USB Interface Management

USB Core - LUFA/Drivers/USB/USB.h

USB Controller definitions for general USB controller management. More...

Modules

USB Interface Management (AVR8) USB Controller definitions for the AVR8 microcontrollers.
USB Interface Management (UC3) USB Controller definitions for the AVR32 UC3 microcontrollers.
USB Interface Management (XMEGA) USB Controller definitions for the AVR XMEGA microcontrollers.

Functions

void **USB_USBTask** (void)

Variables

volatile bool	**USB_IsInitialized**
USB_Request_Header_t	**USB_ControlRequest**

Endpoint Direction Masks

#define **ENDPOINT_DIR_MASK** 0x80
#define **ENDPOINT_DIR_OUT** 0x00
#define **ENDPOINT_DIR_IN** 0x80

Pipe Direction Masks

#define **PIPE_DIR_MASK** 0x80
#define **PIPE_DIR_OUT** 0x00
#define **PIPE_DIR_IN** 0x80

Endpoint/Pipe Type Masks

#define **EP_TYPE_MASK** 0x03
#define **EP_TYPE_CONTROL** 0x00
#define **EP_TYPE_ISOCHRONOUS** 0x01
#define **EP_TYPE_BULK** 0x02
#define **EP_TYPE_INTERRUPT** 0x03

Detailed Description

Functions, macros, variables, enums and types related to the setup and management of the USB interface.

Macro Definition Documentation

#define ENDPOINT_DIR_IN 0x80

Endpoint address direction mask for an IN direction (Device to Host) endpoint. This may be ORed with the index of the address within a device to obtain the full endpoint address.

#define ENDPOINT_DIR_MASK 0x80

Endpoint direction mask, for masking against endpoint addresses to retrieve the endpoint's direction for comparing with the `ENDPOINT_DIR_*` masks.

#define ENDPOINT_DIR_OUT 0x00

Endpoint address direction mask for an OUT direction (Host to Device) endpoint. This may be ORed with the index of the address within a device to obtain the full endpoint address.

#define EP_TYPE_BULK 0x02

Mask for a BULK type endpoint or pipe.

Note

See **Endpoint Management** and **Pipe Management** for endpoint/pipe functions.

#define EP_TYPE_CONTROL 0x00

Mask for a CONTROL type endpoint or pipe.

Note

See **Endpoint Management** and **Pipe Management** for endpoint/pipe functions.

#define EP_TYPE_INTERRUPT 0x03

Mask for an INTERRUPT type endpoint or pipe.

Note

See **Endpoint Management** and **Pipe Management** for endpoint/pipe functions.

#define EP_TYPE_ISOCHRONOUS 0x01

Mask for an ISOCHRONOUS type endpoint or pipe.

Note

See **Endpoint Management** and **Pipe Management** for endpoint/pipe functions.

#define EP_TYPE_MASK 0x03

Mask for determining the type of an endpoint from an endpoint descriptor. This should then be compared with the `EP_TYPE_*` masks to determine the exact type of the endpoint.

#define PIPE_DIR_IN 0x80

Endpoint address direction mask for an IN direction (Device to Host) endpoint. This may be ORed with the index of the address within a device to obtain the full endpoint address.

#define PIPE_DIR_MASK 0x80

Pipe direction mask, for masking against pipe addresses to retrieve the pipe's direction for comparing with the `PIPE_DIR_*` masks.

#define PIPE_DIR_OUT 0x00

Endpoint address direction mask for an OUT direction (Host to Device) endpoint. This may be ORed with the index of the address within a device to obtain the full endpoint address.

Function Documentation

void USB_USBTask (void)

This is the main USB management task. The USB driver requires this task to be executed continuously when the USB system is active (device attached in host mode, or attached to a host in device mode) in order to manage USB communications. This task may be executed inside an RTOS, fast timer ISR or the main user application loop.

The USB task must be serviced within 30ms while in device mode, or within 1ms while in host mode. The task may be serviced at all times, or (for minimum CPU consumption):

- In device mode, it may be disabled at start-up, enabled on the firing of the EVENT_USB_Device_Connect() event and disabled again on the firing of the EVENT_USB_Device_Disconnect() event.
- In host mode, it may be disabled at start-up, enabled on the firing of the EVENT_USB_Host_DeviceAttached() event and disabled again on the firing of the EVENT_USB_Host_DeviceEnumerationComplete() or EVENT_USB_Host_DeviceEnumerationFailed() events.

If in device mode (only), the control endpoint can instead be managed via interrupts entirely by the library by defining the INTERRUPT_CONTROL_ENDPOINT token and passing it to the compiler via the -D switch.

See Also

USB Events for more information on the USB events.

Variable Documentation

USB_Request_Header_t USB_ControlRequest

Structure containing the last received Control request when in Device mode (for use in user-applications inside of the EVENT_USB_Device_ControlRequest() event, or for filling up with a control request to issue when in Host mode before calling USB_Host_SendControlRequest().

Note

The contents of this structure is automatically endian-corrected for the current CPU architecture.

volatile bool USB_IsInitialized

Indicates if the USB interface is currently initialized but not necessarily connected to a host or device (i.e. if USB_Init() has been run). If this is false, all other library globals related to the USB driver are invalid.

Attention

This variable should be treated as read-only in the user application, and never manually changed in value.