

MCUXpresso SDK USB Stack OTG Reference Manual

NXP Semiconductors

Document Number: MCUXSDKUSBOAPIRM

Rev. 0

Dec 2021

Contents

Chapter Definitions and structures

1.1	Overview	1
1.2	Data Structure Documentation	2
1.2.1	struct usb_version_t	2
1.3	Typedef Documentation	3
1.3.1	usb_device_handle	3
1.4	Enumeration Type Documentation	3
1.4.1	usb_status_t	3
1.4.2	usb_controller_index_t	3

Chapter USB OTG driver

2.1	Overview	5
2.2	Data Structure Documentation	7
2.2.1	struct usb_otg_descriptor_t	7
2.2.2	struct usb_otg_instance_t	8
2.3	Typedef Documentation	8
2.3.1	usb_otg_callback_t	8
2.4	Enumeration Type Documentation	9
2.4.1	usb_otg_status_type_t	9
2.4.2	usb_otg_device_state_t	9
2.4.3	usb_otg_stack_init_type_t	10
2.4.4	usb_otg_event_type_t	10
2.5	Function Documentation	10
2.5.1	USB_OtgInit	10
2.5.2	USB_OtgDeinit	11
2.5.3	USB_OtgTaskFunction	11
2.5.4	USB_OtgKhciIsrFunction	12
2.5.5	USB_OtgBusDrop	12
2.5.6	USB_OtgBusRequest	12
2.5.7	USB_OtgBusRelease	13

Contents

Section Number	Title	Page Number
2.5.8	USB_OtgClearError	13
2.5.9	USB_OtgNotifyChange	14
2.6	USB OTG Controller driver	15
2.6.1	Overview	15
2.6.2	Data Structure Documentation	16
2.6.3	Macro Definition Documentation	16
2.6.4	Enumeration Type Documentation	16
2.7	USB OTG Peripheral driver	18
2.7.1	Overview	18
2.7.2	Function Documentation	18
Chapter	USB OS Adapter	
Chapter	Data Structure Documentation	
4.0.3	usb_serial_port_config_t Struct Reference	23

Chapter 1

Definitions and structures

1.1 Overview

This lists the common definitions and structures for USB stack.

Data Structures

- struct `usb_version_t`
USB stack version fields. [More...](#)

Macros

- #define `USB_STACK_VERSION_MAJOR` (0x01UL)
Defines USB stack major version.
- #define `USB_STACK_VERSION_MINOR` (0x00UL)
Defines USB stack minor version.
- #define `USB_STACK_VERSION_BUGFIX` (0x00U)
Defines USB stack bugfix version.
- #define `USB_MAKE_VERSION`(major, minor, bugfix) (((major) << 16) | ((minor) << 8) | (bugfix))
USB stack version definition.
- #define `USB_STACK_COMPONENT_VERSION` MAKE_VERSION(USB_STACK_VERSION_MAJOR, USB_STACK_VERSION_MINOR, USB_STACK_VERSION_BUGFIX)
USB stack component version definition, changed with component in yaml together.

Typedefs

- typedef void * `usb_host_handle`
USB host handle type define.
- typedef void * `usb_device_handle`
USB device handle type define.
- typedef void * `usb_otg_handle`
USB OTG handle type define.

Enumerations

- enum `usb_status_t` {
 `kStatus_USB_Success` = 0x00U,
 `kStatus_USB_Error`,
 `kStatus_USB_Busy`,
 `kStatus_USB_InvalidHandle`,
 `kStatus_USB_InvalidParameter`,
 `kStatus_USB_InvalidRequest`,
 `kStatus_USB_ControllerNotFound`,
 `kStatus_USB_InvalidControllerInterface`,
 `kStatus_USB_NotSupported`,
 `kStatus_USB_Retry`,
 `kStatus_USB_TransferStall`,
 `kStatus_USB_TransferFailed`,
 `kStatus_USB_AllocFail`,
 `kStatus_USB_LackSwapBuffer`,
 `kStatus_USB_TransferCancel`,
 `kStatus_USB_BandwidthFail`,
 `kStatus_USB_MSDDStatusFail` ,
 `kStatus_USB_DataOverRun` }
 USB error code.
- enum `usb_controller_index_t` {
 `kUSB_ControllerKhci0` = 0U,
 `kUSB_ControllerKhci1` = 1U,
 `kUSB_ControllerEhci0` = 2U,
 `kUSB_ControllerEhci1` = 3U,
 `kUSB_ControllerLpcIp3511Fs0` = 4U,
 `kUSB_ControllerLpcIp3511Fs1` = 5U,
 `kUSB_ControllerLpcIp3511Hs0` = 6U,
 `kUSB_ControllerLpcIp3511Hs1` = 7U,
 `kUSB_ControllerOhci0` = 8U,
 `kUSB_ControllerOhci1` = 9U,
 `kUSB_ControllerIp3516Hs0` = 10U,
 `kUSB_ControllerIp3516Hs1` = 11U,
 `kUSB_ControllerDwc30` = 12U,
 `kUSB_ControllerDwc31` = 13U }
 USB controller ID.

1.2 Data Structure Documentation

1.2.1 struct `usb_version_t`

Data Fields

- `uint8_t` `major`

- *Major.*
uint8_t [minor](#)
- *Minor.*
uint8_t [bugfix](#)
- *Bug fix.*

1.3 Typedef Documentation

1.3.1 typedef void* usb_device_handle

For device stack it is the whole device handle; for host stack it is the attached device instance handle

1.4 Enumeration Type Documentation

1.4.1 enum usb_status_t

Enumerator

- kStatus_USB_Success* Success.
- kStatus_USB_Error* Failed.
- kStatus_USB_Busy* Busy.
- kStatus_USB_InvalidHandle* Invalid handle.
- kStatus_USB_InvalidParameter* Invalid parameter.
- kStatus_USB_InvalidRequest* Invalid request.
- kStatus_USB_ControllerNotFound* Controller cannot be found.
- kStatus_USB_InvalidControllerInterface* Invalid controller interface.
- kStatus_USB_NotSupported* Configuration is not supported.
- kStatus_USB_Retry* Enumeration get configuration retry.
- kStatus_USB_TransferStall* Transfer stalled.
- kStatus_USB_TransferFailed* Transfer failed.
- kStatus_USB_AllocFail* Allocation failed.
- kStatus_USB_LackSwapBuffer* Insufficient swap buffer for KHCI.
- kStatus_USB_TransferCancel* The transfer cancelled.
- kStatus_USB_BandwidthFail* Allocate bandwidth failed.
- kStatus_USB_MSDStatusFail* For MSD, the CSW status means fail.
- kStatus_USB_DataOverRun* The amount of data returned by the endpoint exceeded either the size of the maximum data packet allowed from the endpoint or the remaining buffer size.

1.4.2 enum usb_controller_index_t

Enumerator

- kUSB_ControllerKhci0* KHCI 0U.
- kUSB_ControllerKhci1* KHCI 1U, Currently, there are no platforms which have two KHCI IPs, this is reserved to be used in the future.

Enumeration Type Documentation

kUSB_ControllerEhci0 EHCI 0U.

kUSB_ControllerEhci1 EHCI 1U, Currently, there are no platforms which have two EHCI IPs, this is reserved to be used in the future.

kUSB_ControllerLpcIp3511Fs0 LPC USB IP3511 FS controller 0.

kUSB_ControllerLpcIp3511Fs1 LPC USB IP3511 FS controller 1, there are no platforms which have two IP3511 IPs, this is reserved to be used in the future.

kUSB_ControllerLpcIp3511Hs0 LPC USB IP3511 HS controller 0.

kUSB_ControllerLpcIp3511Hs1 LPC USB IP3511 HS controller 1, there are no platforms which have two IP3511 IPs, this is reserved to be used in the future.

kUSB_ControllerOhci0 OHCI 0U.

kUSB_ControllerOhci1 OHCI 1U, Currently, there are no platforms which have two OHCI IPs, this is reserved to be used in the future.

kUSB_ControllerIp3516Hs0 IP3516HS 0U.

kUSB_ControllerIp3516Hs1 IP3516HS 1U, Currently, there are no platforms which have two IP3516HS IPs, this is reserved to be used in the future.

kUSB_ControllerDwc30 DWC3 0U.

kUSB_ControllerDwc31 DWC3 1U Currently, there are no platforms which have two Dwc IPs, this is reserved to be used in the future.

Chapter 2

USB OTG driver

2.1 Overview

Modules

- [USB OTG Controller driver](#)
- [USB OTG Peripheral driver](#)

Data Structures

- struct [usb_otg_descriptor_t](#)
USB OTG descriptor. [More...](#)
- struct [usb_otg_instance_t](#)
USB OTG instance structure. [More...](#)

Macros

- #define [USB_OTG_MSG_COUNT](#) (8)
USB OTG task message queue count.
- #define [USB_OTG_STATUS_HOST_REQUEST_FLAG](#) (0x01U)
USB OTG host request flag.

Typedefs

- typedef void * [usb_otg_controller_handle](#)
USB OTG controller handle type define.
- typedef void(* [usb_otg_callback_t](#))(void *param, uint8_t eventType, uint32_t eventValue)
OTG callback function typedef.

Enumerations

- enum `usb_otg_status_type_t` { ,
 `kOtg_StatusAdpChange` = 0x0002U,
 `kOtg_StatusSrpDet` = 0x0004U,
 `kOtg_StatusVbusVld` = 0x0008U,
 `kOtg_StatusAConn` = 0x0010U,
 `kOtg_StatusBusResume` = 0x0020U,
 `kOtg_StatusBusSuspend` = 0x0040U,
 `kOtg_StatusSe0Srp` = 0x0080U,
 `kOtg_StatusSsendSrp` = 0x0100U,
 `kOtg_StatusSessVld` = 0x0200U,
 `kOtg_StatusBusDrop` = 0x0400U,
 `kOtg_StatusBusReq` = 0x0800U,
 `kOtg_StatusPowerUp` = 0x1000U,
 `kOtg_StatusTimeOut` = 0x2000U,
 `kOtg_StatusBConn` = 0x4000U,
 `kOtg_StatusClrErr` = 0x8000U,
 `kOtg_StatusBSrpDone` = 0x10000U,
 `kOtg_StatusADisconn` = 0x20000U,
 `kOtg_StatusBDisconn` = 0x40000U,
 `kOtg_StatusVbusInvld` = 0x80000U,
 `kOtg_StatusSessInvld` = 0x100000U,
 `kOtg_StatusCheckIdleInAPeripheral` = 0x200000U,
 `kOtg_StatusBHNPFeature` = 0x4000000U,
 `kOtg_StatusChange` = (int)0x80000000U }
 please reference to 7.4 in OTG spec
- enum `usb_otg_device_state_t` { ,
 `kOtg_State_AIdle`,
 `kOtg_State_AWaitVrise`,
 `kOtg_State_AWaitBcon`,
 `kOtg_State_AHost`,
 `kOtg_State_AWaitVfall`,
 `kOtg_State_ASuspend`,
 `kOtg_State_APeripheral`,
 `kOtg_State_AVbusErr`,
 `kOtg_State_BIdleEh`,
 `kOtg_State_BIdle`,
 `kOtg_State_BSrpInit`,
 `kOtg_State_BPeripheral`,
 `kOtg_State_BWaitAcon`,
 `kOtg_State_BHost` }
 Please reference to chapter 7 in OTG spec.
- enum `usb_otg_stack_init_type_t` { ,

```

kOtg_StackHostInit,
kOtg_StackHostDeinit,
kOtg_StackDeviceInit,
kOtg_StackDeviceDeinit }

```

The event value for callback to application when event type is kOtg_EventStackInit.

- enum `usb_otg_event_type_t` {
`kOtg_EventStateChange = 0U,`
`kOtg_EventStackInit }`

The event types for callback to application.

USB OTG APIs

- `usb_status_t USB_OtgInit` (`uint8_t controllerId`, `usb_otg_handle *otgHandle`, `usb_otg_callback_t otgCallbackFn`, `void *callbackParameter`)
Initializes the USB OTG stack.
- `usb_status_t USB_OtgDeinit` (`usb_otg_handle otgHandle`)
Deinitializes the USB OTG stack.
- void `USB_OtgTaskFunction` (`usb_otg_handle otgHandle`)
OTG stack task function.
- void `USB_OtgKhciIsrFunction` (`usb_otg_handle otgHandle`)
OTG KHCI ISR function.
- `usb_status_t USB_OtgBusDrop` (`usb_otg_handle otgHandle`, `uint8_t drop`)
A-device drop bus.
- `usb_status_t USB_OtgBusRequest` (`usb_otg_handle otgHandle`)
bus request.
- `usb_status_t USB_OtgBusRelease` (`usb_otg_handle otgHandle`)
bus request.
- `usb_status_t USB_OtgClearError` (`usb_otg_handle otgHandle`)
clear error.
- `usb_status_t USB_OtgNotifyChange` (`usb_otg_handle otgHandle`, `uint32_t statusType`, `uint32_t statusValue`)
Notify OTG stack about the status changes.

2.2 Data Structure Documentation

2.2.1 struct `usb_otg_descriptor_t`

Data Fields

- `uint8_t bLength`
Size of Descriptor.
- `uint8_t bDescriptorType`
OTG type = 9.
- `uint8_t bmAttributes`
Attribute Fields.
- `uint8_t bcdOTG` [2]
OTG and EH supplement release number in binary-coded decimal.

Typedef Documentation

2.2.1.0.0.1 Field Documentation

2.2.1.0.0.1.1 uint8_t usb_otg_descriptor_t::bmAttributes

D7..3: Reserved (reset to zero) D2: ADP support D1: HNP support D0: SRP support

2.2.2 struct usb_otg_instance_t

Public Member Functions

- [OSA_MSGQ_HANDLE_DEFINE](#) (otgMsgHandleBuffer, [USB_OTG_MSG_COUNT](#),([USB_OTG_MESSAGES_SIZE](#)))
OTG task message queue handle.

Data Fields

- [usb_otg_controller_handle](#) controllerHandle
The low level controller handle.
- [usb_otg_callback_t](#) otgCallback
OTG callback function.
- void * [otgCallbackParameter](#)
OTG callback function parameter.
- const
usb_otg_controller_interface_t * [controllerInterface](#)
controller interface APIs
- uint32_t [otgControllerStatus](#)
please reference to [usb_otg_status_type_t](#)
- uint8_t [otgDeviceState](#)
please reference to [usb_otg_device_state_t](#)
- volatile uint8_t [hasTimeOutMsg](#)
There is timer out message in the message queue.
- volatile uint8_t [hasUpdateMsg](#)
There is update message in the message queue.
- uint8_t [cancelTime](#)
Don't process the timer out message.
- uint8_t [waitInit](#)
Waiting the opposite side board's device stack or host stack initializing.

2.3 Typedef Documentation

2.3.1 typedef void(* usb_otg_callback_t)(void *param, uint8_t eventType, uint32_t eventValue)

This callback function is used to notify application events, the events include [usb_otg_event_type_t](#). This callback pointer is passed when initializing OTG.

Parameters

<i>param</i>	The assigned parameter when initializing OTG.
<i>eventType</i>	Please reference to usb_otg_event_type_t .
<i>event_code</i>	Please reference to usb_otg_device_state_t and usb_otg_stack_init_type_t .

2.4 Enumeration Type Documentation

2.4.1 enum usb_otg_status_type_t

Enumerator

kOtg_StatusAdpChange id
kOtg_StatusSrpDet adp_change
kOtg_StatusVbusVld a_srp_det
kOtg_StatusAConn a_vbus_vld
kOtg_StatusBusResume a_conn
kOtg_StatusBusSuspend a_bus_resume
kOtg_StatusSe0Srp a_bus_suspend
kOtg_StatusSsendSrp b_se0_srp
kOtg_StatusSessVld b_ssend_srp
kOtg_StatusBusDrop b_sess_vld
kOtg_StatusBusReq a_bus_drop
kOtg_StatusPowerUp a_bus_req and b_bus_req
kOtg_StatusTimeOut power_up
kOtg_StatusBConn all the timeout in the state machine
kOtg_StatusClrErr b_conn
kOtg_StatusBSrpDone a_clr_err
kOtg_StatusADisconn b_srp_done
kOtg_StatusBDisconn a_conn(non)
kOtg_StatusVbusInvld b_conn(non)
kOtg_StatusSessInvld a_vbus_vld(non)
kOtg_StatusCheckIdleInAPeripheral b_sess_vld(non)
kOtg_StatusBHNPFeature check the idle timeout when in a_peripheral state
kOtg_StatusChange This status is valid when (1) b_hnp_enable feature is sent when A-device works as host; Or (2) b_hnp_enable feature is received when B-device works as device.

2.4.2 enum usb_otg_device_state_t

Enumerator

kOtg_State_AIdle state state
kOtg_State_AWaitVrise a_idle state

Function Documentation

kOtg_State_AWaitBcon a_wait_vrise state
kOtg_State_AHost a_wait_bcon state
kOtg_State_AWaitVfall a_host state
kOtg_State_ASuspend a_wait_vfall state
kOtg_State_APeripheral a_suspend state
kOtg_State_AVbusErr a_peripheral state
kOtg_State_BIdleEh a_vbus_err state
kOtg_State_BIdle b_idle_eh state
kOtg_State_BSrpInit b_idle or bp_idle state, when the device is peripheral-only B-device it means bp_idle
kOtg_State_BPeripheral b_srp_init or bp_srp_init state, when the device is peripheral-only B-device it means bp_srp_init
kOtg_State_BWaitAcon b_peripheral or bp_peripheral state, when the device is peripheral-only B-device it means bp_peripheral
kOtg_State_BHost b_wait_acon state

2.4.3 enum usb_otg_stack_init_type_t

Enumerator

kOtg_StackHostInit default state
kOtg_StackHostDeinit notify application to initialize host stack
kOtg_StackDeviceInit notify application to de-initialize host stack
kOtg_StackDeviceDeinit notify application to initialize device stack

2.4.4 enum usb_otg_event_type_t

Enumerator

kOtg_EventStateChange OTG state change event, the event values are [usb_otg_device_state_t](#).
kOtg_EventStackInit host/device stack handle event, the event values are [usb_otg_stack_init_type_t](#)

2.5 Function Documentation

2.5.1 usb_status_t USB_OtgInit (uint8_t controllerId, usb_otg_handle * otgHandle, usb_otg_callback_t otgCallbackFn, void * callbackParameter)

This function initializes the USB OTG module specified by the controllerId.

Parameters

in	<i>controllerId</i>	The controller ID of the USB IP. See the enumeration usb_controller_index_t .
out	<i>otgHandle</i>	Return the OTG handle.
in	<i>otgCallbackFn</i>	OTG callback function, it is usb_otg_callback_t .
in	<i>callback-Parameter</i>	The callback parameter.

Return values

<i>kStatus_USB_Success</i>	The OTG is initialized successfully.
<i>kStatus_USB_Invalid-Handle</i>	The otgHandle is a NULL pointer.
<i>kStatus_USB_AllocFail</i>	Allocation memory fail.
<i>kStatus_USB_Error</i>	message queue create fail, controller is not found, controller initialize fail.

2.5.2 **usb_status_t USB_OtgDeinit (usb_otg_handle otgHandle)**

This function deinitializes the USB OTG module specified by the otgHandle.

Parameters

in	<i>otgHandle</i>	the OTG handle.
----	------------------	-----------------

Return values

<i>kStatus_USB_Success</i>	The OTG is initialized successfully.
<i>kStatus_USB_Invalid-Handle</i>	The otgHandle is a NULL pointer.
<i>kStatus_USB_Error</i>	Controller deinitialization fail.

2.5.3 **void USB_OtgTaskFunction (usb_otg_handle otgHandle)**

The function implement the OTG stack state machine. In bare metal environment, this function should be called periodically in the main function. In the RTOS environment, this function should be used as a function entry to create a task.

Function Documentation

Parameters

in	<i>otgHandle</i>	The OTG handle.
----	------------------	-----------------

2.5.4 void USB_OtgKhcilrFunction (usb_otg_handle *otgHandle*)

The function is the KHCI interrupt service routine.

Parameters

in	<i>otgHandle</i>	The OTG handle.
----	------------------	-----------------

2.5.5 usb_status_t USB_OtgBusDrop (usb_otg_handle *otgHandle*, uint8_t *drop*)

This function drop the bus.

Parameters

in	<i>otgHandle</i>	the OTG handle.
in	<i>drop</i>	1 or 0.

Return values

<i>kStatus_USB_Success</i>	Success.
<i>kStatus_USB_Invalid-Handle</i>	The <i>otgHandle</i> is a NULL pointer.
<i>kStatus_USB_Error</i>	The device is not A-device or Send message error.

2.5.6 usb_status_t USB_OtgBusRequest (usb_otg_handle *otgHandle*)

This function can be called in the follow situations:

1. A-device request bus, change from *a_idle* to *a_wait_vrise*.
2. HNP, B-device is in the *b_peripheral* and request the bus.
3. A-device is in the *a_peripheral* and request the bus.
4. B-device request bus (SRP), change from *b_idle* to *b_srp_init*
5. Poll device status, "host request flag" is set.

Parameters

in	<i>otgHandle</i>	the OTG handle.
----	------------------	-----------------

Return values

<i>kStatus_USB_Success</i>	Success.
<i>kStatus_USB_Invalid-Handle</i>	The otgHandle is a NULL pointer.
<i>kStatus_USB_Error</i>	Send message error.

2.5.7 **usb_status_t USB_OtgBusRelease (usb_otg_handle *otgHandle*)**

This function can be called in the follow situations:

1. A-device set the bus request false when in a_idle.
2. A-device release bus when A-device is host (a_host).
3. B-device release bus when B-device is host (b_host).

Parameters

in	<i>otgHandle</i>	the OTG handle.
----	------------------	-----------------

Return values

<i>kStatus_USB_Success</i>	Success.
<i>kStatus_USB_Invalid-Handle</i>	The otgHandle is a NULL pointer.
<i>kStatus_USB_Error</i>	Send message error.

2.5.8 **usb_status_t USB_OtgClearError (usb_otg_handle *otgHandle*)**

This function clears the error.

Parameters

in	<i>otgHandle</i>	the OTG handle.
----	------------------	-----------------

Function Documentation

Return values

<i>kStatus_USB_Success</i>	Success.
<i>kStatus_USB_Invalid-Handle</i>	The otgHandle is a NULL pointer.
<i>kStatus_USB_Error</i>	The device is not in error state or send message error.

2.5.9 `usb_status_t USB_OtgNotifyChange (usb_otg_handle otgHandle, uint32_t statusType, uint32_t statusValue)`

This function notify the [usb_otg_status_type_t](#) and values.

Parameters

in	<i>otgHandle</i>	the OTG handle.
in	<i>statusType</i>	please reference to usb_otg_status_type_t
in	<i>statusValue</i>	the value is 1 or 0

Return values

<i>kStatus_USB_Success</i>	Success.
<i>kStatus_USB_Invalid-Handle</i>	The otgHandle is a NULL pointer.
<i>kStatus_USB_Error</i>	Send message error.

2.6 USB OTG Controller driver

2.6.1 Overview

Data Structures

- struct `usb_otg_msg_t`
OTG stack task message. *More...*

Macros

- #define `USB_OTG_TIMER_A_WAIT_VRISE_TMR` (100U)
a_wait_vrise_tmr in OTG spec, VBUS Rise Time, 100ms
- #define `USB_OTG_TIMER_A_WAIT_VFALL_TMR` (1000U)
a_wait_vfall_tmr in OTG spec, Session end to VOTG_VBUS_LKG, 1sec
- #define `USB_OTG_TIMER_A_WAIT_BCON_TMR` (2000U)
a_wait_bcon_tmr in OTG spec, Wait for B-Connect, 1.1sec ~ 30^15sec
- #define `USB_OTG_TIMER_A_AIDL_BDIS_TMR` (500U)
a_aidl_bdis_tmr in OTG spec, A-Idle to B-Disconnect, 200ms ~ infinity
- #define `USB_OTG_TIMER_B_ASE0_BRST_TMR` (155U)
b_ase0_brst_tmr in OTG spec, A-SE0 to B-Reset, 155ms ~ 200ms
- #define `USB_OTG_TIME_B_DATA_PLS` (7U)
TB_DATA_PLS in OTG spec, Data-Line Pulse Time, 5ms ~ 10ms.
- #define `USB_OTG_TIME_B_DATA_PLS_MIN` (5U)
TB_DATA_PLS in OTG spec, Data-Line Pulse Time's minimum value.
- #define `USB_OTG_TIME_B_DATA_PLS_MAX` (10U)
TB_DATA_PLS in OTG spec, Data-Line Pulse Time's maximum value.
- #define `USB_OTG_TIME_A_BCON_LDB` (100U)
TA_BCON_LDB in OTG spec, B-Connect Long Debounce, 100ms ~ infinity.
- #define `USB_OTG_TIME_A_BCON_SDB` (1U)
TA_BCON_SDB in OTG spec, B-Connect Short Debounce, 2.5us ~ infinity.
- #define `USB_OTG_TIME_B_SSEND_SRP` (1500U)
TB_SSEND_SRP in OTG spec, Session end to SRP init, 1.5sec ~ infinity.
- #define `USB_OTG_TIME_B_SE0_SRP` (1000U)
TB_SE0_SRP in OTG spec, SE0 Time Before SRP, 1sec ~ infinity.
- #define `USB_OTG_TIME_B_AIDL_BDIS` (100U)
TB_AIDL_BDIS in OTG spec, A-Idle to B-Disconnect, 4ms ~ 150ms.
- #define `USB_OTG_TIME_A_BIDL_ADIS` (190U)
TA_BIDL_ADIS in OTG spec, B-Idle to A-Disconnect, Used by an A-device to determine when the B-device has finished being host, 155ms ~ 200ms.
- #define `USB_OTG_TIME_WAIT_DEVICE_INIT` (200U)
wait another device initialize device stack before initializing the host stack
- #define `USB_OTG_TIME_WAIT_BHOST` (1000U)
delay this time before check idle in a_peripheral state, wait another device initialize host stack

USB OTG Controller driver

Enumerations

- enum `usb_otg_control_t` { ,
 `kOtg_ControlPullUp`,
 `kOtg_ControlPullDown`,
 `kOtg_ControlResume`,
 `kOtg_ControlAdpPrb`,
 `kOtg_ControlDataPulse`,
 `kOtg_ControlHNPCheckEnable`,
 `kOtg_ControlSetTimer`,
 `kOtg_ControlCancelTimer`,
 `kOtg_ControlRequestStatus`,
 `kOtg_ControlUpdateStatus` }
 The control types.
- enum `usb_otg_pull_control_t` {
 `kOtg_PullDp` = 0x01U,
 `kOtg_PullDm` = 0x02U }
 Pull up/down parameters.

2.6.2 Data Structure Documentation

2.6.2.1 struct `usb_otg_msg_t`

Data Fields

- `uint32_t otgStatusType`
 The status types please reference to [usb_otg_status_type_t](#).
- `uint32_t otgStatusValue`
 The status values.

2.6.3 Macro Definition Documentation

2.6.3.1 #define `USB_OTG_TIME_B_DATA_PLS` (7U)

generate the data pulse using this time value.

2.6.4 Enumeration Type Documentation

2.6.4.1 enum `usb_otg_control_t`

Enumerator

`kOtg_ControlPullUp` control vbus

kOtg_ControlPullDown pull dp/dm up
kOtg_ControlResume pull dp/dm down
kOtg_ControlAdpPrb do resume
kOtg_ControlDataPulse probe adp
kOtg_ControlHNPCheckEnable generate data pulse
kOtg_ControlSetTimer start to check HNP
kOtg_ControlCancelTimer start timer
kOtg_ControlRequestStatus cancel timer
kOtg_ControlUpdateStatus request the status values [usb_otg_status_type_t](#)

2.6.4.2 enum usb_otg_pull_control_t

Enumerator

kOtg_PullDp pull DP line
kOtg_PullDm pull DM line

USB OTG Peripheral driver

2.7 USB OTG Peripheral driver

2.7.1 Overview

Functions

- [usb_status_t USB_OtgPeripheralEnable](#) (void)
Enable OTG peripheral.
- [usb_status_t USB_OtgPeripheralDisable](#) (void)
Disable OTG peripheral.
- [usb_status_t USB_OtgPeripheralGetStatus](#) (uint32_t statusType, uint32_t *statusValue)
Get the peripheral status.
- [usb_status_t USB_OtgPeripheralControl](#) ([usb_otg_controller_handle](#) controllerHandle, uint32_t controlType, uint32_t controlValue1, uint32_t controlValue2)
Control the peripheral.

2.7.2 Function Documentation

2.7.2.1 [usb_status_t USB_OtgPeripheralEnable](#) (void)

This function enable OTG peripheral function.

Return values

<i>kStatus_USB_Success</i>	success.
<i>other</i>	values Fail.

2.7.2.2 [usb_status_t USB_OtgPeripheralDisable](#) (void)

This function disable OTG peripheral function.

Return values

<i>kStatus_USB_Success</i>	success.
<i>other</i>	values Fail.

2.7.2.3 [usb_status_t USB_OtgPeripheralGetStatus](#) (uint32_t *statusType*, uint32_t **statusValue*)

This function is nonblocking, return the result immediately.

Parameters

in	<i>statusType</i>	Please reference to usb_otg_status_type_t .
out	<i>statusValue</i>	The status value.

Return values

<i>kStatus_USB_Success</i>	success.
<i>other</i>	values Fail.

2.7.2.4 usb_status_t USB_OtgPeripheralControl (usb_otg_controller_handle controllerHandle, uint32_t controlType, uint32_t controlValue1, uint32_t controlValue2)

This function control the peripheral to implement the different functions.

Parameters

<i>controller-Handle</i>	The controller instance handle.
<i>controlType</i>	The control type, please reference to usb_otg_control_t .
<i>controlValue1</i>	The control value, it is 0 or 1 usually.
<i>controlValue2</i>	It only be used in the kOtg_ControlRequestStatus control now.

Return values

<i>kStatus_USB_Success</i>	success.
<i>other</i>	values Fail.



USB OTG Peripheral driver



Chapter 3

USB OS Adapter

Please reference to MCUXpresso SDK API Reference Manual.



Chapter 4

Data Structure Documentation

4.0.3 usb_serial_port_config_t Struct Reference

serial port configuration structure.

```
#include <usb_serial_port.h>
```

Data Fields

- uint32_t [baudRate_Bps](#)
LPUART baud rate.
- uint8_t [isMsb](#)
Data bits order, LSB (default), MSB.
- uint8_t [enableTx](#)
Enable TX.
- uint8_t [enableRx](#)
Enable RX.

4.0.3.1 Detailed Description



How to Reach Us:

Home Page:

nxp.com

Web Support:

nxp.com/support

Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters that may be provided in NXP data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including "typicals," must be validated for each customer application by customer's technical experts. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address:

nxp.com/SalesTermsandConditions.

While NXP has implemented advanced security features, all products may be subject to unidentified vulnerabilities. Customers are responsible for the design and operation of their applications and products to reduce the effect of these vulnerabilities on customer's applications and products, and NXP accepts no liability for any vulnerability that is discovered. Customers should implement appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP, the NXP logo, NXP SECURE CONNECTIONS FOR A SMARTER WORLD, Freescale, the Freescale logo, Kinetis, Processor Expert, and Tower are trademarks of NXP B.V. All other product or service names are the property of their respective owners. Arm, Cortex, Keil, Mbed, Mbed Enabled, and Vision are trademarks or registered trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. The related technology may be protected by any or all of patents, copyrights, designs and trade secrets. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates. The Power Architecture and Power.org word marks and the Power and Power.org logos and related marks are trademarks and service marks licensed by Power.org.

© 2018 NXP B.V.