Atmel Microcontrollers

AT09340: USB Device Interface (UDI) for Vendor Class Device

APPLICATION NOTE

Introduction

USB Device Interface (UDI) for Vendor Class Device provides an interface for the configuration and management of USB Vendor Device.

The outline of this documentation is as follows:

- API Overview
- Quick Start Guide for USB Device Vendor Module (UDI Vendor)
- Configuration File Examples

For more details for Atmel® Software Framework (ASF) USB Device Stack, and USB Device Vendor, refer to following application notes:

- AVR4900: ASF - USB Device Stack
- AVR4901: ASF - USB Device Vendor Class Application.
- AVR4920: ASF - USB Device Stack - Compliance and Performance Figures
- AVR4921: ASF - USB Device Stack Differences between ASF V1 and V2

Table of Contents

1. Software License

2. API Overview

2.1. Variable and Type Definitions

2.1.1. Interface with USB Device Core (UDC)

Variable required by UDC.

2.1.1.1. Variable udi_api_vendor

```
UDC_DESC_STORAGE udi_api_t udi_api_vendor
```

Global structure which contains standard UDI interface for UDC.

2.2. Structure Definitions

2.2.1. Struct udi_vendor_desc_t

Interface descriptor structure for vendor Class interface.

Table 2-1. Members

Type	Name	Description
usb_iface_desc_t	iface0	Standard USB interface descriptor structure
usb_iface_desc_t	iface1	Standard USB interface descriptor structure

2.3. Macro Definitions

2.3.1. USB Interface Descriptors

The following structures provide predefined USB interface descriptors. It must be used to define the final USB descriptors.

2.3.1.1. Macro UDI_VENDOR_EPS_INT_DESC

```
#define UDI_VENDOR_EPS_INT_DESC
```

Endpoint descriptors. Support and FAQ: visit Atmel Support

2.3.1.2. Macro UDI_VENDOR_EPS_INT_DESC_FS

```
#define UDI_VENDOR_EPS_INT_DESC_FS
```

2.3.1.3. Macro UDI_VENDOR_EPS_INT_DESC_HS

```
#define UDI_VENDOR_EPS_INT_DESC_HS
```

2.3.1.4. Macro UDI_VENDOR_EPS_BULK_DESC

```
#define UDI_VENDOR_EPS_BULK_DESC
```

2.3.1.5. Macro UDI_VENDOR_EPS_BULK_DESC_FS

```
#define UDI_VENDOR_EPS_BULK_DESC_FS
```

2.3.1.6. Macro UDI_VENDOR_EPS_BULK_DESC_HS

```
#define UDI_VENDOR_EPS_BULK_DESC_HS
```

2.3.1.7. Macro UDI_VENDOR_EPS_ISO_DESC

```
#define UDI_VENDOR_EPS_ISO_DESC
```

2.3.1.8. Macro UDI_VENDOR_EPS_ISO_DESC_FS

```
#define UDI_VENDOR_EPS_ISO_DESC_FS
```

2.3.1.9. Macro UDI_VENDOR_EPS_ISO_DESC_HS

```
#define UDI_VENDOR_EPS_ISO_DESC_HS
```

2.3.1.10. Macro UDI_VENDOR_STRING_ID

```
#define UDI_VENDOR_STRING_ID
```

By default no string is associated to this interface.

2.3.1.11. Macro UDI_VENDOR_EP_NB_INT

```
#define UDI_VENDOR_EP_NB_INT
```

Maximum six endpoints used by vendor interface.

2.3.1.12. Macro UDI_VENDOR_EP_NB_BULK

```
#define UDI_VENDOR_EP_NB_BULK
```

2.3.1.13. Macro UDI_VENDOR_EP_NB_ISO

```
#define UDI_VENDOR_EP_NB_ISO
```

2.3.1.14. Macro UDI_VENDOR_EP_NB

```
#define UDI_VENDOR_EP_NB
```

2.3.1.15. Macro UDI_VENDOR_DESC

```
#define UDI_VENDOR_DESC
```

Content of vendor interface descriptor for all speeds.

2.3.1.16. Macro UDI_VENDOR_DESC_FS

```
#define UDI_VENDOR_DESC_FS
```

Content of vendor interface descriptor for full speed only.

2.3.1.17. Macro UDI_VENDOR_DESC_HS

```
#define UDI_VENDOR_DESC_HS
```

Content of vendor interface descriptor for high speed only.

2.4. Function Definitions

2.4.1. USB Device Interface (UDI) for Vendor Class

Common APIs used by high level application to use this USB class.

These routines are used to transfer data to/from USB VENDOR endpoints.

See Quick start guide for USB Device vendor module.

2.4.1.1. Function udi_vendor_interrupt_in_run()

Start a transfer on interrupt IN.

```
bool udi_vendor_interrupt_in_run(
        uint8_t * buf,
        iram_size_t buf_size,
        udd_callback_trans_t callback)
```

When the transfer is finished or aborted (stall, reset, ...), the *callback* is called. The *callback* returns the transfer status and eventually the number of byte transferred.

Table 2-2. Parameters

Data direction	Parameter name	Description
[in]	buf	Buffer on Internal RAM to send or fill. It must be align, then use COMPILER_WORD_ALIGNED.
[in]	buf_size	Buffer size to send or fill
[in]	callback	NULL or function to call at the end of transfer

Returns
`1` if function was successfully done, otherwise `0`.

2.4.1.2. Function udi_vendor_interrupt_out_run()

Start a transfer on interrupt OUT.

```
bool udi_vendor_interrupt_out_run(
        uint8_t * buf,
        iram_size_t buf_size,
        udd_callback_trans_t callback)
```

When the transfer is finished or aborted (stall, reset, ...), the *callback* is called. The *callback* returns the transfer status and eventually the number of byte transferred.

Table 2-3. Parameters

Data direction	Parameter name	Description
[in]	buf	Buffer on Internal RAM to send or fill. It must be align, then use COMPILER_WORD_ALIGNED.
[in]	buf_size	Buffer size to send or fill
[in]	callback	NULL or function to call at the end of transfer

Returns
`1` if function was successfully done, otherwise `0`.

2.4.1.3. Function udi_vendor_bulk_in_run()

Start a transfer on bulk IN.

```
bool udi_vendor_bulk_in_run(
        uint8_t * buf,
        iram_size_t buf_size,
        udd_callback_trans_t callback)
```

When the transfer is finished or aborted (stall, reset, ...), the *callback* is called. The *callback* returns the transfer status and eventually the number of byte transferred.

Table 2-4. Parameters

Data direction	Parameter name	Description
[in]	buf	Buffer on Internal RAM to send or fill. It must be align, then use COMPILER_WORD_ALIGNED.
[in]	buf_size	Buffer size to send or fill
[in]	callback	NULL or function to call at the end of transfer

Returns
`1` if function was successfully done, otherwise `0`.

2.4.1.4. Function udi_vendor_bulk_out_run()

Start a transfer on bulk OUT.

```
bool udi_vendor_bulk_out_run(
        uint8_t * buf,
        iram_size_t buf_size,
        udd_callback_trans_t callback)
```

When the transfer is finished or aborted (stall, reset, ...), the *callback* is called. The *callback* returns the transfer status and eventually the number of byte transferred.

Table 2-5. Parameters

Data direction	Parameter name	Description
[in]	buf	Buffer on Internal RAM to send or fill. It must be align, then use COMPILER_WORD_ALIGNED.
[in]	buf_size	Buffer size to send or fill
[in]	callback	NULL or function to call at the end of transfer

Returns
`1` if function was successfully done, otherwise `0`.

2.4.1.5. Function udi_vendor_iso_in_run()

Start a transfer on isochronous IN.

```
bool udi_vendor_iso_in_run(
        uint8_t * buf,
        iram_size_t buf_size,
        udd_callback_trans_t callback)
```

When the transfer is finished or aborted (stall, reset, ...), the *callback* is called. The *callback* returns the transfer status and eventually the number of byte transferred.

Table 2-6. Parameters

Data direction	Parameter name	Description
[in]	buf	Buffer on Internal RAM to send or fill. It must be align, then use COMPILER_WORD_ALIGNED.
[in]	buf_size	Buffer size to send or fill
[in]	callback	NULL or function to call at the end of transfer

Returns
`1` if function was successfully done, otherwise `0`.

2.4.1.6. Function udi_vendor_iso_out_run()

Start a transfer on isochronous OUT.

```
bool udi_vendor_iso_out_run(
        uint8_t * buf,
        iram_size_t buf_size,
        udd_callback_trans_t callback)
```

When the transfer is finished or aborted (stall, reset, ...), the *callback* is called. The *callback* returns the transfer status and eventually the number of byte transferred.

Table 2-7. Parameters

Data direction	Parameter name	Description
[in]	buf	Buffer on Internal RAM to send or fill. It must be align, then use COMPILER_WORD_ALIGNED.
[in]	buf_size	Buffer size to send or fill
[in]	callback	NULL or function to call at the end of transfer

Returns

`1` if function was successfully done, otherwise `0`.

3. Quick Start Guide for USB Device Vendor Module (UDI Vendor)

This is the quick start guide for the USB Device Vendor Module (UDI Vendor) with step-by-step instructions on how to configure and use the modules in a selection of use cases.

The use cases highlights several code fragments. The code fragments in the steps for setup can be copied into a custom initialization function, while the steps for usage can be copied into, e.g., the main application function.

3.1. Basic Use Case

In this basic use case, the "USB Vendor (Single Class support)" module is used. The "USB Vendor (Composite Device)" module usage is described in Advanced Use Cases.

3.1.1. Setup Steps

As a USB device, it follows common USB device setup steps. Refer to USB Device Basic Setup.

3.1.2. Usage Steps

3.1.2.1. Example code

Content of conf_usb.h:

```
* #define UDI_VENDOR_ENABLE_EXT() my_callback_vendor_enable()
* extern bool my_callback_vendor_enable(void);
* #define UDI_VENDOR_DISABLE_EXT() my_callback_vendor_disable()
* extern void my_callback_vendor_disable(void);
*
* #define  UDI_VENDOR_SETUP_OUT_RECEIVED()  my_vendor_setup_out_received()
* extern bool my_vendor_setup_out_received(void);
* #define  UDI_VENDOR_SETUP_IN_RECEIVED()   my_vendor_setup_in_received()
* extern bool my_vendor_setup_in_received(void);
*
* #define  UDI_VENDOR_EPS_SIZE_INT_FS    64
* #define  UDI_VENDOR_EPS_SIZE_BULK_FS   64
* #define  UDI_VENDOR_EPS_SIZE_ISO_FS   256
* #define  UDI_VENDOR_EPS_SIZE_INT_HS    64
* #define  UDI_VENDOR_EPS_SIZE_BULK_HS  512
* #define  UDI_VENDOR_EPS_SIZE_ISO_HS    64

* #include "udi_vendor_conf.h" // At the end of conf_usb.h file
```

Add to application C-file:

```
static bool my_flag_autorize_vendor_transfert = false;
bool my_callback_vendor_enable(void)
{
   my_flag_autorize_vendor_transfert = true;
   return true;
}
void my_callback_vendor_disable(void)
{
   my_flag_autorize_vendor_transfert = false;
}

uint8_t global_buffer[X];
void task(void)
```

```
{
   if (my_flag_autorize_vendor_transfert) {
     // Enable a transfer on OUT interrupt endpoint
     udi_vendor_interrupt_out_run(
            global_buffer,
            sizeof(global_buffer),
            NULL);
     // Enable a transfer on IN interrupt endpoint
     udi_vendor_interrupt_in_run(
            global_buffer,
            sizeof(global_buffer),
            NULL);
     ...
   }
}
```

3.1.3. Workflow

1. Ensure that conf_usb.h is available and contains the following configuration, which is the USB device Vendor configuration:

```
#define UDI_VENDOR_ENABLE_EXT() my_callback_vendor_enable()
extern bool my_callback_vendor_enable(void);
```

Note: After the device enumeration (detecting and identifying USB devices), the USB host starts the device configuration. When the USB Vendor interface from the device is accepted by the host, the USB host enables this interface and the UDI_VENDOR_ENABLE_EXT() callback function is called and return true. Thus, when this event is received, the Vendor transfers can start.

```
#define UDI_VENDOR_DISABLE_EXT() my_callback_vendor_disable()
extern void my_callback_vendor_disable(void);
```

Note: When the USB device is unplugged or is reset by the USB host, the USB interface is disabled and the UDI_VENDOR_DISABLE_EXT() callback function is called. Thus, it is recommended to disable the data Vendor transfer.

```
#define  UDI_VENDOR_SETUP_OUT_RECEIVED()
my_vendor_setup_out_received()
extern bool my_vendor_setup_out_received(void);
#define  UDI_VENDOR_SETUP_IN_RECEIVED()   my_vendor_setup_in_received()
extern bool my_vendor_setup_in_received(void);
```

Note: The control requests for the interface Vendor will be processed through these both callbacks.

```
#define  UDI_VENDOR_EPS_SIZE_INT_FS    64
#define  UDI_VENDOR_EPS_SIZE_BULK_FS   64
#define  UDI_VENDOR_EPS_SIZE_ISO_FS   256
#define  UDI_VENDOR_EPS_SIZE_INT_HS    64
#define  UDI_VENDOR_EPS_SIZE_BULK_HS  512
#define  UDI_VENDOR_EPS_SIZE_ISO_HS    64
```

Note: The endpoint size is defined by the final application, and can be disabled if the full speed size is zero.

2. The Vendor transfers on interrupt, bulk, and isochronous endpoints are done through these functions:

```
// Start a transfer on interrupt IN
udi_vendor_interrupt_in_run();
// Start a transfer on interrupt OUT
udi_vendor_interrupt_out_run();
// Start a transfer on bulk IN
udi_vendor_bulk_in_run();
// Start a transfer on bulk OUT
udi_vendor_bulk_out_run();
// Start a transfer on isochronous IN
udi_vendor_iso_in_run();
// Start a transfer on isochronous OUT
udi_vendor_iso_out_run();
```

3.2. Advanced Use Cases

For more advanced use of the udi vendor module, see the following use cases:

- Vendor in a Composite Device
- Change USB Speed
- Use USB Strings
- Use USB Remote Wakeup Feature
- Bus Power Application Recommendations
- USB Dynamic Serial Number

3.3. Vendor in a Composite Device

A USB Composite Device is a USB Device, which uses more than one USB class. In this use case, the "USB Vendor (Composite Device)" module is used to create a USB composite device. Thus, this USB module can be associated with another "Composite Device" module, like "USB HID Mouse (Composite Device)".

Also, you can refer to application note AVR4902 ASF - USB Composite Device.

3.3.1. Setup Steps

For the setup code of this use case to work, the Basic Use Case must be followed.

3.3.2. Usage Steps

3.3.2.1. Example Code

Content of conf_usb.h:

```
#define USB_DEVICE_EP_CTRL_SIZE  64
#define USB_DEVICE_NB_INTERFACE (X+1)
#define USB_DEVICE_MAX_EP (X) to (X+6)

#define  UDI_VENDOR_EP_INTERRUPT_IN  (1 | USB_EP_DIR_IN)
#define  UDI_VENDOR_EP_INTERRUPT_OUT (2 | USB_EP_DIR_OUT)
#define  UDI_VENDOR_EP_BULK_IN       (3 | USB_EP_DIR_IN)
#define  UDI_VENDOR_EP_BULK_OUT      (4 | USB_EP_DIR_OUT)
#define  UDI_VENDOR_EP_ISO_IN        (5 | USB_EP_DIR_IN)
#define  UDI_VENDOR_EP_ISO_OUT       (6 | USB_EP_DIR_OUT)
```

```
#define UDI_VENDOR_IFACE_NUMBER  X

#define UDI_COMPOSITE_DESC_T \
   udi_vendor_desc_t udi_vendor; \
   ...
#define UDI_COMPOSITE_DESC_FS \
   .udi_vendor = UDI_VENDOR_DESC, \
   ...
#define UDI_COMPOSITE_DESC_HS \
   .udi_vendor = UDI_VENDOR_DESC, \
   ...
#define UDI_COMPOSITE_API \
   &udi_api_vendor, \
   ...
```

3.3.2.2. Workflow

1. Ensure that conf_usb.h is available and contains the following parameters required for a USB composite device configuration:

```
// Endpoint control size, This must be:
// - 8, 16, 32 or 64 for full speed device (8 is recommended to save RAM)
// - 64 for a high speed device
#define USB_DEVICE_EP_CTRL_SIZE  64
// Total Number of interfaces on this USB device.
// Add 1 for Vendor.
#define USB_DEVICE_NB_INTERFACE (X+1)
// Total number of endpoints on this USB device.
// This must include each endpoint for each interface.
// Add 0 to 6 for Vendor interface.
// The number depends on UDI_VENDOR_EPS_SIZE_..._FS defines.
#define USB_DEVICE_MAX_EP (X) to (X+6)
```

2. Ensure that conf_usb.h contains the description of composite device:

```
// The endpoint numbers chosen by you for the Vendor.
// The endpoint numbers starting from 1.
#define  UDI_VENDOR_EP_INTERRUPT_IN  (1 | USB_EP_DIR_IN)
#define  UDI_VENDOR_EP_INTERRUPT_OUT (2 | USB_EP_DIR_OUT)
#define  UDI_VENDOR_EP_BULK_IN       (3 | USB_EP_DIR_IN)
#define  UDI_VENDOR_EP_BULK_OUT      (4 | USB_EP_DIR_OUT)
#define  UDI_VENDOR_EP_ISO_IN        (5 | USB_EP_DIR_IN)
#define  UDI_VENDOR_EP_ISO_OUT       (6 | USB_EP_DIR_OUT)
// The interface index of an interface starting from 0
#define UDI_VENDOR_IFACE_NUMBER  X
```

3. Ensure that conf_usb.h contains the following parameters required for a USB composite device configuration:

```
// USB Interfaces descriptor structure
#define UDI_COMPOSITE_DESC_T \
   ...
   udi_vendor_desc_t udi_vendor; \
   ...
// USB Interfaces descriptor value for Full Speed
#define UDI_COMPOSITE_DESC_FS \
   ...
   .udi_vendor = UDI_VENDOR_DESC_FS, \
   ...
// USB Interfaces descriptor value for High Speed
#define UDI_COMPOSITE_DESC_HS \
   ...
```

```
    .udi_vendor = UDI_VENDOR_DESC_HS, \
    ...
// USB Interface APIs
#define UDI_COMPOSITE_API \
    ...
    &udi_api_vendor, \
    ...
```

Note: The descriptors order given in the four lists above must be the same as the order defined by all interface indexes. The interface index orders are defined through UDI_X_IFACE_NUMBER defines.

3.4. Change USB Speed

In this use case, the USB device is used with different USB speeds.

3.4.1. Setup Steps

Prior to implement this use case, be sure to have already applied the UDI module "basic use case".

3.4.2. Usage Steps

3.4.2.1. Example Code

Content of conf_usb.h:

```
#if // Low speed
#define USB_DEVICE_LOW_SPEED
// #define USB_DEVICE_HS_SUPPORT

#elif // Full speed
// #define USB_DEVICE_LOW_SPEED
// #define USB_DEVICE_HS_SUPPORT
#elif // High speed
// #define USB_DEVICE_LOW_SPEED
#define USB_DEVICE_HS_SUPPORT

#endif
```

3.4.2.2. Workflow

1. Ensure that conf_usb.h is available and contains the following parameters required for a USB device low speed (1.5Mbit/s):

   ```
   #define USB_DEVICE_LOW_SPEED
   //#define  USB_DEVICE_HS_SUPPORT
   ```

2. Ensure that conf_usb.h contains the following parameters required for a USB device full speed (12Mbit/s):

   ```
   //#define USB_DEVICE_LOW_SPEED
   //#define  USB_DEVICE_HS_SUPPORT
   ```

3. Ensure that conf_usb.h contains the following parameters required for a USB device high speed (480Mbit/s):

   ```
   //#define USB_DEVICE_LOW_SPEED
   #define  USB_DEVICE_HS_SUPPORT
   ```

3.5. Use USB Strings

In this use case, the usual USB strings are added in the USB device.

3.5.1. Setup Steps

Prior to implement this use case, be sure to have already applied the UDI module "basic use case".

3.5.2. Usage Steps

3.5.2.1. Example Code

Content of conf_usb.h:

```
#define  USB_DEVICE_MANUFACTURE_NAME      "Manufacture name"
#define  USB_DEVICE_PRODUCT_NAME          "Product name"
#define  USB_DEVICE_SERIAL_NAME           "12...EF"
```

3.5.2.2. Workflow

1. Ensure that conf_usb.h is available and contains the following parameters required to enable different USB strings:

```
// Static ASCII name for the manufacture
#define  USB_DEVICE_MANUFACTURE_NAME "Manufacture name"
```

```
// Static ASCII name for the product
#define  USB_DEVICE_PRODUCT_NAME "Product name"
```

```
// Static ASCII name to enable and set a serial number
#define  USB_DEVICE_SERIAL_NAME "12...EF"
```

3.6. Use USB Remote Wakeup Feature

In this use case, the USB remote wakeup feature is enabled.

3.6.1. Setup Steps

Prior to implement this use case, be sure to have already applied the UDI module "basic use case".

3.6.2. Usage Steps

3.6.2.1. Example Code

Content of conf_usb.h:

```
#define  USB_DEVICE_ATTR \
(USB_CONFIG_ATTR_REMOTE_WAKEUP | USB_CONFIG_ATTR_..._POWERED)
#define UDC_REMOTEWAKEUP_ENABLE() my_callback_remotewakeup_enable()
extern void my_callback_remotewakeup_enable(void);
#define UDC_REMOTEWAKEUP_DISABLE() my_callback_remotewakeup_disable()
extern void my_callback_remotewakeup_disable(void);
```

Add to application C-file:

```
void my_callback_remotewakeup_enable(void)
{
```

```
// Enable application wakeup events (e.g. enable GPIO interrupt)
}
void my_callback_remotewakeup_disable(void)
{
// Disable application wakeup events (e.g. disable GPIO interrupt)
}

void my_interrupt_event(void)
{
   udc_remotewakeup();
}
```

3.6.2.2. Workflow

1. Ensure that conf_usb.h is available and contains the following parameters required to enable the remote wakeup feature:

```
// Authorizes the remote wakeup feature
#define  USB_DEVICE_ATTR (USB_CONFIG_ATTR_REMOTE_WAKEUP |
USB_CONFIG_ATTR_..._POWERED)
```

```
// Define callback called when the host enables the remotewakeup feature
#define UDC_REMOTEWAKEUP_ENABLE() my_callback_remotewakeup_enable()
extern void my_callback_remotewakeup_enable(void);
```

```
// Define callback called when the host disables the remotewakeup feature
#define UDC_REMOTEWAKEUP_DISABLE() my_callback_remotewakeup_disable()
extern void my_callback_remotewakeup_disable(void);
```

2. Send a remote wakeup (USB upstream):

```
udc_remotewakeup();
```

3.7. Bus Power Application Recommendations

In this use case, the USB device bus power feature is enabled. This feature requires a correct power consumption management.

3.7.1. Setup Steps

Prior to implement this use case, be sure to have already applied the UDI module "basic use case".

3.7.2. Usage Steps

3.7.2.1. Example Code

Content of conf_usb.h:

```
#define  USB_DEVICE_ATTR (USB_CONFIG_ATTR_BUS_POWERED)
#define  UDC_SUSPEND_EVENT()         user_callback_suspend_action()
extern void user_callback_suspend_action(void)
#define  UDC_RESUME_EVENT()          user_callback_resume_action()
extern void user_callback_resume_action(void)
```

Add to application C-file:

```
void user_callback_suspend_action(void)
{
   // Disable hardware component to reduce power consumption
```

```
}
void user_callback_resume_action(void)
{
   // Re-enable hardware component
}
```

3.7.2.2. Workflow

1. Ensure that conf_usb.h is available and contains the following parameters:

```
// Authorizes the BUS power feature
#define  USB_DEVICE_ATTR (USB_CONFIG_ATTR_BUS_POWERED)
```

```
// Define callback called when the host suspend the USB line
#define UDC_SUSPEND_EVENT() user_callback_suspend_action()
extern void user_callback_suspend_action(void);
```

```
// Define callback called when the host or device resume the USB line
#define UDC_RESUME_EVENT() user_callback_resume_action()
extern void user_callback_resume_action(void);
```

2. Reduce power consumption in suspend mode (max. 2.5mA on VBUS):

```
void user_callback_suspend_action(void)
{
   turn_off_components();
}
```

3.8. USB Dynamic Serial Number

In this use case, the USB serial strings are dynamic. For a static serial string refer to Use USB Strings.

3.8.1. Setup Steps

Prior to implement this use case, be sure to have already applied the UDI module "basic use case".

3.8.2. Usage Steps

3.8.2.1. Example Code

Content of conf_usb.h:

```
#define  USB_DEVICE_SERIAL_NAME
#define  USB_DEVICE_GET_SERIAL_NAME_POINTER serial_number
#define  USB_DEVICE_GET_SERIAL_NAME_LENGTH  12
extern uint8_t serial_number[];
```

Add to application C-file:

```
uint8_t serial_number[USB_DEVICE_GET_SERIAL_NAME_LENGTH];
void init_build_usb_serial_number(void)
{
serial_number[0] = 'A';
serial_number[1] = 'B';
...
serial_number[USB_DEVICE_GET_SERIAL_NAME_LENGTH-1] = 'C';
}
```

3.8.2.2. Workflow

1. Ensure that conf_usb.h is available and contains the following parameters required to enable a USB serial number string dynamically:

```
#define  USB_DEVICE_SERIAL_NAME // Define this empty
#define  USB_DEVICE_GET_SERIAL_NAME_POINTER serial_number // Give
serial array pointer
#define  USB_DEVICE_GET_SERIAL_NAME_LENGTH  12 // Give size of serial
array
extern uint8_t serial_number[]; // Declare external serial array
```

2. Before starting USB stack, initialize the serial array:

```
uint8_t serial_number[USB_DEVICE_GET_SERIAL_NAME_LENGTH];
void init_build_usb_serial_number(void)
{
serial_number[0] = 'A';
serial_number[1] = 'B';
...
serial_number[USB_DEVICE_GET_SERIAL_NAME_LENGTH-1] = 'C';
}
```

4. Configuration File Examples

4.1. conf_usb.h

4.1.1. UDI Vendor Single

```
/*
 * Support and FAQ: visit <a href="http://www.atmel.com/design-support/">Atmel
Support</a>
 */

#ifndef _CONF_USB_H_
#define _CONF_USB_H_

#include "compiler.h"

#warning You must refill the following definitions with a correct values

#define  USB_DEVICE_VENDOR_ID             USB_VID_ATMEL
#define  USB_DEVICE_PRODUCT_ID            USB_PID_ATMEL_ASF_VENDOR_CLASS
#define  USB_DEVICE_MAJOR_VERSION         1
#define  USB_DEVICE_MINOR_VERSION         0
#define  USB_DEVICE_POWER                 100 // Consumption on VBUS line
(mA)
#define  USB_DEVICE_ATTR                  \
    (USB_CONFIG_ATTR_SELF_POWERED)
// (USB_CONFIG_ATTR_BUS_POWERED)
//  (USB_CONFIG_ATTR_REMOTE_WAKEUP|USB_CONFIG_ATTR_SELF_POWERED)
//  (USB_CONFIG_ATTR_REMOTE_WAKEUP|USB_CONFIG_ATTR_BUS_POWERED)

// #define  USB_DEVICE_MANUFACTURE_NAME      "Manufacture name"
// #define  USB_DEVICE_PRODUCT_NAME          "Product name"
// #define  USB_DEVICE_SERIAL_NAME           "12...EF"

#if (UC3A3||UC3A4)
# define  USB_DEVICE_HS_SUPPORT
#endif



// #define  UDC_VBUS_EVENT(b_vbus_high)      user_callback_vbus_action(b_vbus_high)
// extern void user_callback_vbus_action(bool b_vbus_high);
// #define  UDC_SOF_EVENT()                  user_callback_sof_action()
// extern void user_callback_sof_action(void);
// #define  UDC_SUSPEND_EVENT()              user_callback_suspend_action()
// extern void user_callback_suspend_action(void);
// #define  UDC_RESUME_EVENT()               user_callback_resume_action()
// extern void user_callback_resume_action(void);
// #define  UDC_REMOTEWAKEUP_ENABLE()        user_callback_remotewakeup_enable()
// extern void user_callback_remotewakeup_enable(void);
// #define  UDC_REMOTEWAKEUP_DISABLE()       user_callback_remotewakeup_disable()
// extern void user_callback_remotewakeup_disable(void);
// #define  UDC_GET_EXTRA_STRING()



#define UDI_VENDOR_ENABLE_EXT()           true
#define UDI_VENDOR_DISABLE_EXT()
```

```
#define UDI_VENDOR_SETUP_OUT_RECEIVED()  false
#define UDI_VENDOR_SETUP_IN_RECEIVED()   false
/*
 * #define UDI_VENDOR_ENABLE_EXT() my_callback_vendor_enable()
 * extern bool my_callback_vendor_enable(void);
 * #define UDI_VENDOR_DISABLE_EXT() my_callback_vendor_disable()
 * extern void my_callback_vendor_disable(void);
 *
 * #define  UDI_VENDOR_SETUP_OUT_RECEIVED()  my_vendor_setup_out_received()
 * extern bool my_vendor_setup_out_received(void);
 * #define  UDI_VENDOR_SETUP_IN_RECEIVED()   my_vendor_setup_in_received()
 * extern bool my_vendor_setup_in_received(void);
 */

#define UDI_VENDOR_EPS_SIZE_INT_FS    64
#define UDI_VENDOR_EPS_SIZE_BULK_FS   64
#if SAMG55
#define UDI_VENDOR_EPS_SIZE_ISO_FS   0
#else
#define UDI_VENDOR_EPS_SIZE_ISO_FS   256
#endif

#define UDI_VENDOR_EPS_SIZE_INT_HS    64
#define UDI_VENDOR_EPS_SIZE_BULK_HS  512
#define UDI_VENDOR_EPS_SIZE_ISO_HS    64




#include "udi_vendor_conf.h"

#endif // _CONF_USB_H_
```

4.1.2. UDI Vendor Multiple (Composite)

```
/*
 * Support and FAQ: visit <a href="http://www.atmel.com/design-support/">Atmel
Support</a>
 */

#ifndef _CONF_USB_H_
#define _CONF_USB_H_

#include "compiler.h"

#warning You must refill the following definitions with a correct values

#define  USB_DEVICE_VENDOR_ID             USB_VID_ATMEL
#define  USB_DEVICE_PRODUCT_ID            0xFFFF
#define  USB_DEVICE_MAJOR_VERSION         1
#define  USB_DEVICE_MINOR_VERSION         0
#define  USB_DEVICE_POWER                 100 // Consumption on VBUS line
(mA)
#define  USB_DEVICE_ATTR                  \
        (USB_CONFIG_ATTR_SELF_POWERED)
// (USB_CONFIG_ATTR_BUS_POWERED)
// (USB_CONFIG_ATTR_REMOTE_WAKEUP|USB_CONFIG_ATTR_SELF_POWERED)
// (USB_CONFIG_ATTR_REMOTE_WAKEUP|USB_CONFIG_ATTR_BUS_POWERED)

// #define  USB_DEVICE_MANUFACTURE_NAME      "Manufacture name"
// #define  USB_DEVICE_PRODUCT_NAME          "Product name"
// #define  USB_DEVICE_SERIAL_NAME           "12...EF" // Disk SN for MSC
```

```
//#define  USB_DEVICE_LOW_SPEED

#if (UC3A3||UC3A4)
//#define  USB_DEVICE_HS_SUPPORT
#endif


// #define  UDC_VBUS_EVENT(b_vbus_high)      user_callback_vbus_action(b_vbus_high)
// extern void user_callback_vbus_action(bool b_vbus_high);
// #define  UDC_SOF_EVENT()                  user_callback_sof_action()
// extern void user_callback_sof_action(void);
// #define  UDC_SUSPEND_EVENT()              user_callback_suspend_action()
// extern void user_callback_suspend_action(void);
// #define  UDC_RESUME_EVENT()               user_callback_resume_action()
// extern void user_callback_resume_action(void);
// #define  UDC_REMOTEWAKEUP_ENABLE()        user_callback_remotewakeup_enable()
// extern void user_callback_remotewakeup_enable(void);
// #define  UDC_REMOTEWAKEUP_DISABLE()       user_callback_remotewakeup_disable()
// extern void user_callback_remotewakeup_disable(void);
// #define  UDC_GET_EXTRA_STRING()

#define  USB_DEVICE_EP_CTRL_SIZE       64

#define  USB_DEVICE_NB_INTERFACE       1 // 1 or more

#define  USB_DEVICE_MAX_EP             1 // 0 to max endpoint requested by
interfaces



#define  UDI_CDC_PORT_NB 1

#define  UDI_CDC_ENABLE_EXT(port)              true
#define  UDI_CDC_DISABLE_EXT(port)
#define  UDI_CDC_RX_NOTIFY(port)
#define  UDI_CDC_TX_EMPTY_NOTIFY(port)
#define  UDI_CDC_SET_CODING_EXT(port,cfg)
#define  UDI_CDC_SET_DTR_EXT(port,set)
#define  UDI_CDC_SET_RTS_EXT(port,set)
/*
 * #define UDI_CDC_ENABLE_EXT(port) my_callback_cdc_enable()
 * extern bool my_callback_cdc_enable(void);
 * #define UDI_CDC_DISABLE_EXT(port) my_callback_cdc_disable()
 * extern void my_callback_cdc_disable(void);
 * #define  UDI_CDC_RX_NOTIFY(port) my_callback_rx_notify(port)
 * extern void my_callback_rx_notify(uint8_t port);
 * #define  UDI_CDC_TX_EMPTY_NOTIFY(port) my_callback_tx_empty_notify(port)
 * extern void my_callback_tx_empty_notify(uint8_t port);
 * #define  UDI_CDC_SET_CODING_EXT(port,cfg) my_callback_config(port,cfg)
 * extern void my_callback_config(uint8_t port, usb_cdc_line_coding_t * cfg);
 * #define  UDI_CDC_SET_DTR_EXT(port,set) my_callback_cdc_set_dtr(port,set)
 * extern void my_callback_cdc_set_dtr(uint8_t port, bool b_enable);
 * #define  UDI_CDC_SET_RTS_EXT(port,set) my_callback_cdc_set_rts(port,set)
 * extern void my_callback_cdc_set_rts(uint8_t port, bool b_enable);
 */

#define  UDI_CDC_LOW_RATE

#define  UDI_CDC_DEFAULT_RATE             115200
#define  UDI_CDC_DEFAULT_STOPBITS         CDC_STOP_BITS_1
#define  UDI_CDC_DEFAULT_PARITY           CDC_PAR_NONE
#define  UDI_CDC_DEFAULT_DATABITS         8
```

```
#define  UDI_CDC_DATA_EP_IN_0          (1 | USB_EP_DIR_IN)  // TX
#define  UDI_CDC_DATA_EP_OUT_0         (2 | USB_EP_DIR_OUT) // RX
#define  UDI_CDC_COMM_EP_0             (3 | USB_EP_DIR_IN)  // Notify
endpoint
#define  UDI_CDC_DATA_EP_IN_2          (4 | USB_EP_DIR_IN)  // TX
#define  UDI_CDC_DATA_EP_OUT_2         (5 | USB_EP_DIR_OUT) // RX
#define  UDI_CDC_COMM_EP_2             (6 | USB_EP_DIR_IN)  // Notify
endpoint
#define  UDI_CDC_DATA_EP_IN_3          (7 | USB_EP_DIR_IN)  // TX
#define  UDI_CDC_DATA_EP_OUT_3         (8 | USB_EP_DIR_OUT) // RX
#define  UDI_CDC_COMM_EP_3             (9 | USB_EP_DIR_IN)  // Notify
endpoint

#define  UDI_CDC_COMM_IFACE_NUMBER_0   0
#define  UDI_CDC_DATA_IFACE_NUMBER_0   1
#define  UDI_CDC_COMM_IFACE_NUMBER_2   2
#define  UDI_CDC_DATA_IFACE_NUMBER_2   3
#define  UDI_CDC_COMM_IFACE_NUMBER_3   4
#define  UDI_CDC_DATA_IFACE_NUMBER_3   5



#define UDI_MSC_GLOBAL_VENDOR_ID            \
   'A', 'T', 'M', 'E', 'L', ' ', ' ', ' '
#define UDI_MSC_GLOBAL_PRODUCT_VERSION           \
   '1', '.', '0', '0'

#define  UDI_MSC_ENABLE_EXT()          true
#define  UDI_MSC_DISABLE_EXT()
#define  UDI_MSC_NOTIFY_TRANS_EXT()
/*
 * #define UDI_MSC_ENABLE_EXT() my_callback_msc_enable()
 * extern bool my_callback_msc_enable(void);
 * #define UDI_MSC_DISABLE_EXT() my_callback_msc_disable()
 * extern void my_callback_msc_disable(void);
 * #define  UDI_MSC_NOTIFY_TRANS_EXT()    msc_notify_trans()
 * extern void msc_notify_trans(void) {
 */

#define  UDI_MSC_EP_IN                 (1 | USB_EP_DIR_IN)
#define  UDI_MSC_EP_OUT                (2 | USB_EP_DIR_OUT)

#define  UDI_MSC_IFACE_NUMBER          0



#define  UDI_HID_MOUSE_ENABLE_EXT()        true
#define  UDI_HID_MOUSE_DISABLE_EXT()
// #define UDI_HID_MOUSE_ENABLE_EXT() my_callback_mouse_enable()
// extern bool my_callback_mouse_enable(void);
// #define UDI_HID_MOUSE_DISABLE_EXT() my_callback_mouse_disable()
// extern void my_callback_mouse_disable(void);

#define  UDI_HID_MOUSE_EP_IN           (1 | USB_EP_DIR_IN)

#define  UDI_HID_MOUSE_IFACE_NUMBER    0
```

```
#define  UDI_HID_KBD_ENABLE_EXT()       true
#define  UDI_HID_KBD_DISABLE_EXT()
// #define UDI_HID_KBD_ENABLE_EXT() my_callback_keyboard_enable()
// extern bool my_callback_keyboard_enable(void);
// #define UDI_HID_KBD_DISABLE_EXT() my_callback_keyboard_disable()
// extern void my_callback_keyboard_disable(void);
#define  UDI_HID_KBD_CHANGE_LED(value)
// #define  UDI_HID_KBD_CHANGE_LED(value) my_callback_keyboard_led(value)
// extern void my_callback_keyboard_led(uint8_t value)

#define  UDI_HID_KBD_EP_IN           (1 | USB_EP_DIR_IN)

#define  UDI_HID_KBD_IFACE_NUMBER    0



#define  UDI_HID_GENERIC_ENABLE_EXT()       true
#define  UDI_HID_GENERIC_DISABLE_EXT()
#define  UDI_HID_GENERIC_REPORT_OUT(ptr)
#define  UDI_HID_GENERIC_SET_FEATURE(f)
/*
 * #define UDI_HID_GENERIC_ENABLE_EXT() my_callback_generic_enable()
 * extern bool my_callback_generic_enable(void);
 * #define UDI_HID_GENERIC_DISABLE_EXT() my_callback_generic_disable()
 * extern void my_callback_generic_disable(void);
 * #define  UDI_HID_GENERIC_REPORT_OUT(ptr) my_callback_generic_report_out(ptr)
 * extern void my_callback_generic_report_out(uint8_t *report);
 * #define  UDI_HID_GENERIC_SET_FEATURE(f) my_callback_generic_set_feature(f)
 * extern void my_callback_generic_set_feature(uint8_t *report_feature);
 */
#define  UDI_HID_REPORT_IN_SIZE             64
#define  UDI_HID_REPORT_OUT_SIZE            64
#define  UDI_HID_REPORT_FEATURE_SIZE        4
#define  UDI_HID_GENERIC_EP_SIZE            64

#define  UDI_HID_GENERIC_EP_OUT   (2 | USB_EP_DIR_OUT)
#define  UDI_HID_GENERIC_EP_IN    (1 | USB_EP_DIR_IN)

#define  UDI_HID_GENERIC_IFACE_NUMBER    0



#define  UDI_PHDC_ENABLE_EXT()          true
#define  UDI_PHDC_DISABLE_EXT()

#define  UDI_PHDC_DATAMSG_FORMAT       USB_PHDC_DATAMSG_FORMAT_11073_20601
#define  UDI_PHDC_SPECIALIZATION       {0x2345} // Define in 11073_20601

#define  UDI_PHDC_QOS_OUT              \
        (USB_PHDC_QOS_MEDIUM_BETTER|USB_PHDC_QOS_HIGH_BEST)
#define  UDI_PHDC_QOS_IN               \
        (USB_PHDC_QOS_LOW_GOOD|USB_PHDC_QOS_MEDIUM_BETTER|
USB_PHDC_QOS_MEDIUM_BEST)

#define  UDI_PHDC_METADATA_DESC_BULK_IN   {0x01,0x02,0x03}
#define  UDI_PHDC_METADATA_DESC_BULK_OUT  {0x01,0x02,0x03}
#define  UDI_PHDC_METADATA_DESC_INT_IN    {0x01,0x02,0x03}

#define  UDI_PHDC_EP_BULK_OUT          (1 | USB_EP_DIR_OUT)
#define  UDI_PHDC_EP_BULK_IN           (2 | USB_EP_DIR_IN)
#if ((UDI_PHDC_QOS_IN&USB_PHDC_QOS_LOW_GOOD)==USB_PHDC_QOS_LOW_GOOD)
```

```
// Only if UDI_PHDC_QOS_IN include USB_PHDC_QOS_LOW_GOOD
#  define  UDI_PHDC_EP_INTERRUPT_IN      (3 | USB_EP_DIR_IN)
#endif

#define  UDI_PHDC_EP_SIZE_BULK_OUT     32
#define  UDI_PHDC_EP_SIZE_BULK_IN      32
#define  UDI_PHDC_EP_SIZE_INT_IN       8


#define  UDI_PHDC_IFACE_NUMBER          0



#define  UDI_VENDOR_ENABLE_EXT()          true
#define  UDI_VENDOR_DISABLE_EXT()
#define  UDI_VENDOR_SETUP_OUT_RECEIVED()  false
#define  UDI_VENDOR_SETUP_IN_RECEIVED()   false
/*
 * #define UDI_VENDOR_ENABLE_EXT() my_callback_vendor_enable()
 * extern bool my_callback_vendor_enable(void);
 * #define UDI_VENDOR_DISABLE_EXT() my_callback_vendor_disable()
 * extern void my_callback_vendor_disable(void);
 *
 * #define  UDI_VENDOR_SETUP_OUT_RECEIVED()  my_vendor_setup_out_received()
 * extern bool my_vendor_setup_out_received(void);
 * #define  UDI_VENDOR_SETUP_IN_RECEIVED()   my_vendor_setup_in_received()
 * extern bool my_vendor_setup_in_received(void);
 */

#define  UDI_VENDOR_EPS_SIZE_INT_FS    64
#define  UDI_VENDOR_EPS_SIZE_BULK_FS   64
#define  UDI_VENDOR_EPS_SIZE_ISO_FS   256

#define  UDI_VENDOR_EPS_SIZE_INT_HS    64
#define  UDI_VENDOR_EPS_SIZE_BULK_HS  512
#define  UDI_VENDOR_EPS_SIZE_ISO_HS    64

#define  UDI_VENDOR_EP_INTERRUPT_IN  (1 | USB_EP_DIR_IN)
#define  UDI_VENDOR_EP_INTERRUPT_OUT (2 | USB_EP_DIR_OUT)
#define  UDI_VENDOR_EP_BULK_IN       (3 | USB_EP_DIR_IN)
#define  UDI_VENDOR_EP_BULK_OUT      (4 | USB_EP_DIR_OUT)
#define  UDI_VENDOR_EP_ISO_IN        (5 | USB_EP_DIR_IN)
#define  UDI_VENDOR_EP_ISO_OUT       (6 | USB_EP_DIR_OUT)


#define  UDI_VENDOR_IFACE_NUMBER     0



//... Eventually add other Interface Configuration



#define UDI_COMPOSITE_DESC_T

#define UDI_COMPOSITE_DESC_FS

#define UDI_COMPOSITE_DESC_HS

#define UDI_COMPOSITE_API

/* Example for device with cdc, msc and hid mouse interface
#define UDI_COMPOSITE_DESC_T \
    usb_iad_desc_t udi_cdc_iad; \
```

```
    udi_cdc_comm_desc_t udi_cdc_comm; \
    udi_cdc_data_desc_t udi_cdc_data; \
    udi_msc_desc_t udi_msc; \
    udi_hid_mouse_desc_t udi_hid_mouse

#define UDI_COMPOSITE_DESC_FS \
    .udi_cdc_iad               = UDI_CDC_IAD_DESC_0, \
    .udi_cdc_comm              = UDI_CDC_COMM_DESC_0, \
    .udi_cdc_data              = UDI_CDC_DATA_DESC_0_FS, \
    .udi_msc                   = UDI_MSC_DESC_FS, \
    .udi_hid_mouse             = UDI_HID_MOUSE_DESC

#define UDI_COMPOSITE_DESC_HS \
    .udi_cdc_iad               = UDI_CDC_IAD_DESC_0, \
    .udi_cdc_comm              = UDI_CDC_COMM_DESC_0, \
    .udi_cdc_data              = UDI_CDC_DATA_DESC_0_HS, \
    .udi_msc                   = UDI_MSC_DESC_HS, \
    .udi_hid_mouse             = UDI_HID_MOUSE_DESC

#define UDI_COMPOSITE_API  \
    &udi_api_cdc_comm,       \
    &udi_api_cdc_data,       \
    &udi_api_msc,            \
    &udi_api_hid_mouse
*/




/* Example of include for interface
#include "udi_msc.h"
#include "udi_hid_kbd.h"
#include "udi_hid_mouse.h"
#include "udi_cdc.h"
#include "udi_phdc.h"
#include "udi_vendor.h"
*/
/* Declaration of callbacks used by USB
#include "callback_def.h"
*/

#endif // _CONF_USB_H_
```

4.2. conf_clock.h

4.2.1. SAM3X and SAM3A Devices (UOTGHS: USB OTG High Speed)

```
/*
 * Support and FAQ: visit <a href="http://www.atmel.com/design-support/">Atmel
Support</a>
 */

#ifndef CONF_CLOCK_H_INCLUDED
#define CONF_CLOCK_H_INCLUDED

// ===== System Clock (MCK) Source Options
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_SLCK_RC
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_SLCK_XTAL
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_SLCK_BYPASS
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_MAINCK_4M_RC
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_MAINCK_8M_RC
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_MAINCK_12M_RC
```

```
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_MAINCK_XTAL
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_MAINCK_BYPASS
#define CONFIG_SYSCLK_SOURCE          SYSCLK_SRC_PLLACK
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_UPLLCK

// ===== System Clock (MCK) Prescaler Options   (Fmck = Fsys / (SYSCLK_PRES))
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_1
#define CONFIG_SYSCLK_PRES            SYSCLK_PRES_2
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_4
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_8
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_16
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_32
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_64
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_3

// ===== PLL0 (A) Options   (Fpll = (Fclk * PLL_mul) / PLL_div)
// Use mul and div effective values here.
#define CONFIG_PLL0_SOURCE            PLL_SRC_MAINCK_XTAL
#define CONFIG_PLL0_MUL               14
#define CONFIG_PLL0_DIV               1

// ===== UPLL (UTMI) Hardware fixed at 480MHz.

// ===== USB Clock Source Options   (Fusb = FpllX / USB_div)
// Use div effective value here.
//#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_PLL0
#define CONFIG_USBCLK_SOURCE          USBCLK_SRC_UPLL
#define CONFIG_USBCLK_DIV             1

// ===== Target frequency (System clock)
// - XTAL frequency: 12MHz
// - System clock source: PLLA
// - System clock prescaler: 2 (divided by 2)
// - PLLA source: XTAL
// - PLLA output: XTAL * 14 / 1
// - System clock is: 12 * 14 / 1 /2 = 84MHz
// ===== Target frequency (USB Clock)
// - USB clock source: UPLL
// - USB clock divider: 1 (not divided)
// - UPLL frequency: 480MHz
// - USB clock: 480 / 1 = 480MHz


#endif /* CONF_CLOCK_H_INCLUDED */
```

4.2.2. SAM4L Device (USBC)

```
/*
 * Support and FAQ: visit <a href="http://www.atmel.com/design-support/">Atmel
Support</a>
 */
#ifndef CONF_CLOCK_H_INCLUDED
#define CONF_CLOCK_H_INCLUDED

//#define CONFIG_SYSCLK_INIT_CPUMASK  (1 << SYSCLK_OCD)
//#define CONFIG_SYSCLK_INIT_PBAMASK  (1 << SYSCLK_IISC)
//#define CONFIG_SYSCLK_INIT_PBBMASK  (1 << SYSCLK_USBC_REGS)
//#define CONFIG_SYSCLK_INIT_PBCMASK  (1 << SYSCLK_CHIPID)
//#define CONFIG_SYSCLK_INIT_PBDMASK  (1 << SYSCLK_AST)
//#define CONFIG_SYSCLK_INIT_HSBMASK  (1 << SYSCLK_PDCA_HSB)

//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_RCSYS
```

```
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_OSC0
#define CONFIG_SYSCLK_SOURCE          SYSCLK_SRC_PLL0
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_DFLL
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_RC80M
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_RCFAST
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_RC1M

/* RCFAST frequency selection: 0 for 4MHz, 1 for 8MHz and 2 for 12MHz */
//#define CONFIG_RCFAST_FRANGE    0
//#define CONFIG_RCFAST_FRANGE    1
//#define CONFIG_RCFAST_FRANGE    2

/* Fbus = Fsys / (2 ^ BUS_div) */
#define CONFIG_SYSCLK_CPU_DIV         0
#define CONFIG_SYSCLK_PBA_DIV         0
#define CONFIG_SYSCLK_PBB_DIV         0
#define CONFIG_SYSCLK_PBC_DIV         0
#define CONFIG_SYSCLK_PBD_DIV         0

// ===== Disable all non-essential peripheral clocks
//#define CONFIG_SYSCLK_INIT_CPUMASK  0
//#define CONFIG_SYSCLK_INIT_PBAMASK  SYSCLK_USART1
//#define CONFIG_SYSCLK_INIT_PBBMASK  0
//#define CONFIG_SYSCLK_INIT_PBCMASK  0
//#define CONFIG_SYSCLK_INIT_PBDMASK  0
//#define CONFIG_SYSCLK_INIT_HSBMASK  0

// ===== PLL Options
#define CONFIG_PLL0_SOURCE          PLL_SRC_OSC0
//#define CONFIG_PLL0_SOURCE        PLL_SRC_GCLK9

/* Fpll0 = (Fclk * PLL_mul) / PLL_div */
//#define CONFIG_PLL0_MUL           (48000000UL / BOARD_OSC0_HZ)
//#define CONFIG_PLL0_DIV           1
#define CONFIG_PLL0_MUL               (192000000 / FOSC0) /* Fpll = (Fclk *
PLL_mul) / PLL_div */
#define CONFIG_PLL0_DIV               4 /* Fpll = (Fclk * PLL_mul) / PLL_div */

// ==== DFLL Options
//#define CONFIG_DFLL0_SOURCE         GENCLK_SRC_OSC0
//#define CONFIG_DFLL0_SOURCE         GENCLK_SRC_RCSYS
//#define CONFIG_DFLL0_SOURCE         GENCLK_SRC_OSC32K
//#define CONFIG_DFLL0_SOURCE         GENCLK_SRC_RC120M
//#define CONFIG_DFLL0_SOURCE         GENCLK_SRC_RC32K

/* Fdfll = (Fclk * DFLL_mul) / DFLL_div */
//#define CONFIG_DFLL0_FREQ           48000000UL
//#define CONFIG_DFLL0_MUL            ((4 * CONFIG_DFLL0_FREQ) / BOARD_OSC32_HZ)
//#define CONFIG_DFLL0_DIV            4
//#define CONFIG_DFLL0_MUL            (CONFIG_DFLL0_FREQ / BOARD_OSC32_HZ)
//#define CONFIG_DFLL0_DIV            1

// ===== USB Clock Source Options
//#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_OSC0
#define CONFIG_USBCLK_SOURCE          USBCLK_SRC_PLL0
//#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_DFLL

/* Fusb = Fsys / USB_div */
#define CONFIG_USBCLK_DIV             1

// ===== GCLK9 option
//#define CONFIG_GCLK9_SOURCE           GENCLK_SRC_GCLKIN0
```

```
//#define CONFIG_GCLK9_DIV              1

#endif /* CONF_CLOCK_H_INCLUDED */
```

4.3. conf_clocks.h

4.3.1. SAM D21 Device (USB)

```
/*
 * Support and FAQ: visit <a href="http://www.atmel.com/design-support/">Atmel
Support</a>
 */
#include <clock.h>

#ifndef CONF_CLOCKS_H_INCLUDED
#  define CONF_CLOCKS_H_INCLUDED

/* System clock bus configuration */
#  define CONF_CLOCK_CPU_CLOCK_FAILURE_DETECT     false
#  define CONF_CLOCK_FLASH_WAIT_STATES            2
#  define CONF_CLOCK_CPU_DIVIDER                  SYSTEM_MAIN_CLOCK_DIV_1
#  define CONF_CLOCK_APBA_DIVIDER                 SYSTEM_MAIN_CLOCK_DIV_1
#  define CONF_CLOCK_APBB_DIVIDER                 SYSTEM_MAIN_CLOCK_DIV_1
#  define CONF_CLOCK_APBC_DIVIDER                 SYSTEM_MAIN_CLOCK_DIV_1

/* SYSTEM_CLOCK_SOURCE_OSC8M configuration - Internal 8MHz oscillator */
#  define CONF_CLOCK_OSC8M_PRESCALER              SYSTEM_OSC8M_DIV_1
#  define CONF_CLOCK_OSC8M_ON_DEMAND              true
#  define CONF_CLOCK_OSC8M_RUN_IN_STANDBY         false

/* SYSTEM_CLOCK_SOURCE_XOSC configuration - External clock/oscillator */
#  define CONF_CLOCK_XOSC_ENABLE                  false
#  define CONF_CLOCK_XOSC_EXTERNAL_CRYSTAL
SYSTEM_CLOCK_EXTERNAL_CRYSTAL
#  define CONF_CLOCK_XOSC_EXTERNAL_FREQUENCY      12000000UL
#  define CONF_CLOCK_XOSC_STARTUP_TIME            SYSTEM_XOSC_STARTUP_32768
#  define CONF_CLOCK_XOSC_AUTO_GAIN_CONTROL       true
#  define CONF_CLOCK_XOSC_ON_DEMAND               true
#  define CONF_CLOCK_XOSC_RUN_IN_STANDBY          false

/* SYSTEM_CLOCK_SOURCE_XOSC32K configuration - External 32KHz crystal/clock
oscillator */
#  define CONF_CLOCK_XOSC32K_ENABLE               false
#  define CONF_CLOCK_XOSC32K_EXTERNAL_CRYSTAL
SYSTEM_CLOCK_EXTERNAL_CRYSTAL
#  define CONF_CLOCK_XOSC32K_STARTUP_TIME
SYSTEM_XOSC32K_STARTUP_65536
#  define CONF_CLOCK_XOSC32K_AUTO_AMPLITUDE_CONTROL  false
#  define CONF_CLOCK_XOSC32K_ENABLE_1KHZ_OUPUT    false
#  define CONF_CLOCK_XOSC32K_ENABLE_32KHZ_OUTPUT  true
#  define CONF_CLOCK_XOSC32K_ON_DEMAND            true
#  define CONF_CLOCK_XOSC32K_RUN_IN_STANDBY       false

/* SYSTEM_CLOCK_SOURCE_OSC32K configuration - Internal 32KHz oscillator */
#  define CONF_CLOCK_OSC32K_ENABLE                false
#  define CONF_CLOCK_OSC32K_STARTUP_TIME          SYSTEM_OSC32K_STARTUP_130
#  define CONF_CLOCK_OSC32K_ENABLE_1KHZ_OUTPUT    true
#  define CONF_CLOCK_OSC32K_ENABLE_32KHZ_OUTPUT   true
#  define CONF_CLOCK_OSC32K_ON_DEMAND             true
#  define CONF_CLOCK_OSC32K_RUN_IN_STANDBY        false
```

```
/* SYSTEM_CLOCK_SOURCE_DFLL configuration - Digital Frequency Locked Loop */
#  define CONF_CLOCK_DFLL_ENABLE                    true
#  define CONF_CLOCK_DFLL_LOOP_MODE
SYSTEM_CLOCK_DFLL_LOOP_MODE_USB_RECOVERY
#  define CONF_CLOCK_DFLL_ON_DEMAND                 true

/* DFLL open loop mode configuration */
#  define CONF_CLOCK_DFLL_FINE_VALUE                (512)

/* DFLL closed loop mode configuration */
#  define CONF_CLOCK_DFLL_SOURCE_GCLK_GENERATOR   GCLK_GENERATOR_1
#  define CONF_CLOCK_DFLL_MULTIPLY_FACTOR         (48000000 / 32768)
#  define CONF_CLOCK_DFLL_QUICK_LOCK              true
#  define CONF_CLOCK_DFLL_TRACK_AFTER_FINE_LOCK   true
#  define CONF_CLOCK_DFLL_KEEP_LOCK_ON_WAKEUP     true
#  define CONF_CLOCK_DFLL_ENABLE_CHILL_CYCLE      true
#  define CONF_CLOCK_DFLL_MAX_COARSE_STEP_SIZE    (0x1f / 4)
#  define CONF_CLOCK_DFLL_MAX_FINE_STEP_SIZE      (0xff / 4)

/* SYSTEM_CLOCK_SOURCE_DPLL configuration - Digital Phase-Locked Loop */
#  define CONF_CLOCK_DPLL_ENABLE                    false
#  define CONF_CLOCK_DPLL_ON_DEMAND                 true
#  define CONF_CLOCK_DPLL_RUN_IN_STANDBY            false
#  define CONF_CLOCK_DPLL_LOCK_BYPASS               false
#  define CONF_CLOCK_DPLL_WAKE_UP_FAST              false
#  define CONF_CLOCK_DPLL_LOW_POWER_ENABLE          false

#  define CONF_CLOCK_DPLL_LOCK_TIME
SYSTEM_CLOCK_SOURCE_DPLL_LOCK_TIME_DEFAULT
#  define CONF_CLOCK_DPLL_REFERENCE_CLOCK
SYSTEM_CLOCK_SOURCE_DPLL_REFERENCE_CLOCK_XOSC32K
#  define CONF_CLOCK_DPLL_FILTER
SYSTEM_CLOCK_SOURCE_DPLL_FILTER_DEFAULT

#  define CONF_CLOCK_DPLL_REFERENCE_FREQUENCY     32768
#  define CONF_CLOCK_DPLL_REFERENCE_DIVIDER       1
#  define CONF_CLOCK_DPLL_OUTPUT_FREQUENCY        48000000

/* DPLL GCLK reference configuration */
#  define CONF_CLOCK_DPLL_REFERENCE_GCLK_GENERATOR  GCLK_GENERATOR_1
/* DPLL GCLK lock timer configuration */
#  define CONF_CLOCK_DPLL_LOCK_GCLK_GENERATOR     GCLK_GENERATOR_1

/* Set this to true to configure the GCLK when running clocks_init. If set to
 * false, none of the GCLK generators will be configured in clocks_init(). */
#  define CONF_CLOCK_CONFIGURE_GCLK                 true

/* Configure GCLK generator 0 (Main Clock) */
#  define CONF_CLOCK_GCLK_0_ENABLE                  true
#  define CONF_CLOCK_GCLK_0_RUN_IN_STANDBY          true
#  define CONF_CLOCK_GCLK_0_CLOCK_SOURCE            SYSTEM_CLOCK_SOURCE_DFLL
#  define CONF_CLOCK_GCLK_0_PRESCALER               1
#  define CONF_CLOCK_GCLK_0_OUTPUT_ENABLE           false

/* Configure GCLK generator 1 */
#  define CONF_CLOCK_GCLK_1_ENABLE                  false
#  define CONF_CLOCK_GCLK_1_RUN_IN_STANDBY          false
#  define CONF_CLOCK_GCLK_1_CLOCK_SOURCE
SYSTEM_CLOCK_SOURCE_XOSC32K
#  define CONF_CLOCK_GCLK_1_PRESCALER               1
#  define CONF_CLOCK_GCLK_1_OUTPUT_ENABLE           false
```

```
/* Configure GCLK generator 2 (RTC) */
#  define CONF_CLOCK_GCLK_2_ENABLE                 false
#  define CONF_CLOCK_GCLK_2_RUN_IN_STANDBY         false
#  define CONF_CLOCK_GCLK_2_CLOCK_SOURCE
SYSTEM_CLOCK_SOURCE_OSC32K
#  define CONF_CLOCK_GCLK_2_PRESCALER              32
#  define CONF_CLOCK_GCLK_2_OUTPUT_ENABLE          false

/* Configure GCLK generator 3 */
#  define CONF_CLOCK_GCLK_3_ENABLE                 false
#  define CONF_CLOCK_GCLK_3_RUN_IN_STANDBY         false
#  define CONF_CLOCK_GCLK_3_CLOCK_SOURCE           SYSTEM_CLOCK_SOURCE_OSC8M
#  define CONF_CLOCK_GCLK_3_PRESCALER              1
#  define CONF_CLOCK_GCLK_3_OUTPUT_ENABLE          false

/* Configure GCLK generator 4 */
#  define CONF_CLOCK_GCLK_4_ENABLE                 false
#  define CONF_CLOCK_GCLK_4_RUN_IN_STANDBY         false
#  define CONF_CLOCK_GCLK_4_CLOCK_SOURCE           SYSTEM_CLOCK_SOURCE_OSC8M
#  define CONF_CLOCK_GCLK_4_PRESCALER              1
#  define CONF_CLOCK_GCLK_4_OUTPUT_ENABLE          false

/* Configure GCLK generator 5 */
#  define CONF_CLOCK_GCLK_5_ENABLE                 false
#  define CONF_CLOCK_GCLK_5_RUN_IN_STANDBY         false
#  define CONF_CLOCK_GCLK_5_CLOCK_SOURCE           SYSTEM_CLOCK_SOURCE_OSC8M
#  define CONF_CLOCK_GCLK_5_PRESCALER              1
#  define CONF_CLOCK_GCLK_5_OUTPUT_ENABLE          false

/* Configure GCLK generator 6 */
#  define CONF_CLOCK_GCLK_6_ENABLE                 false
#  define CONF_CLOCK_GCLK_6_RUN_IN_STANDBY         false
#  define CONF_CLOCK_GCLK_6_CLOCK_SOURCE           SYSTEM_CLOCK_SOURCE_OSC8M
#  define CONF_CLOCK_GCLK_6_PRESCALER              1
#  define CONF_CLOCK_GCLK_6_OUTPUT_ENABLE          false

/* Configure GCLK generator 7 */
#  define CONF_CLOCK_GCLK_7_ENABLE                 false
#  define CONF_CLOCK_GCLK_7_RUN_IN_STANDBY         false
#  define CONF_CLOCK_GCLK_7_CLOCK_SOURCE           SYSTEM_CLOCK_SOURCE_OSC8M
#  define CONF_CLOCK_GCLK_7_PRESCALER              1
#  define CONF_CLOCK_GCLK_7_OUTPUT_ENABLE          false

/* Configure GCLK generator 8 */
#  define CONF_CLOCK_GCLK_8_ENABLE                 false
#  define CONF_CLOCK_GCLK_8_RUN_IN_STANDBY         false
#  define CONF_CLOCK_GCLK_8_CLOCK_SOURCE           SYSTEM_CLOCK_SOURCE_OSC8M
#  define CONF_CLOCK_GCLK_8_PRESCALER              1
#  define CONF_CLOCK_GCLK_8_OUTPUT_ENABLE          false

#endif /* CONF_CLOCKS_H_INCLUDED */
```

4.4. conf_board.h

4.4.1. SAM3X and SAM3A Devices (UOTGHS: USB OTG High Speed)

```
/*
 * Support and FAQ: visit <a href="http://www.atmel.com/design-support/">Atmel
```

```
Support</a>
 */

#ifndef CONF_BOARD_H_INCLUDED
#define CONF_BOARD_H_INCLUDED

// USB pins are used
#define CONF_BOARD_USB_PORT

#endif /* CONF_BOARD_H_INCLUDED */
```

4.4.2. SAM4L Device (USBC)

```
/*
 * Support and FAQ: visit <a href="http://www.atmel.com/design-support/">Atmel
Support</a>
 */

#ifndef CONF_BOARD_H_INCLUDED
#define CONF_BOARD_H_INCLUDED

// Auto-initialize USART GPIOs when board_init() is called
//#define CONF_BOARD_COM_PORT

// Enable USB interface (USB)
#define CONF_BOARD_USB_PORT

#endif /* CONF_BOARD_H_INCLUDED */
```

4.4.3. SAM D21 Device (USB)

```
/*
 * Support and FAQ: visit <a href="http://www.atmel.com/design-support/">Atmel
Support</a>
 */

#ifndef CONF_BOARD_H_INCLUDED
#define CONF_BOARD_H_INCLUDED

/* Enable USB VBUS detect */
#define CONF_BOARD_USB_VBUS_DETECT

#endif /* CONF_BOARD_H_INCLUDED */
```

5. USB Device Basic Setup

5.1. Custom Configuration

The following USB Device configuration must be included in the conf_usb.h file of the application:

1. USB_DEVICE_VENDOR_ID (Word).

Vendor ID provided by USB org (Atmel 0x03EB).

2. USB_DEVICE_PRODUCT_ID (Word).

Product ID (Referenced in usb_atmel.h).

3. USB_DEVICE_MAJOR_VERSION (Byte).

Major version of the device.

4. USB_DEVICE_MINOR_VERSION (Byte).

Minor version of the device.

5. USB_DEVICE_MANUFACTURE_NAME (string).

ASCII name for the manufacture.

6. USB_DEVICE_PRODUCT_NAME (string).

ASCII name for the product.

7. USB_DEVICE_SERIAL_NAME (string).

ASCII name to enable and set a serial number.

8. USB_DEVICE_POWER (Numeric).

(unit mA) Maximum device power.

9. USB_DEVICE_ATTR (Byte).

USB attributes available:

- USB_CONFIG_ATTR_SELF_POWERED
- USB_CONFIG_ATTR_REMOTE_WAKEUP

Note: If remote wake is enabled, this defines remotewakeup callbacks.

10. USB_DEVICE_LOW_SPEED (Only defined).

Force the USB Device to run in low speed.

11. USB_DEVICE_HS_SUPPORT (Only defined).

Authorize the USB Device to run in high speed.

12. USB_DEVICE_MAX_EP (Byte).

Define the maximum endpoint number used by the USB Device.

This one is already defined in the UDI default configuration. E.g.:

- When endpoint control 0x00, endpoint 0x01, and endpoint 0x82 is used, then USB_DEVICE_MAX_EP=2
- When only endpoint control 0x00 is used, then USB_DEVICE_MAX_EP=0

- When endpoint 0x01 and endpoint 0x81 is used, then USB_DEVICE_MAX_EP=1 (configuration not possible on USBB interface)

5.2. VBUS Monitoring

The VBUS monitoring is used only for USB SELF Power application.

- By default the USB device is automatically attached when VBUS is high or when USB starts for devices without internal VBUS monitoring. conf_usb.h file does not contain definition USB_DEVICE_ATTACH_AUTO_DISABLE.

```
//#define USB_DEVICE_ATTACH_AUTO_DISABLE
```

- Add custom VBUS monitoring. conf_usb.h file contains define USB_DEVICE_ATTACH_AUTO_DISABLE:

```
#define USB_DEVICE_ATTACH_AUTO_DISABLE
```

User C-file contains:

```
// Authorize VBUS monitoring
if (!udc_include_vbus_monitoring()) {
  // Implement custom VBUS monitoring via GPIO or other
}
Event_VBUS_present() // VBUS interrupt or GPIO interrupt or other
{
  // Attach USB Device
  udc_attach();
}
```

- Case of battery charging. conf_usb.h file contains define USB_DEVICE_ATTACH_AUTO_DISABLE:

```
#define USB_DEVICE_ATTACH_AUTO_DISABLE
```

User C-file contains:

```
Event VBUS present() // VBUS interrupt or GPIO interrupt or ..
{
  // Authorize battery charging, but wait key press to start USB.
}
Event Key press()
{
  // Stop batteries charging
  // Start USB
  udc_attach();
}
```

5.3. USB Device Basic Setup

5.3.1. USB Device Controller (UDC) - Prerequisites

Common prerequisites for all USB devices.

This module is based on USB device stack full interrupt driven, and supporting sleepmgr. For AVR® and Atmel® | SMART ARM®-based SAM3/4 devices the clock services is supported. For SAM D21 devices the clock driver is supported.

The following procedure must be executed to set up the project correctly:

- Specify the clock configuration:
  - XMEGA® USB devices need 48MHz clock input. XMEGA USB devices need CPU frequency higher than 12MHz. You can use either an internal RC 48MHz auto calibrated by Start of Frames or an external OSC.
  - UC3 and SAM3/4 devices without USB high speed support need 48MHz clock input. You must use a PLL and an external OSC.
  - UC3 and SAM3/4 devices with USB high speed support need 12MHz clock input. You must use an external OSC.
  - UC3 devices with USBC hardware need CPU frequency higher than 25MHz
  - SAM D21 devices without USB high speed support need 48MHz clock input. You should use DFLL with USBCRM.
- In conf_board.h, the define CONF_BOARD_USB_PORT must be added to enable USB lines. (Not mandatory for all boards).
- Enable interrupts
- Initialize the clock service

The usage of sleep manager service is optional, but recommended to reduce power consumption:

- Initialize the sleep manager service
- Activate sleep mode when the application is in IDLE state

For AVR and SAM3/4 devices, add to the initialization code:

```
sysclk_init();
irq_initialize_vectors();
cpu_irq_enable();
board_init();
sleepmgr_init(); // Optional
```

For SAM D21 devices, add to the initialization code:

```
system_init();
irq_initialize_vectors();
cpu_irq_enable();
sleepmgr_init(); // Optional
```

Add to the main IDLE loop:

```
sleepmgr_enter_sleep(); // Optional
```

5.3.2. USB Device Controller (UDC) - Example Code

Common example code for all USB devices.

Content of conf_usb.h:

```
#define USB_DEVICE_VENDOR_ID 0x03EB
#define USB_DEVICE_PRODUCT_ID 0xXXXX
#define USB_DEVICE_MAJOR_VERSION 1
#define USB_DEVICE_MINOR_VERSION 0
#define USB_DEVICE_POWER 100
#define USB_DEVICE_ATTR USB_CONFIG_ATTR_BUS_POWERED
```

Add to application C-file:

```
void usb_init(void)
{
```

```
  udc_start();
}
```

5.3.3. USB Device Controller (UDC) - Workflow

Common workflow for all USB devices.

1. Ensure that conf_usb.h is available and contains the following configuration, which is the main USB device configuration:

```
// Vendor ID provided by USB org (Atmel 0x03EB)
#define USB_DEVICE_VENDOR_ID 0x03EB // Type Word
// Product ID (Atmel PID referenced in usb_atmel.h)
#define USB_DEVICE_PRODUCT_ID 0xXXXX // Type Word
// Major version of the device
#define USB_DEVICE_MAJOR_VERSION 1 // Type Byte
// Minor version of the device
#define USB_DEVICE_MINOR_VERSION 0 // Type Byte
// Maximum device power (mA)
#define USB_DEVICE_POWER 100 // Type 9-bits
// USB attributes to enable features
#define USB_DEVICE_ATTR USB_CONFIG_ATTR_BUS_POWERED // Flags
```

2. Call the USB device stack start function to enable stack and start USB:

```
udc_start();
```

Note: In case of USB dual roles (Device and Host) managed through USB OTG connector (USB ID pin), the call of `udc_start()` must be removed and replaced by uhc_start(). Refer to section "Dual roles" for further information in the application note: Atmel AVR4950: ASF - USB Host Stack

5.4. conf_clock.h Examples

Content of XMEGA conf_clock.h:

```
// Configuration based on internal RC:
// USB clock need of 48MHz
#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_RCOSC
#define CONFIG_OSC_RC32_CAL         48000000UL
#define CONFIG_OSC_AUTOCAL_RC32MHZ_REF_OSC  OSC_ID_USBSOF
// CPU clock need of clock > 12MHz to run with USB (Here 24MHz)
#define CONFIG_SYSCLK_SOURCE     SYSCLK_SRC_RC32MHZ
#define CONFIG_SYSCLK_PSADIV     SYSCLK_PSADIV_2
#define CONFIG_SYSCLK_PSBCDIV    SYSCLK_PSBCDIV_1_1
```

Content of conf_clock.h for AT32UC3A0, AT32UC3A1, and AT32UC3B devices (USBB):

```
// Configuration based on 12MHz external OSC:
#define CONFIG_PLL1_SOURCE          PLL_SRC_OSC0
#define CONFIG_PLL1_MUL             8
#define CONFIG_PLL1_DIV             2
#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_PLL1
#define CONFIG_USBCLK_DIV           1 // Fusb = Fsys/(2 ^ USB_div)
```

Content of conf_clock.h for AT32UC3A3 and AT32UC3A4 devices (USBB with high speed support):

```
// Configuration based on 12MHz external OSC:
#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_OSC0
#define CONFIG_USBCLK_DIV           1 // Fusb = Fsys/(2 ^ USB_div)
```

Content of conf_clock.h for AT32UC3C, ATUCXXD, ATUCXXL3U, and ATUCXXL4U devices (USBC):

```
// Configuration based on 12MHz external OSC:
#define CONFIG_PLL1_SOURCE          PLL_SRC_OSC0
#define CONFIG_PLL1_MUL             8
#define CONFIG_PLL1_DIV             2
#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_PLL1
#define CONFIG_USBCLK_DIV           1 // Fusb = Fsys/(2 ^ USB_div)
// CPU clock need of clock > 25MHz to run with USBC
#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_PLL1
```

Content of conf_clock.h for SAM3S, SAM3SD, and SAM4S devices (UPD: USB Peripheral Device):

```
// PLL1 (B) Options   (Fpll = (Fclk * PLL_mul) / PLL_div)
#define CONFIG_PLL1_SOURCE          PLL_SRC_MAINCK_XTAL
#define CONFIG_PLL1_MUL             16
#define CONFIG_PLL1_DIV             2
// USB Clock Source Options   (Fusb = FpllX / USB_div)
#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_PLL1
#define CONFIG_USBCLK_DIV           2
```

Content of conf_clock.h for SAM3U device (UPDHS: USB Peripheral Device High Speed):

```
// USB Clock Source fixed at UPLL.
```

Content of conf_clock.h for SAM3X and SAM3A devices (UOTGHS: USB OTG High Speed):

```
// USB Clock Source fixed at UPLL.
#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_UPLL
#define CONFIG_USBCLK_DIV           1
```

Content of conf_clocks.h for SAM D21 devices (USB):

```
// System clock bus configuration
#  define CONF_CLOCK_FLASH_WAIT_STATES            2

// USB Clock Source fixed at DFLL.
// SYSTEM_CLOCK_SOURCE_DFLL configuration - Digital Frequency Locked Loop
#  define CONF_CLOCK_DFLL_ENABLE                  true
#  define CONF_CLOCK_DFLL_LOOP_MODE
SYSTEM_CLOCK_DFLL_LOOP_MODE_USB_RECOVERY
#  define CONF_CLOCK_DFLL_ON_DEMAND               true

// Set this to true to configure the GCLK when running clocks_init.
// If set to false, none of the GCLK generators will be configured in clocks_init().
#  define CONF_CLOCK_CONFIGURE_GCLK               true

// Configure GCLK generator 0 (Main Clock)
#  define CONF_CLOCK_GCLK_0_ENABLE                true
#  define CONF_CLOCK_GCLK_0_RUN_IN_STANDBY        true
#  define CONF_CLOCK_GCLK_0_CLOCK_SOURCE          SYSTEM_CLOCK_SOURCE_DFLL
#  define CONF_CLOCK_GCLK_0_PRESCALER             1
#  define CONF_CLOCK_GCLK_0_OUTPUT_ENABLE         false
```

6. Document Revision History

Doc. Rev.	Date	Comments
42345B	12/2015	Fixed typos
42345A	12/2014	Initial release

Atmel Corporation 1600 Technology Drive, San Jose, CA 95110 USA **T:** (+1)(408) 441.0311 **F:** (+1)(408) 436.4200 | **www.atmel.com**