# AVR277: On-The-Go (OTG) add-on to USB Software Library

8-bit AVR® Microcontrollers

AT90USB647
AT90USB1287

Application Note

## 1. Introduction

This document describes the new features brought by the OTG working group and how they are integrated in the AT90USBxxx USB software library, illustrating how to develop customizable USB OTG applications.

### 1.1 Intended Audience

This document is written for the software developers to help in the development of the OTG applications for the AT90USBxxx. It assumes that readers are familiar with the AT90USBxxx architecture and its USB Software library (not OTG but including Host and Device mode). A minimum knowledge of the USB specifications is also required to understand the content of this document.

### 1.2 Overview

The AT90USBxxx software library has been designed to minimize the complexity of USB development (and especially enumeration stage) from software designers.

The aim of this document is to present the integration of the OTG features in this library. It describes the main files and explain how to customize the firmware for the user to build his own application.

This software package also provides an OTG application example (template demonstration sofware) to illustrate the usage of this library. Note that this firmware is ready to pass the USB Device and/or OTG compliance tests.

### 1.3 References

- USB Specification : http://www.usb.org/developers/docs/
- OTG Supplement and Errata : http://www.usb.org/developers/onthego/
- OTG Compliance Plan : http://www.usb.org/developers/onthego/

7719A–USB–07/07

# 2. About OTG

## 2.1 Overview

### 2.1.1 Need

USB has become a popular interface to exchange data between a Host PC and peripherals. Due to its low cost and high speed, a large number of portable devices have a peripheral USB port. Although communications between two portable devices would sometimes be useful, the original USB specification only allowed communications between a host and a peripheral port.

For this reason, the On-The-Go supplement has been developped. It enables USB connectivity between portable devices by specifying smaller connectors, embedded host capability for portable devices, low power features, etc.

### 2.1.2 Main Features

What's new with OTG ? Just look at the list:

- Embedded Host capability
- Full-Speed (12Mbits/s) operation as a Peripheral, High-Speed (480Mbits/s) optional
- Full-Speed operation as an Host, Low-Speed (1.5Mbits/s) and High-Speed optional
- Targeted Peripheral List
- Session Request Protocol
- Host Negociation Protocol
- One and only one on-board receptacle (for one OTG port) : Micro-AB (Mini-AB is obsolete)
- Low power Vbus features
- Means for communicating messages to user

Be aware that High Speed operation is not supported by our AT90USBxxx products.

### 2.1.3 Terminology

To facilitate understanding of this document, it is advised to get informed about the minimum required terminology given below :

- **Device** : since OTG concerns only dual-role-devices, the word "Device" must not be employed to indicate a role in a connection. "Device" is simply a general purpose name for the application (a PDA for example)
- **Host** : this is one role that can have an OTG device
- **Peripheral** : this is the other role that can have an OTG device
- **Micro-AB** : the OTG has introduced the Micro-AB receptacle, able to accept both Micro-A and Micro-B plugs. Standards A and B receptacles/plugs are not supported by OTG
- **Mini-AB** : these connectors have been introduced before Micro-AB, but have been declared as obsolete after the Micro-AB introduction. However, this notation can remain in some documents.
- **ID** : fifth pin of the Micro-AB receptacle, that makes the user able to identify the plug inserted (Micro-A or Micro-B, that also include this signal)
- **A-Device** : device with a Micro-A plug inserted into its Micro-AB receptacle. By default (at start-up) it becomes Host (A-Host state)

- **B-Device** : device with a Micro-B plug inserted into its Micro-AB receptacle. By default (at start-up) it becomes Peripheral (B-Peripheral state)
- **Vbus** : power line of the USB / OTG bus. Several requirements have been specified for it
- **Session** : defined by the period of time that Vbus is above the session valid threshold specified by the OTG supplement
- **SRP** : **S**ession **R**equest **P**rotocol ; initiated by the B-Device when a session is closed, that allows it to request the opening of a new session
- **HNP** : **H**ost **N**egociation **P**rotocol ; initiated by A-Device, that allows the two devices to invert their role “on the go”, i.e. without any hardware action. After this event, the A-Host becomes A-Peripheral, and the B-Peripheral becomes B-Host.
- **SOF** : **S**tart **O**f **F**rame ; short packet sent by the Host over the bus every time a new frame begins (every 1ms at Full-Speed, every 125µs at High-Speed)
- **Suspend** : bus condition defined by an idle state on the bus during more than 3ms.
- **USB-IF** : USB Implementers Forum : consortium of companies that have founded the USB specifications, and that define the compliance tests.
- **TPL** : Targeted Peripheral List : list of the VID/PID of the peripherals supported by the Host

## 2.2 Features description

Below are described the OTG features as specified in the OTG supplement. The way they have been coded and customized in the Atmel USB OTG firmware will be described in the next section.

### 2.2.1 Targeted Peripheral List

When acting as a Host (embedded), an OTG device is only required to support a specified list of devices that are identified by their VID/PID values. This is the “Targeted Peripheral List” (TPL), and it can contain only one device.

The TPL shall not use USB classes (with no specific VID/PID) or “similar devices” as selection mean to accept a device connection (for example, if an Host supports HID mice, it must only accept the mice with a VID/PID that matches the TPL).

During compliance tests, the laboratory will test inter-operability with all the devices of the Targeted Peripheral List.

### 2.2.2 Power considerations

Several requirements have been specified for the power delivery and threshold voltages.

A primary rule is that Vbus is always supplied by the A-Device, even if the role is being inverted (and that the A-Device is in the A-Peripheral state).

#### *2.2.2.1 Delivery conditions*

An A-Device has different possibilities for supplying Vbus :

- Insertion based Vbus : Vbus is turned ON continuously as long as a Micro-A plug is inserted into the receptacle
- Usage based Vbus : Vbus is turned ON on request (even if both devices are physically connected together) : user can request Vbus delivery on the A-Device side (by a software feature), but the B-Device can also request Vbus from A-Device by itself (or by user action), by initiating a SRP (See 2.2.3).

#### 2.2.2.2 *Voltage*

A session is valid for an A-Device when Vbus is above *Va_sess_vld* (threshold between 0.8 and 2.0V). For a B-Device this threshold is *Vb_sess_vld*, that can be between 0.8 and 4.0V. That means that a B-Device must connect (i.e. enable its pull-up) before Vbus is greater than *Vb_sess_vld(max)* (4.0V).

To ensure correct bus powering, the A-Device must consider that Vbus is valid (that means no delivery problems, no current overload) if Vbus is above *Va_vbus_vld* threshold (4.4V).

#### 2.2.2.3 *Current*

The A-Device must be able to supply at least 8mA.

Any B-Device in unconfigured state must not draw more than 150µA.

Any OTG device must have a capacitance on its Vbus line between 1µF and 6.5µF, to limit inrush currents but ensure power stability.

An OTG device should have a mean to know if, while it is in Host mode, the Peripheral is drawing more current than available. This may be an “Overcurrent” signal from the power supply block.

### 2.2.3 SRP

When two devices are connected together, and the A-Device does not supply Vbus (or has stopped to supply Vbus), the B-Device can still request it to (re-)enable Vbus in order to start a new session. This is done through the SRP protocol, that is very simple. This consists in two pulses that are sent over the bus lines in order to be detected by the A-Device.

The B-Device may first send a pulse on the D+ line by powering its pull-up. This pulse must have a duration between 5 and 10ms. After that, if there is no reaction from the A-Device (Vbus OFF), the B-Device may send a pulse on the Vbus line (through a resistor), that must rise higher than *Vb_otg_out* level (2.1V).

The B-Device is required to wait for at least 6 seconds before considering that the A-Device has not responded. Conversely, once the A-Device has detected the SRP, it has 5 seconds to deliver Vbus supply, and once Vbus is turned ON, it must keep it available during at least 1 second, waiting for the B-Device to connect.

### 2.2.4 HNP

The Micro-AB receptacle allows any OTG device to be a Peripheral or a Host by default, since both Micro-A plug or Micro-B plug can fit into this receptacle. But some applications shall require to exchange this role (either if the user default connection is wrong, or if the application needs to exchange role for any reason at any moment).

The HNP protocol allows the devices to exchange role without swapping the cable. This protocol can be initiated only between devices that support it. The B-Device informs the A-Device about its capabilities with the OTG descriptors (See 2.2.6). HNP can be divided in two protocols :

1. Hardware side : the role exchange protocol is mainly performed in hardware with specific lines states
2. Software side : the role exchange results of a preliminary software negociation

#### 2.2.4.1 *Hardware side*

First, the A-Host must let the bus in idle state (no activity, no SOF). After *Ta_aidl_bdis(min)* and before *Ta_aidl_bdis(max)* (resp. 5 and 150ms) of idle state, the B-Peripheral must disconnect. When the A-Host detects this disconnect, it connects its own pull-up. Now the roles are exchanged ; and the B-Peripheral that has become B-Host must send a reset on the bus under *Tb_acon_se0* (1ms) after the A-Peripheral connection has been detected. Now the bus can be used. Once the B-Host has finished using the bus, it lets it idle. After the *Ta_aidl_bdis* delay, the A-Peripheral must release its pull-up and, if bus is not needed, release Vbus delivery. If Vbus is still present, the B-Device has the possibility to turn on its pull-up if it needs to return to Peripheral state (otherwise the session will be closed by the A-Device).

#### 2.2.4.2 *Software side*

As described previously, the Host initiates the HNP by setting the bus in idle state, but it is the Peripheral that really starts the exchange by disconnecting itself. And this agreement must be prepared by software... First, the Peripheral informs the Host of its HNP support, thanks to the OTG descriptor (See 2.2.6). After that, the Host informs the Peripheral that it also supports the HNP protocol, by sending a *Set_Feature(a_hnp_support)* command (See 2.2.7). If the Host supports the Peripheral, it will start using the bus, but the Peripheral is allowed to initiate an HNP later. If the Host does not support the Device (not in targeted peripheral list), it must initiate the HNP now. Once the decision is taken to start the HNP, and before stopping activity on bus, the Host must send a Set_Feature(b_hnp_enable) to the Peripheral to enable it to respond to the idle state by disconnecting its pull-up.

Several applications and special cases where HNP is needed are listed and tested in the OTG compliance plan and the OTG supplement.

### 2.2.5 Messaging

The “No Silent Failure” specification of the OTG supplement implies that an OTG compliant device must support specific means for communicating error or warning messages to the user.

Several cases where messaging should be used are listed in the OTG supplement and in the compliance plan documents. To be OTG compliant, a device must support at least the 3 following messages :

– “**Device Not Attached/Responding**” : displayed by a device that detects that the connected device does not work as expected, or does not respond to requests

– “**Attached Device Unsupported**” : displayed by a device when the connected device does not comply with its requirements (not in the TPL, overcurrent condition, etc.)

– “**Unsupported Hub Topology**” (see note below) : used when the A-Device is connected to a hub and does not support it. Please note that this message should not be required if the device does not support any hub in its targeted peripheral list, but according to the USB-IF, hub recognition must be handled and this message must be supported for compliance.

That means that an OTG device must provide at least 3 LEDs to differentiate the messages. A LCD display remains the best solution, especially if one is already included in the application.

Additional messages can be used to help the user understand a failure.

### 2.2.6 OTG Descriptor

During the enumeration process, the host asks the device several descriptor values to identify it and load the correct drivers. To be recognized by the host, each USB device should have at least the following descriptors :

– Device descriptor

– Configuration descriptor

– Interface descriptor

– Endpoint descriptor

– String descriptor

– Class-specific descriptors

The OTG supplement introduces the OTG Descriptor. This descriptor is sent by the B-Device after a *Get_Descriptor(config)* request (for example inserted between the configuration and the first interface descriptors).

**Table 2-1.** OTG Descriptor

| Offset | Field | Size | Description |
|---|---|---|---|
| 0 | bLength | 1 | Descriptor size (0x03) |
| 1 | bDescriptorType | 1 | OTG descriptor (0x09) |
| 2 | bInterfaceNumber | 1 | Attribute fields :<br>D7...2 : reserved<br>D1 : HNP Support<br>D0 : SRP Support |

Notes: 1. The *HNP Support* bit is set if the device supports HNP.
2. The *SRP Support* bit is set if the device supports SRP.

### 2.2.7 Set_Feature commands

Here are described the three OTG cases with the Set_Feature() command.

**Table 2-2.** Set_Feature command

| bmRequestType | bRequest | wValue | wIndex | wLength | Data |
|---|---|---|---|---|---|
| 0x00 | SET FEATURE (0x03) | Feature Selector | 0x00 | 0x00 | None |

**Table 2-3.** Feature Selector

| Feature Selector | Value |
|---|---|
| b_hnp_enable | 3 |
| a_hnp_support | 4 |
| a_alt_hnp_support | 5 |

The two first features have been described in section 2.2.4. More precisions (special cases, acceptance conditions) can be found in the OTG supplement.

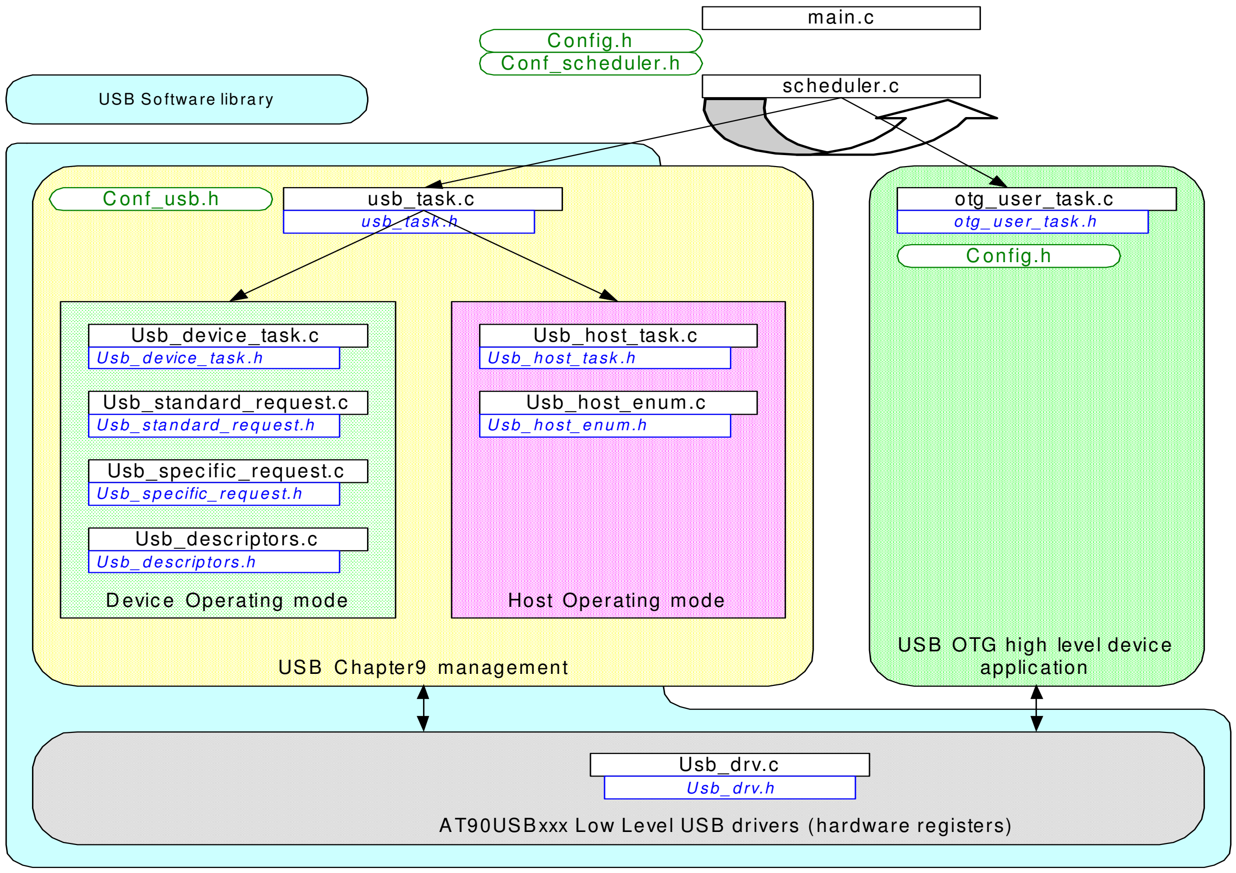
The third feature *a_alt_hnp_support* will not be explained here because it concerns an Host that have an alternate port that supports HNP (and AT90USBxxx products only have one port).

These features can only be cleared on a bus reset or when a session ends. Using a *Clear_Feature()* command has no effect on them.

# 3. Firmware Architecture

As shown in the figure below, the architecture of the USB firmware does not change with the OTG add-on. This document assumes that the reader is familiar with this architecture.

**Figure 3-1.** AT90USBxxx USB OTG Firmware Architecture



The OTG features have been included with this firmware and all files have been impacted. Major changes are detailed in the section 3.1.

## 3.1 OTG Upgrade Listing

The file architecture has not been modified in relation to the first USB library release. But changes have been done on almost all the files, and on the state machine.

### 3.1.1 Requests

Several actions can be initiated at any moment for an OTG connection, like “Initiate a SRP”, “Turn ON Vbus”, “Suspend Bus”, etc.

User can call the functions “Set_user_request_xxxx()”, where xxxx is the action requested. All existing functions are defined in the file “usb_task.h”.

### 3.1.2 Timings

The OTG supplement specifies several delays for events and failures management (for example, the maximum time that a B-Device must wait for Vbus from A-Device after having sent an SRP, before considering the operation as "aborted"). These timings are mandatory, and cannot be handled by a simple SOF counter because several events must be timed without bus activity.

One 16 bits timer (one among two) has been reserved for this operation. It generates an interrupt every 2ms, and each delay required is managed by a specific counter variable. See at the end of "usb_task.c" for the function.

### 3.1.3 Peripheral mode

In Peripheral mode, the OTG Device is managed by a state machine using the following states, coded in the "usb_device_task.c" file :

– **B_IDLE** : no session currently open, the device waits for Vbus to rise, or waits for a SRP request from the user

– **B_SRP_INIT** : a SRP is being performed on the bus, software waits it has finished to come back to B_IDLE (wait Host response)

– **B_PERIPHERAL** : a session is open, the device is connected. This state waits for events (even if major events are interrupt-handled) or user requests

– **B_HOST** : state entered after a HNP success : the B-DEVICE has become Host ; handle user requests

– **B_END_HNP_SUSPEND** : state entered when a role exchange ends (device is B-HOST, and receives an user request to disconnect)

Be aware that almost all events are interrupt handled, in "usb_task.c".

Moreover, whatever the current state is executing, the Peripheral task checks if a request has been received on the Control endpoint.

### 3.1.4 Host mode

The current state machine has been enhanced, receiving five additional states :

– **A_PERIPHERAL** : state entered when the Device has entered Peripheral role after an HNP. In this state, the program ends the session when a suspend condition is detected (B-HOST finished its bus usage). Also ends the session on user request (disconnect or Vbus toggle requests).

– **A_INIT_HNP** : device enters this state when user requested an HNP ; handles SetFeature commands, failures...

– **A_SUSPEND** : A-HOST enters this state with a suspend condition detected after a HNP request done (so waiting for Peripheral disconnection). Also detects HNP time-out and upstream-resume signalling.

– **A_END_HNP_WAIT_VFALL** : this state is entered when the A-PERIPHERAL detects a Suspend condition. In this case the OTG device does not come back into the initial configuration (A-HOST), but ends the session by turning OFF Vbus

– **A_TEMPO_VBUS_DISCHARGE** : this state is entered when shutting down Vbus, and when it has been detected under the Vbus_valid threshold. Here we wait for a delay (that can be customized in "usb_host_task.h" as *TA_VBUS_FALL*, by multiple of 2ms), because the OTG peripheral can remain attached while Vbus is decreasing. If the attached delay is too long (i.e. falling time too slow) a SRP will be detected, and the session will never be open again.

Moreover, several OTG features and user requests have been implemented into the other states of the machine (that were already existing without OTG).

## 3.2 About the sample application

The sample application does not reflect the OTG firmware complexity. Users must press a push-button on the Peripheral side to toggle a LED located on the Host board.

These are the followings actions possible for the user :

- **Host side**
  - turn ON/OFF Vbus by pushing HWB button : this is only possible from the device that have the Micro-A plug inserted. That action turns OFF Vbus and closes the current session.
  - request for a HNP by pushing CENTER button : from the device that is in A-HOST state, this will conduce to role exchange. Once the role exchanged (A-PERIPH and B-HOST), pressing this key on the B-HOST side will lead to a session end (Vbus OFF).
- **Peripheral side**
  - send a SRP by pushing HWB button : only possible for a default Peripheral (Micro-B plug inserted), and when Vbus is OFF. Note that the Host that receives the SRP is required to exchange role immediately (using HNP). So sending a SRP from the default Peripheral will conduce the A-HOST to become the Peripheral and the default Peripheral to become the B-HOST.
  - toggle LED by pushing CENTER button : only possible when enumeration is done, but in any Peripheral mode (A-PERIPH or B-PERIPH).

This sample application uses a simple HID mouse USB configuration. This means that the Peripheral only needs one INTERRUPT endpoint to send reports to the Host. The report structure is identical to a mouse report : the information that indicate that the user is pressing the button is contained in the bitfield normally reserved to the “Left click” mouse event. Pushbutton debouncing is automatically handled since the refresh rate (endpoint polling period) is 50ms.

### 3.2.1 Role exchanging

The “role exchanged” state is deceptive, since the ID pin state does not change when the device changes its role, whereas in the “usb_task.c” file the choice between usb_host_task() and usb_device_task() is done in function of that ID level!

So on both sides (Peripheral / Host), a special state has been created to solve this problem. And in each state, the program calls the alternate task...For example, for an A-Device, after an HNP, the usb_task() still calls the usb_host_task() function, and the state entered is A_PERIPHERAL (determined upon HNP success interrupt). But in that state, the program calls the usb_device_task()....It’s the same thing in B-Device mode, when the state is B_HOST, and where a call is usb_host_task() is done.

# 4. Configuring the USB OTG software library

## 4.1 Global configuration

All common configuration parameters for the application are defined in the "config.h" file (XTAL frequency, CPU type...). Modules specific parameters are defined in their respective configuration files.

## 4.2 Scheduler configuration

The sample application provides a simple tasks scheduler that allows the user to create and add applicative tasks without modifying the global application architecture and organisation. This scheduler just calls all predefined tasks in a predefined order without any pre-emption. A task is executed until its finished, then the scheduler calls the next task.

The scheduler tasks are defined in the "conf_scheduler.h" file, where the user can declare its tasks functions.

For the sample USB dual role application the following scheduler configuration parameters are used:

```
#define Scheduler_task_1_init   usb_task_init
#define Scheduler_task_1        usb_task
#define Scheduler_task_2_init   otg_user_task_init
#define Scheduler_task_2        otg_user_task
```

The Scheduler_task_X_init functions are executed only once upon scheduler startup whereas the Scheduler_task_X functions are executed in an infinite loop.

## 4.3 USB library configuration

The USB library can be configured thanks to the "conf_usb.h" file. This file contains both USB modes configuration parameters for device and host. The configuration file is split into tree sections for device, host, and now also OTG configuration parameters.

### 4.3.1 Peripheral configuration

All the parameters of the USB Device part remain unchanged, please refer to the USB Library description document for more details. Below are the main points to verify :

– In OTG mode, the USB_DEVICE_FEATURE must be enabled.

– The user application can still execute specific functions upon events thanks to the user defined actions of the "conf_usb.h" file.

– User must also define the endpoints used by the configuration of the Peripheral.

– Finally, the "usb_descriptors.h" file must be correctly updated.

### 4.3.2 Host configuration

Same remark that for the USB Device side, ensure that :

– The USB_HOST_FEATURE must be enabled in OTG mode.

– The HOST_STRICT_VID_PID_TABLE must be enabled in OTG mode, since the Targeted Peripheral List does not allow connexion acceptance based on Class, but only on VID and PID knowledge. Just fill in the VID_PID_TABLE with the devices that your application supports.

  - The user application can execute specific functions upon events thanks to the user defined actions of the "conf_usb.h" file.
  - All other parameters can be used as indicated in the USB Library description.

### 4.3.3 OTG configuration

#### *4.3.3.1 Descriptor*

User is free to modify the "usb_descriptors.h" file that contains as well device general information than class specific specifications.

Moreover the OTG has brought a new descriptor, which contains one efficient byte field, bmAttributes. For an OTG compliant device this descriptor OTG_BMATTRIBUTES should be equal to (HNP_SUPPORT | SRP_SUPPORT). But user can choose to disable HNP to make a simple "SRP Capable" Peripheral.

#### *4.3.3.2 Mode configuration*

The following parameters concern new features brought by OTG. For almost all parameters, you have to choose between ENABLED and DISABLED. Others cases are specified. All are included in the "conf_usb.h" file of the project.

- **USB_OTG_FEATURE** : must be set to *ENABLED* for OTG mode support
- **OTG_VBUS_AUTO_WHEN_A_PLUG** :
  - set to *ENABLED* if user wants a "Insertion Based Vbus Supply", that means that the A-Device will keep VBUS turned ON continuously, while the A-plug is inserted.
  - the *DISABLED* value codes for "Usage Based Vbus Supply", that means that Vbus is supplied on user request : either on a SRP received from the B-PERIPH, or from a software request on the A-HOST side (pushbutton, etc.).
- **OTG_MESSAGING_OUTPUT** : defines the messaging method used on Device :
  - set to *OTGMSG_ALL* if all programmed messages are needed (both events messages like "Device connected" and mandatory failure messages like "Unsupported Device". In this case, 5 functions that are called by the OTG library must be defined by the user : Initialisation function, Display/Clear message (event, failure) functions. In the OTG library, messages are identified thanks to an ID number, that are available in the file "usb_task.h".
  - set to *OTGMSG_FAIL* if only failure messages are required. The same functions that for OTGMSG_ALL have to be defined except those that concern event messages...
  - set to *OTGMSG_NONE* if messages are not required in this application. In this case the device is not OTG compliant. Nothing else has to be done.
- **OTG_USE_TIMER** : defines which 16 bits timer will be used by the OTG library : must be either *TIMER16_1* or *TIMER16_3*.
- **USE_TIMER16** : defines if the global application uses another timer or not :
  - set to *BOTH_TIMER16* if your application uses the other 16 bits timer for a general purpose application
  - set to *OTG_USE_TIMER* if the remaining 16 bits timer is not used by user
- **OTG_ENABLE_HNP_AFTER_SRP** :
  - if *ENABLED*, the A-Device will automatically initiate a HNP (and become A-PERIPH) if the B-Device has sent a SRP and has connected within the correct delay once Vbus has been supplied. Default (and OTG compliant) value is ENABLED.

– if *DISABLED*, the A-Device will start a normal session (supply Vbus and be A-HOST) when it will receive a SRP from the B-Device

- **OTG_ADEV_SRP_REACTION** : defines the SRP pulse for which the A-Device will react and turn ON Vbus. Possible values are : *VBUS_PULSE* or *DATA_PULSE*.

##### *4.3.3.3 Power*

- An OTG compliant device should have a power circuitry able to inform the Host of overcurrent conditions. If your board has such a system, you must define in “conf_usb.h” the macro **Is_vbus_overcurrent()** that returns TRUE or FALSE according to the situation.
- As explained at the section 3.1.4, a session closing failure may happen if the VBUS falling time is too slow, and that the Host detects the end of the session before the Peripheral that remains attached. This may lead to a false SRP detection, that will open a new session automatically...To prevent that the Host waits for a specified delay between the Vbus OFF detection and releasing the USB macro and enabling it to detect new SRP. This delay is 100ms by default, and should only be modified in case of failure. It is specified by **TA_VBUS_FALL** (16 bits constant, where 1 unit = 2ms) in the “usb_host_task.h” file.

# 5. Using the OTG software library within high level USB OTG applications

## 5.1 Standard functions

The high-level library management keeps all the properties of the previous USB release (see appnote AVR329).

So all functions and features described remain available, and have the same effect and behavior.

Developping an OTG applications is not so different than an application that supports both embedded Host and Peripheral mode, since the OTG supplement mainly concerns the role exchange and the session control.

## 5.2 OTG functions

### 5.2.1 User Requests

Additionnaly, this library introduces the notion of user request. Such a request does not have a direct effect on the USB macro but this event is saved and will be processed as soon as possible. It allows user to modify bus or session state, or device role, without care for low-level management, that will be handled by the library.

#### *5.2.1.1 Set_user_request_vbus()*

In A-Device mode, this request controls Vbus delivery. If the configuration word OTG_VBUS_AUTO_WHEN_A_PLUG is enabled (continuous Vbus supply while A-plug inserted), this request has only an effect in A-PERIPHERAL state, to close the current session (and stop Vbus for a short time). If OTG_VBUS_AUTO_WHEN_A_PLUG is disabled, this request can be used to toggle Vbus state (and close session) at any moment.

This request has no effect in Peripheral mode.

#### 5.2.1.2 *Set_user_request_hnp()*

In A-HOST mode, this request will make the Host initiate a HNP (if supported by the B-Device), that means that firmware will send Set_Feature(b_hnp_enable) command, put bus into Idle state and handle failure cases or role exchange in case of success.

In B-HOST mode, this request conduces to a new role exchange that will lead in the current version of firmware to a session closure.

#### 5.2.1.3 *Set_user_request_suspend()*

On A-HOST or B-HOST, this request has the same effect than *Set_user_request_hnp(),* because the Idle state (bus Suspended) is not really used on OTG devices. Either a Suspend condition leads to an HNP, or it must be advantageously replaced by a Session shut-down, that saves more power, since Vbus is OFF and OTG B-Device will only need to send a SRP to restart a session, instead of an upstream resume (remote wake-up).

However, in “usb_task.c” file, an explanation is given to implement an upstream resume in Peripheral mode (that can also be used in a non-OTG application).

#### 5.2.1.4 *Set_user_request_disc()*

In B-PERIPHERAL state, this request makes the Device disconnect.

In A-PERIPHERAL state, this request leads to session closure (like a vbus request).

In B-HOST state, this request leads also to session closure.

#### 5.2.1.5 *Set_user_request_srp()*

In Peripheral mode, this request must be used only if no session is currently open (Vbus = OFF). When reading this request, the OTG firmware will initiate a SRP, waits for Vbus delivery, and also handle failure cases (delays, no response...).

This request has no effect in Host mode.

### 5.2.2 Device information

OTG supplement adds some device specific features, that can be considered as “device information”, since they are integrated into the descriptors, such as HNP or SRP support.

Some pieces of information are available for user :

- ***Host mode***
  - Is_peripheral_otg_device() : returns TRUE if the connected device is an OTG device (has an OTG descriptor), else return FALSE
  - Is_host_session_started_srp() : returns TRUE if the current session has been initiated by B-Device with a SRP, else returns FALSE

- ***Peripheral mode***
  - Is_host_requested_hnp() : returns TRUE if A-Device has sent a Set_Feature(b_hnp_enable), else returns FALSE.

## 6. Coding Style

The coding style explained hereunder is important to understand the firmware:

- Defined contants use caps letters.

```
#define FOSC 8000
```

- Macros Functions use the first letter as cap

```
#define Is_usb_sof() ((UDINT &   MSK_SOFI)    ? TRUE: FALSE)
```

- The user application can execute its own specific instructions upon each usb events thanks to hooks defined as following in *usb_conf.h.*

```
#define Usb_sof_action()         sof_action()
```

```
Note: The hook function should perform only short time requirement
operations !
```

- Usb_unicode() macro function should be used everywhere (String descriptors...) an unicode char is exchanged on the USB protocol.

## Atmel Corporation

2325 Orchard Parkway
San Jose, CA 95131, USA
Tel: 1(408) 441-0311
Fax: 1(408) 487-2600

## Regional Headquarters

*Europe*
Atmel Sarl
Route des Arsenaux 41
Case Postale 80
CH-1705 Fribourg
Switzerland
Tel: (41) 26-426-5555
Fax: (41) 26-426-5500

*Asia*
Room 1219
Chinachem Golden Plaza
77 Mody Road Tsimshatsui
East Kowloon
Hong Kong
Tel: (852) 2721-9778
Fax: (852) 2722-1369

*Japan*
9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
Japan
Tel: (81) 3-3523-3551
Fax: (81) 3-3523-7581

## Atmel Operations

*Memory*
2325 Orchard Parkway
San Jose, CA 95131, USA
Tel: 1(408) 441-0311
Fax: 1(408) 436-4314

*Microcontrollers*
2325 Orchard Parkway
San Jose, CA 95131, USA
Tel: 1(408) 441-0311
Fax: 1(408) 436-4314

La Chantrerie
BP 70602
44306 Nantes Cedex 3, France
Tel: (33) 2-40-18-18-18
Fax: (33) 2-40-18-19-60

*ASIC/ASSP/Smart Cards*
Zone Industrielle
13106 Rousset Cedex, France
Tel: (33) 4-42-53-60-00
Fax: (33) 4-42-53-60-01

1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906, USA
Tel: 1(719) 576-3300
Fax: 1(719) 540-1759

Scottish Enterprise Technology Park
Maxwell Building
East Kilbride G75 0QR, Scotland
Tel: (44) 1355-803-000
Fax: (44) 1355-242-743

*RF/Automotive*
Theresienstrasse 2
Postfach 3535
74025 Heilbronn, Germany
Tel: (49) 71-31-67-0
Fax: (49) 71-31-67-2340

1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906, USA
Tel: 1(719) 576-3300
Fax: 1(719) 540-1759

*Biometrics/Imaging/Hi-Rel MPU/*
*High Speed Converters/RF Datacom*
Avenue de Rochepleine
BP 123
38521 Saint-Egreve Cedex, France
Tel: (33) 4-76-58-30-00
Fax: (33) 4-76-58-34-80

*Literature Requests*
www.atmel.com/literature