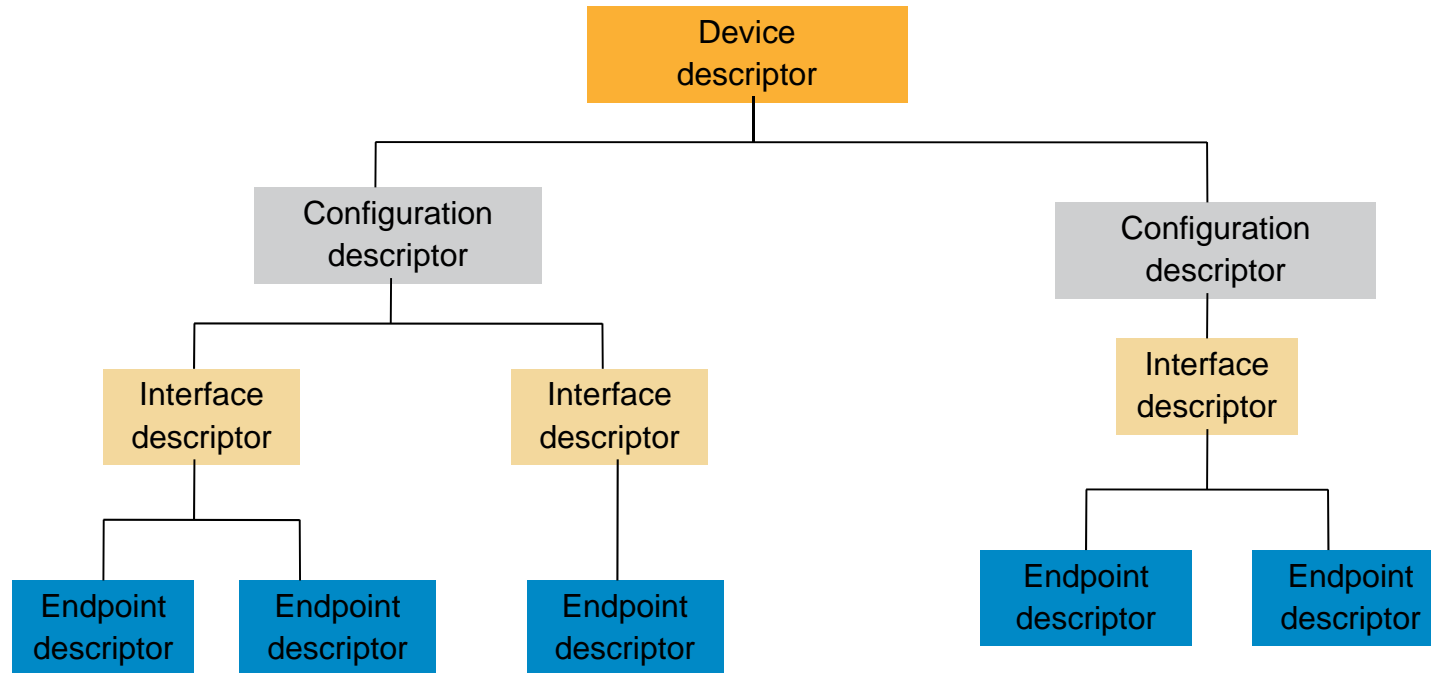
---

# Building Blocks of a USB Device Application

# Building Blocks of a USB Application



# USB Descriptors



- **Device Descriptor:** includes information of the device (PID, VID, Class) and the number of supported configurations
- **Configuration Descriptor:**
  - Includes the power configuration information, the number of supported interfaces in this configuration
  - Configurations are mutually exclusive
- **Interface Descriptor:** provides information about function or feature that device implements (class, subclass,...) also it indicates the number of endpoint it supports
- **Endpoint Descriptor:** provides information about the endpoint (address, type, max packet size)

# Device Descriptor



Offset	Field	Size	Description
0	bLength	1	Descriptor size in bytes (12h)
1	bDescriptor Type	1	01h
2	bcdUSB	2	USB spec release number (BCD) (0200h)
4	bDeviceClass	1	Class code (00h when interface desc defines class)
5	bDeviceSubclass	1	Subclass code
6	bDeviceProtocol	1	Protocol code
7	bMaxPacketSize0	1	Maximum endpoint size for endpoint 0 (64)
8	<b>idVendor</b>	2	Vendor ID
10	<b>idProduct</b>	2	Product ID
12	bcdDevice	2	Device release number (BCD)
14	iManufacturer	1	Index of string descriptor for the manufacturer
15	iProduct	1	Index of string descriptor for the product
16	iSerialNumber	1	Index of string descriptor for the serial number
17	<b>bNumConfigurations</b>	1	Number of possible configuration



# Configuration Descriptor

Offset	Field	Size (byte)	Description
0	bLength	1	Descriptor size in bytes (09h)
1	bDescriptorType	1	02h
2	<b>wTotalLength</b>	2	The number of bytes in the configuration descriptor and all of its subordinate description
4	<b>bNumInterfaces</b>	1	Number of interfaces in the configuration
5	bConfiguration Value	1	Identification for the configuration
6	iConfiguration	1	Index for string descriptor for the configuration
7	<b>bmAttributes</b>	1	Self or bus powered / remote wakeup setting
8	<b>bMaxPower</b>	1	Bus power required in units of 2mA



# Interface descriptor

Offset	Field	Size (byte)	Description
0	bLength	1	Descriptor size in bytes (09h)
1	bDescriptorType	1	04h
2	<b>bInterfaceNumber</b>	1	Number identifying the interface
3	bAlternateSetting	1	Number that identifies an Alternate interface for bInterfaceNumber
4	<b>bNumEndpoints</b>	1	Number of endpoints supported by the interface (without counting endpoint zero)
5	<b>bInterfaceClass</b>	1	Class code (ex HID = 03h)
6	<b>bInterfaceSubclass</b>	1	Subclass code (ex Boot Interface Subclass = 01h)
7	bInterfaceProtocol	1	Protocol code (ex Mouse = 02h)
8	iInterface	1	Index for string descriptor for the interface



# Endpoint Descriptor

Offset	Field	Size (byte)	Description
0	bLength	1	Descriptor size in bytes (07h)
1	bDescriptorType	1	05h
2	<b>bEndpointAddress</b>	1	Endpoint number and direction
3	<b>bmAttributes</b>	1	Transfer Type (bulk, interrupt, isochronous)
4	<b>wMaxPacketSize</b>	2	Maximum packet size supported
6	<b>bInterval</b>	1	Polling time for interrupt or isochronous EP





# Class Layer

---

- A class specifies the operation of group of USB devices that have similar functionalities, ex:
  - Audio class (speaker, microphone)
  - Communication device class (virtual COM-port, modems, Ethernet adapters,...)
  - Human Interface Device class (mouse, keyboard, joystick,...)
- Defining a USB class allows to have unique host driver for all devices belonging to the class
- A USB class defines
  - Required or optional endpoints
  - Needed Interfaces
  - Class-specific descriptors
  - Required values for fields in the standard descriptors
  - Class Control requests
  - Format of data to be transferred and optionally the protocol layer for data transfer (ex Bulk-only transfer for Mass-Storage class)



# Human Interface Device (HID)

---

- HID class is mainly intended for devices that have interaction with human inputs (moving a joystick, a mouse, pressing a keyboard...) but can be used for other application
- Supported natively by MS Windows operating system
- HID needs one IN interrupt endpoint to transfer data, the host will poll the IN endpoint periodically to check if device has data to transfer
- Transferred data is formatted in fixed structure called HID Report
- HID defines six specific control requests
  - Get/Set Report: allows to get/send report to device
  - Set/Get Idle: allows to get/set the idle rate
  - Set/Get Protocol: allows to get/set used protocol (boot or report protocol)
- Two class specific descriptors are defined
  - HID class descriptor
  - HID Report descriptor



# USB Control Transfer Management

---

- Should implement a state machine for managing the three stages of a control transfer on endpoint0
  - SETUP stage
  - Optional Data IN or OUT stage
  - Handshake stage
  
- During the SETUP stage, the received request can be
  - Standard USB Request as defined in USB spec Chapter 9
  - Standard Class USB request (HID, CDC,...)
  - Vendor Class USB request
  
- In case a request is not supported, the control endpoint should send a STALL handshake packet to host
  
- A control request can target
  - Device
  - Interface
  - Endpoint



# SETUP request Structure

Offset	Field	Size	Value	Description
0	bmRequestType	1	Bit-Map	D7 Data Phase Transfer Direction 0 = Host to Device 1 = Device to Host D6..5 Type 0 = Standard 1 = Class 2 = Vendor 3 = Reserved D4..0 Recipient 0 = Device 1 = Interface 2 = Endpoint 3 = Other 4..31 = Reserved
1	bRequest	1	Value	Request
2	wValue	2	Value	Value
4	wIndex	2	Index or Offset	Index
6	wLength	2	Count	Number of bytes to transfer if there is a data phase



# Standard USB Control Requests

---

- Control requests are defined in chapter 9 of the USB specification
- Used mainly during the enumeration phase by the host to get needed descriptors information about the device and to select the needed configuration
- Some standard requests (Get Feature/ Set Feature, Get Status) may be used as a class requests to get/set class specific features

# Standard USB control Requests



<b>bRequest</b>	<b>Target</b>	<b>Description</b>
<b>GET_STATUS (0x00)</b>	Device or Interface or Endpoint	-Device: allows to get the device power settings -Endpoint: allows to check STALL status of endpoint
<b>CLEAR_FEATURE (0x01)</b>	Device or Interface or Endpoint	-Device: allows to clear Device remote wakeup feature -Endpoint: allows to clear STALL status for an endpoint
<b>SET_FEATURE (0x03)</b>	Device or Interface or Endpoint	-Device: Sets Remote wakeup feature -Endpoint: Sets STALL condition on endpoint
<b>SET_ADDRESS (0x05)</b>	Device	Sets device address
<b>GET_DESCRIPTOR (0x06)</b>	Device	Gets a Descriptor (Device, Configuration, String)
<b>GET_CONFIGURATION (0x08)</b>	Device	Gets the value of current configuration
<b>SET_CONFIGURATION (0x09)</b>	Device	Sets the configuration to use
<b>GET_INTERFACE (0x0A)</b>	Interface	For interfaces that have alternate interfaces, the host requests the current alternate interface
<b>SET_INTERFACE (0x0B)</b>	Interface	For interfaces that have alternate interfaces, the host requests to set a particular alternate interface



# USB Low Level Driver

---

- Functions/Macros that do direct access to USB block registers for
  - Device global initializations (ex: device address, speed,..)
  - Endpoints initialization (ex: address, transfer type, ...)
  - Transfer initialization
  - Packet memory area or FIFO access
  
- Manage the USB interrupts with callbacks to application layers
  - Global device interrupts (USB Reset, USB suspend,..)
  - Endpoints transfer related interrupts (correct transfer interrupt, TxFIFO empty,...)
  
- Power management functions during Suspend/resume
  - Entry in low-power mode during suspend mode
  - Remote wakeup management

# STM32 USB device firmware library







# USB Device Developer Kit

---

- Allows to easily get started with USB device development on STM32 platforms
- The kit provides all the needs firmware blocks including
  - Low level USB drivers
  - Firmware for handling the standard control requests (chapter 9)
  - USB class layer implementation with demos running on evalboards for
    - Mass-Storage
    - HID
    - CDC Virtual COM port
    - Audio (speaker, microphone)
    - Device Firmware Upgrade
- The Library allows easy development of Custom vendor class
- All the class implementations are validated using the USB Command Verifier tool provided by the USB-IF

# Folder Organization of the STM32 USB Developer Kit



## ■ Libraries

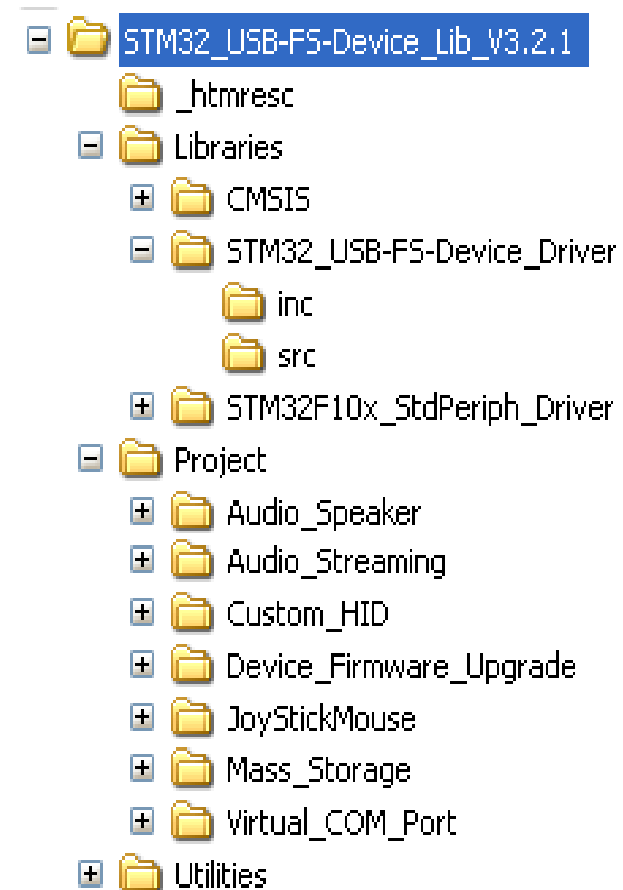
- **CMSIS** : Cortex Microcontroller Software Interface Standard files (startup files, NVIC, clock config)
- **STM32\_USB-FS-Device\_Driver**: Includes the USB low level driver + firmware for handling control transfer and standard control requests
- **STM3210x\_StdPeriph\_Driver**: low level drivers for standard peripherals (timer, clocks, ..)

## ■ Project

- Implementation for the various class demos
- Each demo includes workspaces for different third-party toolsets (IAR, KEIL, RIDE, HITEX, Attolic)

## ■ Utilities

- Evalboard utilities (buttons, LCD, SD card access,..)



# STM32\_USB-FS-Device\_Driver

## Low Level Driver



- The folder includes low level drivers for:
  - The USB Full-speed device controller
  - The OTG Full-speed controller (only for device mode)
- Source files for the OTG Full-speed device controller:

file	Description
<b>otg_fs_int.c</b>	All USB Interrupt handling with callbacks to application layer
<b>otg_fs_pcd.c</b>	High level functions for Endpoint access (Read/Write, Open/Close,..)
<b>otg_fs_cal.c</b>	Core access layer (function doing register access for global device configuration, endpoint initializations, transfer initializations, FIFOs R/W)
<b>otg_fs_dev.c</b>	Wrapper for device/endpoints high level access, implemented for compatibility reason with the USB full-speed controller
<b>Usb_sil.c</b>	Serial Interface Layer (SIL) :Wrapper for endpoint R/W access implemented for compatibility reason with the USB full-speed controller

# STM32\_USB-FS-Device\_Driver

## Control transfer & Library initialization

---



- All the handling of control transfer and the standard USB requests is done in file *usb\_core.c*
- The implementation allows the handling of the standard control requests and the dispatching to class layer when receiving a class control request
- File *usb\_init.c* includes one function *USB\_Init()* that should be called to initialize the library structures and to do needed device initialization (control configuration, memory allocation,...)



# Class Layer Implementation

---

- The various USB classes implementations are found in “Project” folder
- For each class, the class specific control requests are implemented in file `usb_prop.c`
- Non-control endpoints correct transfer interrupt handling is done in file `usb_endp.c`
- The USB descriptors for the device are implemented in file `usb_desc.c`
- Device power management is implemented in file `usb_pwr.c`

# Overview about PHDC class and Continua Stack



# Continua Health Alliance

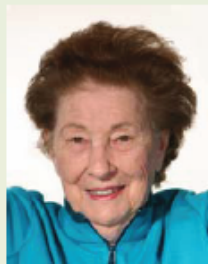
---

- The Continua Health Alliance is an open industry consortium with more than 200 member companies around the world
- Their Mission is to establish an eco-system of interoperable personal health systems in the healthcare sector.
- ST is a contributing member
- ST is allowed to use the Continua Logo
- After certification the logo can be used on products.

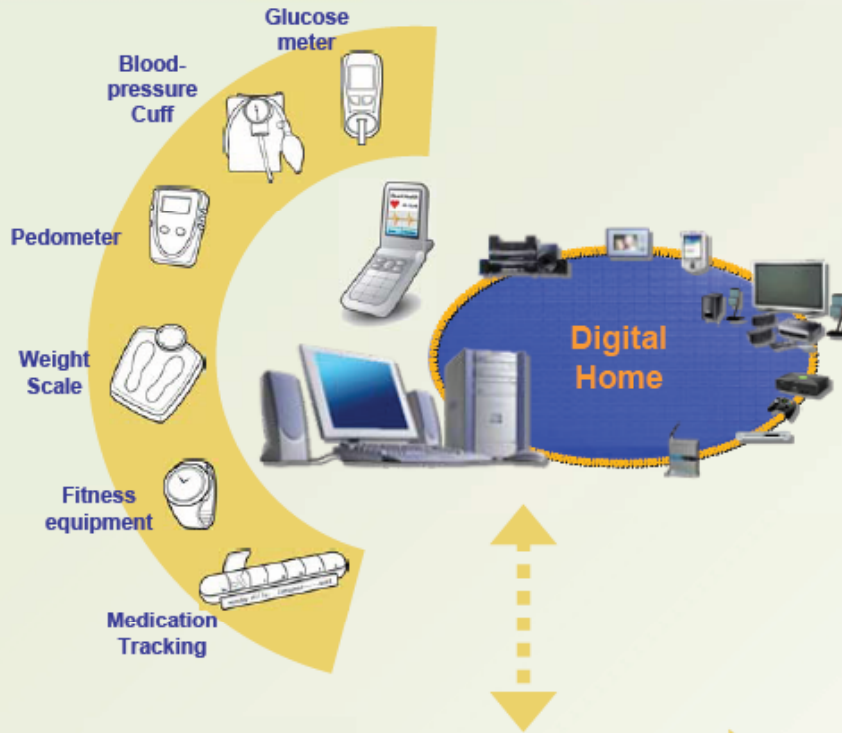




Healthy family



Elderly



## Health & Wellness

- Weight loss
- Fitness
- Email / chat / video
- Appt scheduling
- Personal Health Records

## Disease Management

- Vital sign monitoring (RPM)
- Medication reminders and compliance
- Trend analysis and alerts
- Connect with family care givers

## Aging Independently

- An adult child helping their elderly parents age gracefully in their own home.
- Basic life monitoring as appropriate (ADL)

Internet



Weight loss and fitness coaching



Personal Health Records



Healthcare Professionals



Disease management service



Family care givers





## SENSORS

Home sensing & control 	Weight Scale 
Bed / Chair Sensors 	Blood-pressure 
Implant Monitors 	Glucose Meter 
Baby Monitors 	Pulse Oximeter 
PERS 	Spirometer 
Consumer Electronics 	Medication Tracking 
	Pedometer 
	Fitness equipment 

## CONNECTIVITY

ZigBee™

Bluetooth®

USB

Z-WAVE®

MICS / MEDS

Ethernet

Wi Fi ALLIANCE

HOMEPLUG®  
POWERLINE ALLIANCE

## AGGREGATION COMPUTATION

PC

Personal Health System

Cell Phone

Set Top Box

Aggregator

NETWORK (POTS, Cellular, BB)

## SERVICES

Healthcare Provider Service

Disease Management Service


Diet or Fitness Service


Personal Health Record Service


Implant Monitoring Service


# ST Healthcare Library features





Thermometer 


Pulse Oximeter 


Pulse / Blood Pressure 

Weight Scale 

Glucose Meter 

Cardiovascular and Strength Fitness Monitor 

Independent Living Activity 

Medication Adherence 

**Transport Independent**



- 11073-10404 = Pulse Oximeter
- 11073-10406 = Pulse / Heart Rate
- 11073-10407 = Blood Pressure
- 11073-10408 = Thermometer
- 11073-10415 = Weighing Scale
- 11073-10417 = Glucose
- 11073-10441 = Cardiovascular Fitness Monitor
- 11073-10442 = Strength Fitness Equipment
- 11073-10471 = Independent Living Activity
- 11073-10472 = Medication Monitor
- 11073-20601 = Base Framework Protocol



Personal Health Device  
Class Specification

PC 

Personal Health System 

Cell Phone 

Set Top Box 

Aggregator 

# Thermometer Demo based on latest CESL

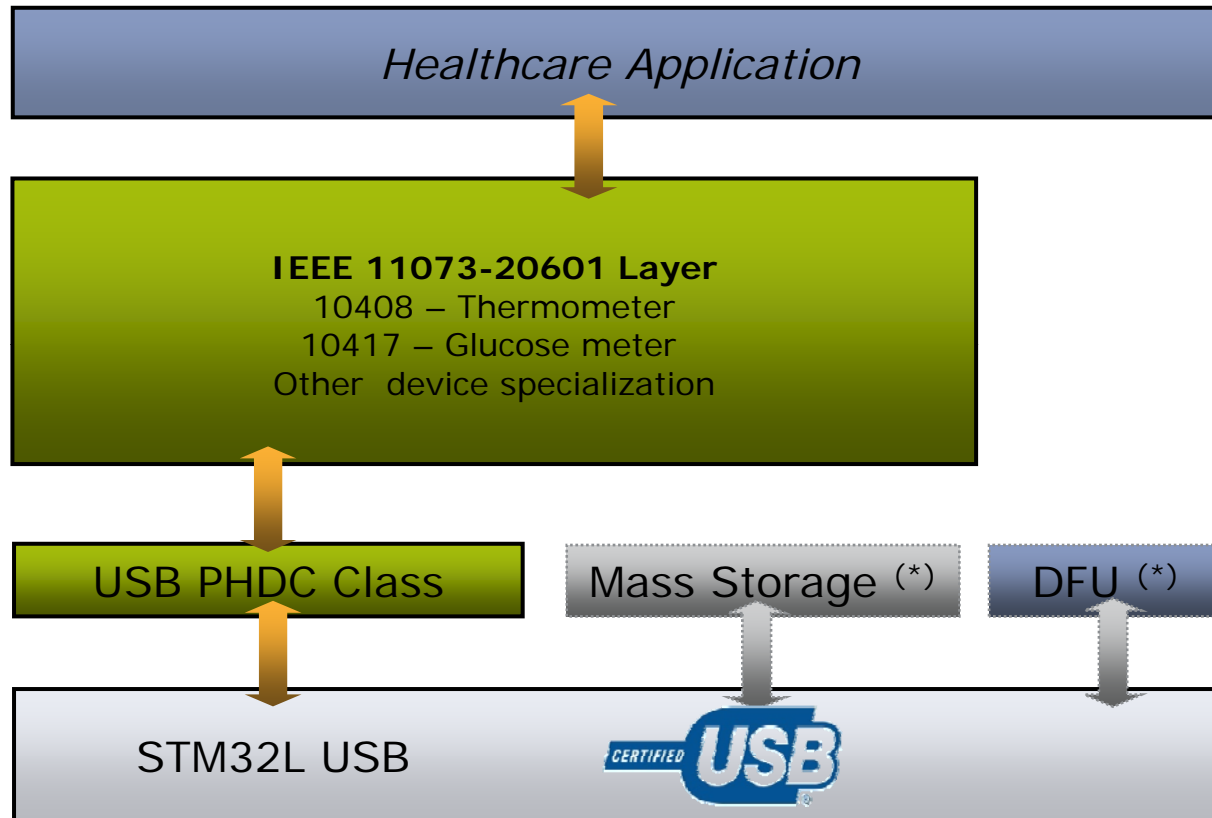


The screenshot displays the Continua Manager application window. At the top, the title bar reads "Continua Manager" and the menu bar includes "File", "Edit", and "Help". The "Current State" is set to "Operating". Below this, the "Select Shim Directory" field is populated with "C:\Program Files\CESL\_Binary\bin\" and includes "Browse" and "Start Transport" buttons. The "Device List" table contains one entry:

Device Name	Transport	Device Specialization	Address
53 Medic Agent	USB	Thermometer	

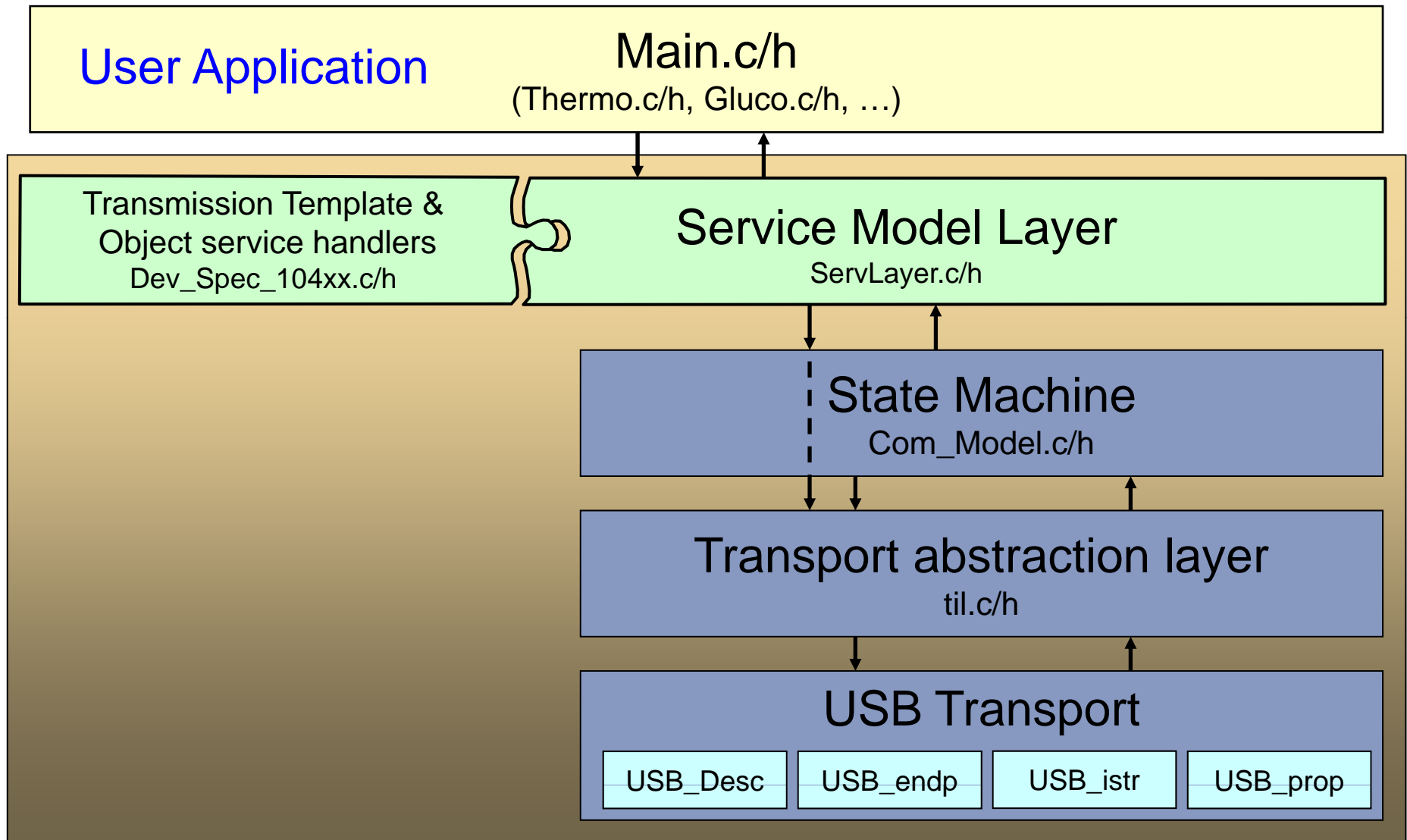
Below the table are "Discover", "Disconnect", "Unassociate", and "Abort" buttons. The "Device Information" section shows "Connected to: VASC-MDS", "System Model: Thermometer", "System Manufacturer: STMicroelectronics", and "System ID: 0x31-0x32-0x33-0x34-0x35-0x36-0x37-0x38". A grid of icons is visible, and a log shows three temperature readings: "Timestamp: 2:47:00 pm on Thu April 15 2010" and "Body Temp.: 37 °C". The "Output" window shows log messages: "0x388d80:\AssociateMessages.c(563)EventReceiverHandlerConnection Event received" and "0x387008:\ManagerFSM.c(626)MgrCheckingConfigAccepting Config -> Entering Operating state". The footer features logos for "Continua HEALTH ALLIANCE" and "Lni", along with "Transport: Enabled" and icons for "USB", "Bluetooth", and "TCP/IP". The Windows taskbar at the bottom shows the Start button and several open applications, including "IAR Embedded W...", "Continua Manager", "PHDC.bmp - Paint", and "Internet Expl...", with the system clock at 2:47 PM.

# Healthcare Firmware stack (1/2)



(\*) Optional classes not linked to Medical

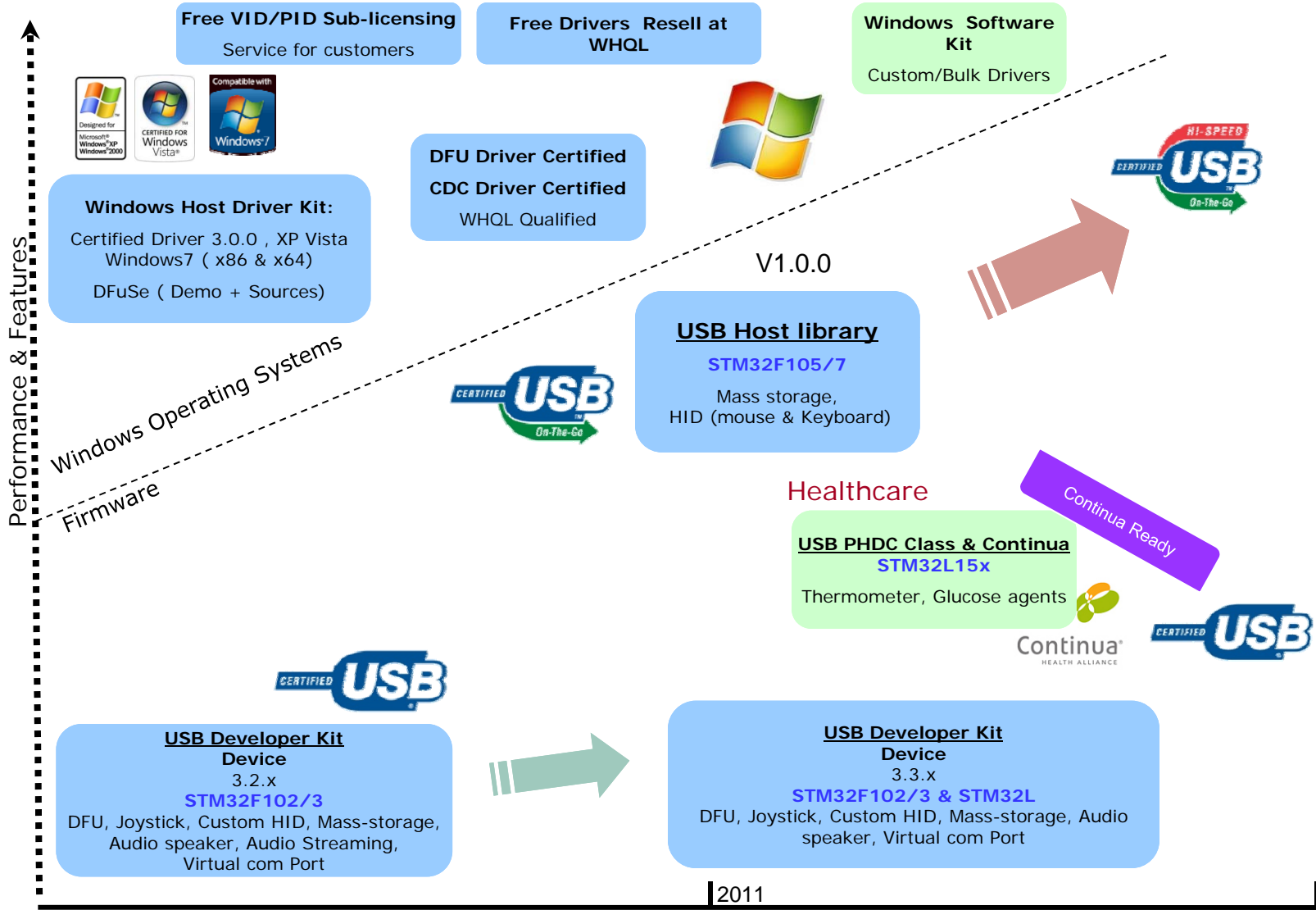
# Healthcare Firmware stack (3/3)



## ST offer for PC software USB



# USB Offer with STM32 ( F1, L & F2)



# New PC Windows Driver (1/2)



## Description :

- USB driver for Microsoft operating systems.
- Designed to work with USB DFU and all specific USB classes
- Allow access to the Control, Interrupt and Bulk pipes and is an alternative to WinUSB driver without limitations.
  
- WHQL Certified and in production
  - Used for STMicroelectronics USB Bootloader ( DFU)
  - At many customers since 2008.
  
- Full documented API and Reference examples with Visual C++ (6, 2005, 2008 and 2010 : x86 and x64)

## Applications usage:

- Device Firmware Upgrade STMicroelectronics Extension (DfuSe)
- Vendor Class involving Bulk/Interrupts Pipes
- Any classes, except those with Isochronous Pipes.



# New PC Windows Driver ( 2/2)



- Compatible Operating Systems
  - MS Windows 98SE
  - MS Windows 2000
  - MS Windows XP (x86 & x64)
  - MS Windows Vista (x86 & x64)
  - MS Windows Seven (x86 & x64)



- WHQL Certification Added Value to our customers
  - Providing quality end-to-end experience of customers
  - Retailers expect the logo on devices & concentrate on own business
  - Consumers & customers look for logo-qualified products
- On-line Windows Update by clicking the Update Driver button in Device Manager

# Microsoft Logo Certification



## Windows Logo Verification Report: Approved

Submission ID: 1377238  
Submission Date: 11/23/2009  
Logo Completion Date: 11/24/2009  
Company: STMicroelectronics  
Product Name: STMicroelectronics Device  
Category: Device  
Subcategory: Unclassified  
Qualification Level: Signature Only - Microsoft Windows 2000 family - Unclassified  
Signature Only - Microsoft Windows XP family, x86 - Unclassified  
Signature Only - Microsoft Windows XP family, x64 - Unclassified  
Signature Only - Microsoft Windows Vista family, x86 - Unclassified  
Signature Only - Microsoft Windows Vista family, x64 - Unclassified  
Signature Only - Device - Compatible with Windows 7  
Signature Only - Device - Compatible with Windows 7 x64  
Marketing Names: N/A  
Additional Information:  
Firmware version: 3.1.0



# New Service for STM32 USB small customers

---

- **Description :**
  - USB “PID” and VID (0x0483) from STMicroelectronics Sub-licensing
  
- **Program Process**
  - Receive Requests from our Customers thru sales offices with customer details:
    - 1) COMPANY NAME AUTHORIZING USE TO :
    - 2) Contact Name /Address and E-mail address:
    - 3) Name/Sales type of the ST Microcontroller product name :
    - 4) Name of USB end-product : { if possible USB device string Product}
  - PID Booked in an internal ST Database
  
  - Final Step :
    - ST will send the approval list to USB-IF
    - Approval by USB-IF
    - PID sent to the customer with a “letter form Agreement”

# End-to-end experience of customers



The screenshot displays a Windows XP desktop environment. In the background, the Microsoft Windows Update website is open in Internet Explorer. The website shows a 'Review Your Installation' summary with a 'Successful Update' status for 'STMicroelectronics STI'. In the foreground, the Device Manager window is open, showing the 'STM Device in DFU Mode' selected under 'System devices'. A 'Driver File Details' dialog box is open, displaying the following information:

- Device: STM Device in DFU Mode
- Driver files: C:\WINDOWS\System32\Drivers\STTub30.sys
- Provider: STMicroelectronics
- File version: 3.0B
- Copyright: Copyright (C) STMicroelectronics 2009
- Digital Signer: Microsoft Windows Hardware Compatibility

The taskbar at the bottom shows the Start button, a '2 - Paint' window, and the 'Microsoft Windows U...' window. The system tray includes the Internet icon, a 100% zoom level, and the time 7:34 PM.