Atmel Microcontrollers

AT09339: USB Host Interface (UHI) for Mass Storage Class (MSC)

APPLICATION NOTE

Introduction

USB Host Interface (UHI) for Mass Storage Class (MSC) provides an interface for the configuration and management of USB MSC host.

The outline of this documentation is as follows:

- API Overview
- Quick Start Guide for USB Host Mass-Storage Module (UHI MSC)
- Configuration File Examples

For more details for Atmel® Software Framework (ASF) USB Host Stack, refer to following application note:

- AVR4950: ASF - USB Host Stack

Table of Contents

1. Software License

2. API Overview

2.1. Variable and Type Definitions

2.1.1. Type uhi_msc_scsi_callback_t

```
typedef void(* uhi_msc_scsi_callback_t )(bool)
```

Callback type used by uhi_msc_scsi() functions.

2.2. Structure Definitions

2.2.1. Struct uhi_msc_lun_t

Logical Unit Number (LUN) structure information.

Table 2-1. Members

Type	Name	Description
bool	b_write_protected	Write protect enable
struct sbc_read_capacity10_data	capacity	SBC-2 read capacity (10) parameter data
lun_status_t	status	Status of LUN

2.3. Macro Definitions

2.3.1. Interface with USB Host Core (UHC)

Definition and functions required by UHC.

2.3.1.1. Macro UHI_MSC

```
#define UHI_MSC
```

Global definition which contains standard UHI API for UHC. It must be added in USB_HOST_UHI definition from conf_usb_host.h file.

2.4. Function Definitions

2.4.1. Functions Required by UHC

2.4.1.1. Function uhi_msc_install()

Install interface.

```
uhc_enum_status_t uhi_msc_install(
        uhc_device_t * dev)
```

Allocate interface endpoints if supported.

Table 2-2. Parameters

Data direction	Parameter name	Description
[in]	uhc_device_t	Device to request

Returns
Status of the install.

2.4.1.2. Function uhi_msc_enable()

Enable the interface.

```
void uhi_msc_enable(
        uhc_device_t * dev)
```

Enable a USB interface corresponding to UHI.

Table 2-3. Parameters

Data direction	Parameter name	Description
[in]	uhc_device_t	Device to request

2.4.1.3. Function uhi_msc_uninstall()

Uninstall the interface (if installed).

```
void uhi_msc_uninstall(
        uhc_device_t * dev)
```

Table 2-4. Parameters

Data direction	Parameter name	Description
[in]	uhc_device_t	Device to request

2.4.2. UHI for Mass Storage Class

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

2.4.2.1. Function uhi_msc_is_available()

Tests if the interface UHI Mass Storage is available.

```
bool uhi_msc_is_available( void )
```

The UHI Mass Storage can be busy during the enumeration of a USB Device MSC.

Returns
True, if UHI Mass Storage is available.

2.4.2.2. Function uhi_msc_get_lun()

Gives the number of LUN available.

```
uint8_t uhi_msc_get_lun( void )
```

Note: A LUN can be available, but with a status LUN_NOT_PRESENT.

It is the case for a card reader without card.

Returns
Number of LUN available.

2.4.2.3. Function uhi_msc_get_lun_desc()

Gives information about a LUN.

```
uhi_msc_lun_t * uhi_msc_get_lun_desc(
        uint8_t lun)
```

Table 2-5. Parameters

Data direction	Parameter name	Description
[in]	lun	LUN number

Returns
Pointer on the LUN information structure.

2.4.2.4. Function uhi_msc_scsi_test_unit_ready()

Checks and update the status of the LUN.

```
bool uhi_msc_scsi_test_unit_ready(
        uint8_t lun,
        uhi_msc_scsi_callback_t callback)
```

Table 2-6. Parameters

Data direction	Parameter name	Description
[in]	lun	LUN number
[in]	callback	Callback to call at the end of SCSI command

Returns
True, if the Small Computer System Interface (SCSI) command has been accepted.

2.4.2.5. Function uhi_msc_scsi_read_10()

Reads a LUN data section to RAM buffer.

```
bool uhi_msc_scsi_read_10(
        uint8_t lun,
        uint32_t addr,
        uint8_t * ram,
        uint8_t nb_sector,
        uhi_msc_scsi_callback_t callback)
```

Note: The sector size used to define the data section is the sector size returned by LUN in field.

Table 2-7. Parameters

Data direction	Parameter name	Description
[in]	lun	LUN number
[in]	addr	Sector address to read
[out]	ram	RAM address used to store the data
[in]	nb_sector	Number of sector to read
[in]	callback	Callback to call at the end of SCSI command

Returns
True, if the SCSI command has been accepted.

2.4.2.6. Function uhi_msc_scsi_write_10()

Writes a RAM buffer in a LUN data section.

```
bool uhi_msc_scsi_write_10(
        uint8_t lun,
        uint32_t addr,
        const uint8_t * ram,
        uint8_t nb_sector,
        uhi_msc_scsi_callback_t callback)
```

Note: The sector size used to define the data section is the sector size returned by LUN in field.

Table 2-8. Parameters

Data direction	Parameter name	Description
[in]	lun	LUN number
[in]	addr	Sector address to write
[in]	ram	RAM address of data to write
[in]	nb_sector	Number of sector to write
[in]	callback	Callback to call at the end of SCSI command

Returns
True, if the SCSI command has been accepted.

2.4.3. USB Host Mass Storage Interface for Control Access Module

Layer added on UHI MSC interface to allow the usage of control access module. The control access module provides a common access at all memories and it is used by the File Systems available in ASF.

2.4.3.1. Function uhi_msc_mem_get_lun()

Gives the number of available LUN.

```
uint8_t uhi_msc_mem_get_lun( void )
```

Note: A LUN can be available, but with a status not present.

It is the case for a card reader without card.

Returns
Number of available LUN.

2.4.3.2. Function uhi_msc_mem_test_unit_ready()

Checks and update the status of the LUN.

```
Ctrl_status uhi_msc_mem_test_unit_ready(
        uint8_t lun)
```

Table 2-9. Parameters

Data direction	Parameter name	Description
[in]	lun	LUN number

Returns
Status of the LUN.

2.4.3.3. Function uhi_msc_mem_read_capacity()

Returns the capacity of the LUN.

```
Ctrl_status uhi_msc_mem_read_capacity(
        uint8_t lun,
        uint32_t * u32_nb_sector)
```

Table 2-10. Parameters

Data direction	Parameter name	Description
[in]	lun	LUN number
[in]	u32_nb_sector	Pointer to store the last sector address possible on this LUN

Returns
Status of the LUN.

2.4.3.4. Function uhi_msc_mem_read_sector_size()

Returns the sector size of the LUN.

```
uint8_t uhi_msc_mem_read_sector_size(
        uint8_t lun)
```

Table 2-11. Parameters

Data direction	Parameter name	Description
[in]	lun	LUN number

Returns
Sector size (unit 512B).

2.4.3.5. Function uhi_msc_mem_wr_protect()

Checks if the LUN is write protected.

```
bool uhi_msc_mem_wr_protect(
        uint8_t lun)
```

Table 2-12. Parameters

Data direction	Parameter name	Description
[in]	lun	LUN number

Returns
True, if write protected.

2.4.3.6. Function uhi_msc_mem_removal()

Checks if the device is removed.

```
bool uhi_msc_mem_removal( void )
```

Returns
Always true for USB Device.

2.4.3.7. Function uhi_msc_mem_read_10_ram()

Reads 512 bytes from the current LUN.

```
Ctrl_status uhi_msc_mem_read_10_ram(
        uint32_t addr,
        void * ram)
```

The LUN is selected by uhi_msc_mem_test_unit_ready() or uhi_msc_mem_read_capacity() function.

Table 2-13. Parameters

Data direction	Parameter name	Description
[in]	addr	Disk address (unit 512B)
[out]	ram	Pointer to store the data

Returns
Status of the LUN.

2.4.3.8. Function uhi_msc_mem_write_10_ram()

Writes 512 bytes to the current LUN.

```
Ctrl_status uhi_msc_mem_write_10_ram(
        uint32_t addr,
        const void * ram)
```

The LUN is selected by uhi_msc_mem_test_unit_ready() or uhi_msc_mem_read_capacity() function.

Table 2-14. Parameters

Data direction	Parameter name	Description
[in]	addr	Disk address (unit 512B)
[in]	ram	Pointer on the data

Returns
Status of the LUN.

2.5. Enumeration Definitions

2.5.1. Enum lun_status_t

Status of LUN.

Table 2-15. Members

Enum value	Description
LUN_GOOD	Success, memory ready
LUN_FAIL	An error occurred
LUN_NOT_PRESENT	Memory unplugged
LUN_BUSY	Memory not initialized or changed

3. Quick Start Guide for USB Host Mass-Storage Module (UHI MSC)

This is the quick start guide for the USB Host Mass-Storage Module (UHI MSC) 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 Host MSC (Single Class support)" module is used.

The "USB Host MSC (Multiple Classes support)" module usage is described in Advanced Use Cases.

This example do a simple physical memory access, but a file system module can be added to decode the USB memory file system, see FatFS examples.

3.1.1. Setup Steps

As a USB host, it follows common USB host setup steps. Refer to USB Host Basic Setup.

3.1.2. Usage Steps

3.1.2.1. Example Code

Content of conf_usb_host.h:

```
#define USB_HOST_UHI        UHI_MSC
#define UHI_MSC_CHANGE(dev, b_plug) my_callback_msc_change(dev, b_plug)
extern bool my_callback_msc_change(uhc_device_t* dev, bool b_plug);
#include "uhi_msc_mem.h" // At the end of conf_usb_host.h file
```

Add to application C-file:

```
static bool my_flag_autorize_msc_check = false;
bool my_callback_msc_change(uhc_device_t* dev, bool b_plug)
{
   if (b_plug) {
      my_flag_autorize_msc_check = true;
   } else {
      my_flag_autorize_msc_check = false;
   }
}

void my_task(void)
{
   if (!my_flag_autorize_msc_check) {
      return;
   }
   my_flag_autorize_msc_check = false;

   // Check all new USB disks plugged
   for (uint8_t lun=0; lun < uhi_msc_mem_get_lun(); lun++) {
      // Wait the end of USB disk install
      while (CTRL_BUSY == uhi_msc_mem_test_unit_ready(lun));
      if (CTRL_GOOD != uhi_msc_mem_test_unit_ready(lun)) {
         // Removal disk not present or fail
         continue;
```

```
        }
        // Read capacity
        uint32_t max_lba;
        uhi_msc_mem_read_capacity(lun, &max_lba);
    }
}
```

3.1.2.2. Workflow

1. Ensure that conf_usb_host.h is available and contains the following configuration, which is the USB host MSC configuration:

   ```
   #define USB_HOST_UHI   UHI_MSC
   ```

   Note: It defines the list of UHI supported by USB host.

   ```
   #define UHI_MSC_CHANGE(dev, b_plug) my_callback_msc_change(dev, b_plug)
   extern bool my_callback_msc_change(uhc_device_t* dev, bool b_plug);
   ```

   Note: This callback is called when a USB device MSC is plugged or unplugged.
2. The access of the USB memories is allowed through functions described in API Overview.

3.2. Advanced Use Cases

For more advanced use of the UHI MSC module, see the following use cases:

- Enable USB High Speed Support
- Multiple Classes Support
- Dual Roles Support

3.3. Enable USB High Speed Support

In this use case, the USB host is used to support USB high speed.

3.3.1. Setup Steps

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

3.3.2. Usage Steps

3.3.2.1. Example Code

Content of conf_usb_host.h:

```
#define USB_HOST_HS_SUPPORT
```

3.3.2.2. Workflow

1. Ensure that conf_usb_host.h is available and contains the following parameters required for a USB device high speed (480Mbit/s):

   ```
   #define  USB_HOST_HS_SUPPORT
   ```

3.4. Multiple Classes Support

In this use case, the USB host is used to support several USB classes.

3.4.1. Setup Steps

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

3.4.2. Usage Steps

3.4.2.1. Example Code

Content of conf_usb_host.h:

```
#define USB_HOST_UHI   UHI_HID_MOUSE, UHI_MSC, UHI_CDC
```

3.4.2.2. Workflow

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

   ```
   #define USB_HOST_UHI   UHI_HID_MOUSE, UHI_MSC, UHI_CDC
   ```

   Note: USB_HOST_UHI defines the list of UHI supported by USB host. Here, you must add all classes that you want to support.

3.5. Dual Roles Support

In this use case, the USB host and USB device are enabled, it is the dual role.

Note: On the Atmel boards, the switch of USB role is managed automatically by the USB stack thank to a USB On-The-Go (OTG) connector and its USB ID pin. Refer to section "Dual roles" for further information in the application note:

- Atmel AVR4950: ASF - USB Host Stack

3.5.1. Setup Steps

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

3.5.2. Usage Steps

3.5.2.1. Example Code

Content of conf_usb_host.h:

```
#define UHC_MODE_CHANGE(b_host_mode)   my_callback_mode_change(b_host_mode)
extern void my_callback_mode_change(bool b_host_mode);
```

Add to application C-file:

```
void usb_init(void)
{
  //udc_start();
  uhc_start();
}

bool my_host_mode;
void my_callback_mode_change(bool b_host_mode)
{
  my_host_mode = b_host_mode;
}

void my_usb_task(void)
{
  if (my_host_mode) {
```

```
    // CALL USB Host task
  } else {
    // CALL USB Device task
  }
}
```

3.5.2.2. Workflow

1. In case of USB dual roles (Device and Host), the USB stack must be enabled by `uhc_start()` and the udc_start() must not be called.

```
//udc_start();
uhc_start();
```

2. In dual role, to know the current USB mode, the callback to notify the mode changes can be used.
   - Ensure that conf_usb_host.h contains the following parameters:

```
#define UHC_MODE_CHANGE(b_host_mode)
my_callback_mode_change(b_host_mode)
extern void my_callback_mode_change(bool b_host_mode);
```

   - Ensure that application contains the following code:

```
bool my_host_mode;
void my_callback_mode_change(bool b_host_mode)
{
  my_host_mode = b_host_mode;
}

void my_usb_task(void)
{
  if (my_host_mode) {
    // CALL USB Host task
  } else {
    // CALL USB Device task
  }
}
```

4. Configuration File Examples

4.1. conf_usb_host.h

4.1.1. UHI MSC Single

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

#ifndef _CONF_USB_HOST_H_
#define _CONF_USB_HOST_H_

#include "compiler.h"

#define USB_HOST_UHI         UHI_MSC

#define USB_HOST_POWER_MAX   500

// #define USB_HOST_HUB_SUPPORT

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

//#define UHC_MODE_CHANGE(b_host_mode)         usb_host_mode_change(b_host_mode)

//#define UHC_VBUS_CHANGE(b_present)           usb_host_vbus_change(b_present)

//#define UHC_VBUS_ERROR()                     usb_host_vbus_error()

//#define UHC_CONNECTION_EVENT(dev,b_present)
usb_host_connection_event(dev,b_present)

//#define UHC_WAKEUP_EVENT()                   usb_host_wakeup_event()

//#define UHC_SOF_EVENT()                      usb_host_sof_event()

//#define UHC_DEVICE_CONF(dev)                 uint8_t usb_host_device_conf(dev)

//#define UHC_ENUM_EVENT(dev,b_status)         usb_host_enum_event(dev,b_status)



#define UHI_MSC_CHANGE(dev,b_plug)



#include "uhi_msc.h"

#endif // _CONF_USB_HOST_H_
```

4.1.2. UHI MSC Multiple (Composite)

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

#ifndef _CONF_USB_HOST_H_
#define _CONF_USB_HOST_H_

#include "compiler.h"

#define USB_HOST_UHI         // UHI_MSC, UHI_HID_MOUSE, UHI_CDC, UHI_VENDOR

#define USB_HOST_POWER_MAX  500

// #define USB_HOST_HUB_SUPPORT

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

//#define UHC_MODE_CHANGE(b_host_mode)        usb_host_mode_change(b_host_mode)

//#define UHC_VBUS_CHANGE(b_present)          usb_host_vbus_change(b_present)

//#define UHC_VBUS_ERROR()                    usb_host_vbus_error()

//#define UHC_CONNECTION_EVENT(dev,b_present)
usb_host_connection_event(dev,b_present)

//#define UHC_WAKEUP_EVENT()                  usb_host_wakeup_event()

//#define UHC_SOF_EVENT()                     usb_host_sof_event()

//#define UHC_DEVICE_CONF(dev)                uint8_t usb_host_device_conf(dev)

//#define UHC_ENUM_EVENT(dev,b_status)        usb_host_enum_event(dev,b_status)




#define UHI_HID_MOUSE_CHANGE(dev,b_plug)
#define UHI_HID_MOUSE_EVENT_BTN_LEFT(b_state)
#define UHI_HID_MOUSE_EVENT_BTN_RIGHT(b_state)
#define UHI_HID_MOUSE_EVENT_BTN_MIDDLE(b_state)
#define UHI_HID_MOUSE_EVENT_MOUVE(x,y,scroll)


#define UHI_MSC_CHANGE(dev,b_plug)


#define UHI_CDC_CHANGE(dev,b_plug)
#define UHI_CDC_RX_NOTIFY()


#define UHI_VENDOR_CHANGE(dev, b_plug)
#define UHI_VENDOR_VID_PID_LIST {USB_VID_ATMEL,
USB_PID_ATMEL_ASF_VENDOR_CLASS}
```

```
//#include "uhi_msc.h"
//#include "uhi_hid_mouse.h"

#endif // _CONF_USB_HOST_H_
```

4.2. conf_clock.h

4.2.1. AT32UC3A0, AT32UC3A1, and AT32UC3B Devices (USBB)

```
/*
 * 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 Source Options
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_RCSYS
#define CONFIG_SYSCLK_SOURCE          SYSCLK_SRC_OSC0
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_PLL0

// ===== PLL0 Options
//#define CONFIG_PLL0_SOURCE          PLL_SRC_OSC0
//#define CONFIG_PLL0_SOURCE          PLL_SRC_OSC1
//#define CONFIG_PLL0_MUL             4 /* Fpll = (Fclk * PLL_mul) / PLL_div */
//#define CONFIG_PLL0_DIV             1 /* Fpll = (Fclk * PLL_mul) / PLL_div */

// ===== PLL1 Options
#define CONFIG_PLL1_SOURCE            PLL_SRC_OSC0
//#define CONFIG_PLL1_SOURCE          PLL_SRC_OSC1
#define CONFIG_PLL1_MUL               8 /* Fpll = (Fclk * PLL_mul) / PLL_div */
#define CONFIG_PLL1_DIV               2 /* Fpll = (Fclk * PLL_mul) / PLL_div */

// ===== System Clock Bus Division Options
//#define CONFIG_SYSCLK_CPU_DIV       0 /* Fcpu = Fsys/(2 ^ CPU_div) */
//#define CONFIG_SYSCLK_PBA_DIV       0 /* Fpba = Fsys/(2 ^ PBA_div) */
//#define CONFIG_SYSCLK_PBB_DIV       0 /* Fpbb = Fsys/(2 ^ PBB_div) */

// ===== Peripheral Clock Management Options
//#define CONFIG_SYSCLK_INIT_CPUMASK  ((1 << SYSCLK_SYSTIMER) | (1 << SYSCLK_OCD))
//#define CONFIG_SYSCLK_INIT_PBAMASK  (1 << SYSCLK_USART0)
//#define CONFIG_SYSCLK_INIT_PBBMASK  (1 << SYSCLK_HMATRIX)
//#define CONFIG_SYSCLK_INIT_HSBMASK  (1 << SYSCLK_MDMA_HSB)

// ===== USB Clock Source Options
//#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_OSC0
//#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_PLL0
#define CONFIG_USBCLK_SOURCE          USBCLK_SRC_PLL1
#define CONFIG_USBCLK_DIV             1 /* Fusb = Fsys/(2 ^ USB_div) */

#endif /* CONF_CLOCK_H_INCLUDED */
```

4.2.2. AT32UC3A3 and AT32UC3A4 Devices (USBB with High Speed Support)

```
/*
 * 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 Source Options
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_RCSYS
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_OSC0
#define CONFIG_SYSCLK_SOURCE          SYSCLK_SRC_PLL0

// ===== PLL0 Options
#define CONFIG_PLL0_SOURCE            PLL_SRC_OSC0
//#define CONFIG_PLL0_SOURCE          PLL_SRC_OSC1
#define CONFIG_PLL0_MUL               11 /* Fpll = (Fclk * PLL_mul) / PLL_div
*/
#define CONFIG_PLL0_DIV               2  /* Fpll = (Fclk * PLL_mul) / PLL_div
*/

// ===== PLL1 Options
//#define CONFIG_PLL1_SOURCE          PLL_SRC_OSC0
//#define CONFIG_PLL1_SOURCE          PLL_SRC_OSC1
//#define CONFIG_PLL1_MUL             8 /* Fpll = (Fclk * PLL_mul) / PLL_div */
//#define CONFIG_PLL1_DIV             2 /* Fpll = (Fclk * PLL_mul) / PLL_div */

// ===== System Clock Bus Division Options
#define CONFIG_SYSCLK_CPU_DIV         0 /* Fcpu = Fsys/(2 ^ CPU_div) */
#define CONFIG_SYSCLK_PBA_DIV         0 /* Fpba = Fsys/(2 ^ PBA_div) */
//#define CONFIG_SYSCLK_PBB_DIV       0 /* Fpbb = Fsys/(2 ^ PBB_div) */

// ===== Peripheral Clock Management Options
//#define CONFIG_SYSCLK_INIT_CPUMASK  ((1 << SYSCLK_SYSTIMER) | (1 << SYSCLK_OCD))
//#define CONFIG_SYSCLK_INIT_PBAMASK  (1 << SYSCLK_USART0)
//#define CONFIG_SYSCLK_INIT_PBBMASK  (1 << SYSCLK_HMATRIX)
//#define CONFIG_SYSCLK_INIT_HSBMASK  (1 << SYSCLK_MDMA_HSB)

// ===== USB Clock Source Options
#define CONFIG_USBCLK_SOURCE          USBCLK_SRC_OSC0
//#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_PLL0
//#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_PLL1
#define CONFIG_USBCLK_DIV             1 /* Fusb = Fsys/(2 ^ USB_div) */

#endif /* CONF_CLOCK_H_INCLUDED */
```

4.2.3. AT32UC3C, ATUCXXD, ATUCXXL3U, and ATUCXXL4U Devices (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

// ===== System Clock Source Options
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_RCSYS
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_OSC0
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_OSC1
#define CONFIG_SYSCLK_SOURCE          SYSCLK_SRC_PLL0
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_PLL1
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_RC8M

// ===== PLL0 Options
```

```
#define CONFIG_PLL0_SOURCE              PLL_SRC_OSC0
//#define CONFIG_PLL0_SOURCE          PLL_SRC_OSC1
//#define CONFIG_PLL0_SOURCE          PLL_SRC_RC8M
#define CONFIG_PLL0_MUL                 3 /* Fpll = (Fclk * PLL_mul) / PLL_div */
#define CONFIG_PLL0_DIV                 1 /* Fpll = (Fclk * PLL_mul) / PLL_div */

// ===== PLL1 Options
//#define CONFIG_PLL1_SOURCE          PLL_SRC_OSC0
//#define CONFIG_PLL1_SOURCE          PLL_SRC_OSC1
//#define CONFIG_PLL1_SOURCE          PLL_SRC_RC8M
//#define CONFIG_PLL1_MUL             3 /* Fpll = (Fclk * PLL_mul) / PLL_div */
//#define CONFIG_PLL1_DIV             1 /* Fpll = (Fclk * PLL_mul) / PLL_div */

// ===== System Clock Bus Division Options
//#define CONFIG_SYSCLK_CPU_DIV       0 /* Fcpu = Fsys/(2 ^ CPU_div) */
//#define CONFIG_SYSCLK_PBA_DIV       0 /* Fpba = Fsys/(2 ^ PBA_div) */
//#define CONFIG_SYSCLK_PBB_DIV       0 /* Fpbb = Fsys/(2 ^ PBB_div) */
//#define CONFIG_SYSCLK_PBC_DIV       0 /* Fpbc = Fsys/(2 ^ PBC_div) */

// ===== Peripheral Clock Management Options
//#define CONFIG_SYSCLK_INIT_CPUMASK  ((1 << SYSCLK_SYSTIMER) | (1 << SYSCLK_OCD))
//#define CONFIG_SYSCLK_INIT_PBAMASK  (1 << SYSCLK_USART0)
//#define CONFIG_SYSCLK_INIT_PBBMASK  (1 << SYSCLK_HMATRIX)
//#define CONFIG_SYSCLK_INIT_HSBMASK  (1 << SYSCLK_MDMA_HSB)

// ===== USB Clock Source Options
//#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_OSC0
//#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_OSC1
#define CONFIG_USBCLK_SOURCE            USBCLK_SRC_PLL0
//#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_PLL1
#define CONFIG_USBCLK_DIV               1 /* Fusb = Fsys/(2 ^ USB_div) */

#endif /* CONF_CLOCK_H_INCLUDED */
```

4.2.4. 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 devider: 1 (not devided)
- UPLL frequency: 480MHz
- USB clock: 480 / 1 = 480MHz
*/


#endif /* CONF_CLOCK_H_INCLUDED */
```

4.3. conf_clocks.h

4.3.1. SAM D21 Devices (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         true
```

```
/* 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                 true
#  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              false
#  define CONF_CLOCK_XOSC32K_RUN_IN_STANDBY         true

/* 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      false
#  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_CLOSED
#  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 / 8)
#  define CONF_CLOCK_DFLL_MAX_FINE_STEP_SIZE        (0xff / 8)

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

#  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                true
#  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. AT32UC3A0, AT32UC3A1, and AT32UC3B Devices (USBB)

```
/*
 * 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

// Only the default board init (switchs/leds) is necessary for this example

#endif /* CONF_BOARD_H_INCLUDED */
```

4.4.2. AT32UC3A3 and AT32UC3A4 Devices (USBB with High Speed Support)

```
/*
 * 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

// Only the default board init (switchs/leds) is necessary for this example

#endif /* CONF_BOARD_H_INCLUDED */
```

4.4.3. AT32UC3C, ATUCXXD, ATUCXXL3U, and ATUCXXL4U Devices (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

// Only the default board init (switchs/leds) is necessary for this example

#endif /* CONF_BOARD_H_INCLUDED */
```

4.4.4. 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

/* Pins description corresponding to Rxd,Txd, (UART pins) */
//#define CONSOLE_PINS        {PINS_UART}

/* Usart Hw ID used by the console (UART0) */
//#define CONSOLE_UART_ID          ID_UART

/* Configure UART pins */
//#define CONF_BOARD_UART_CONSOLE

/* Configure ADC example pins */
//#define CONF_BOARD_ADC

/* Configure PWM LED0 pin */
//#define CONF_BOARD_PWM_LED0

/* Configure PWM LED1 pin */
//#define CONF_BOARD_PWM_LED1

/* Configure PWM LED2 pin */
//#define CONF_BOARD_PWM_LED2

/* Configure SPI0 pins */
//#define CONF_BOARD_SPI0
//#define CONF_BOARD_SPI0_NPCS0
//#define CONF_BOARD_SPI0_NPCS1
//#define CONF_BOARD_SPI0_NPCS2
//#define CONF_BOARD_SPI0_NPCS3

/* Configure SPI1 pins */
//#define CONF_BOARD_SPI1
//#define CONF_BOARD_SPI1_NPCS0
//#define CONF_BOARD_SPI1_NPCS1
//#define CONF_BOARD_SPI1_NPCS2
//#define CONF_BOARD_SPI1_NPCS3

//#define CONF_BOARD_TWI0

//#define CONF_BOARD_TWI1

/* Configure USART RXD pin */
//#define CONF_BOARD_USART_RXD

/* Configure USART TXD pin */
//#define CONF_BOARD_USART_TXD
```

```
/* Configure USART CTS pin */
//#define CONF_BOARD_USART_CTS

/* Configure USART RTS pin */
//#define CONF_BOARD_USART_RTS

/* Configure USART synchronous communication SCK pin */
//#define CONF_BOARD_USART_SCK

/* Configure ADM3312 enable pin */
//#define CONF_BOARD_ADM3312_EN

/* Configure IrDA transceiver shutdown pin */
//#define CONF_BOARD_TFDU4300_SD

/* Configure RS485 transceiver ADM3485 RE pin */
//#define CONF_BOARD_ADM3485_RE

//#define CONF_BOARD_SMC_PSRAM

/* Configure LCD EBI pins */
//#define CONF_BOARD_HX8347A

/* Configure Backlight control pin */
//#define CONF_BOARD_AAT3194

/* Configure USB pins */
#define CONF_BOARD_USB_PORT

#endif /* CONF_BOARD_H_INCLUDED */
```

4.4.5. SAM D21 Devices (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
/* ID detect enabled */
#define CONF_BOARD_USB_ID_DETECT

#endif /* CONF_BOARD_H_INCLUDED */
```

5. USB Host Basic Setup

5.1. USB Host User Configuration

The following USB host configuration must be included in the conf_usb_host.h file of the application:

1. USB_HOST_UHI (List of UHI APIs).

Define the list of UHI supported by USB host. (E.g.: UHI_MSC, UHI_HID_MOUSE).

2. USB_HOST_POWER_MAX (mA).

Maximum current allowed on Vbus.

3. USB_HOST_HS_SUPPORT (Only defined).

Authorize the USB host to run in High Speed.

4. USB_HOST_HUB_SUPPORT (Only defined).

Authorize the USB HUB support.

5.2. USB Host User Callback

The following optional USB host callback can be defined in the conf_usb_host.h file of the application:

1. void UHC_MODE_CHANGE (bool b_host_mode).

To notify that the USB mode are switched automatically. This is possible only when ID pin is available.

2. void UHC_VBUS_CHANGE (bool b_present).

To notify that the Vbus level has changed (Available only in USB hardware with Vbus monitoring).

3. void UHC_VBUS_ERROR (void).

To notify that a Vbus error has occurred (Available only in USB hardware with Vbus monitoring).

4. void UHC_CONNECTION_EVENT (uhc_device_t* dev, bool b_present).

To notify that a device has been connected or disconnected.

5. void UHC_WAKEUP_EVENT (void).

Called when a USB device or the host have wake up the USB line.

6. void UHC_SOF_EVENT (void).

Called for each received SOF each 1ms. Available in High and Full speed mode.

7. uint8_t UHC_DEVICE_CONF (uhc_device_t* dev).

Called when a USB device configuration must be chosen. Thus, the application can choose either a configuration number for this device or a configuration number 0 to reject it. If callback not defined the configuration 1 is chosen.

8. void UHC_ENUM_EVENT (uhc_device_t* dev, uint8_t b_status).

Called when a USB device enumeration is completed or failed.

5.3. USB Host Setup Steps

5.3.1. USB Host Controller (UHC) - Prerequisites

Common prerequisites for all USB hosts.

This module is based on USB host 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 setup the project correctly:

- Specify the clock configuration:
  - 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 must use a DFLL and an external OSC.
- 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 Host Controller (UHC) - Example Code

Common example code for all USB hosts.

Content of conf_usb_host.h:

```
#define USB_HOST_POWER_MAX  500
```

Add to application C-file:

```
void usb_init(void)
{
    uhc_start();
}
```

5.3.3. USB Device Controller (UHC) - Workflow

Common workflow for all USB devices.

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

```
// Maximum current allowed on Vbus (mA) which depends of 5V generator
#define USB_HOST_POWER_MAX  500 // (500mA)
```

2. Call the USB host stack start function to enable USB Host stack:

```
uhc_start();
```

5.4. conf_clock.h Examples

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 device (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 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):

```
// USB Clock Source fixed at DFLL.
// SYSTEM_CLOCK_SOURCE_XOSC32K configuration - External 32KHz crystal/clock
oscillator
#  define CONF_CLOCK_XOSC32K_ENABLE                true
```

```
#  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            false
#  define CONF_CLOCK_XOSC32K_RUN_IN_STANDBY       true
// 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_CLOSED
#  define CONF_CLOCK_DFLL_ON_DEMAND               true

// 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 / 8)
#  define CONF_CLOCK_DFLL_MAX_FINE_STEP_SIZE      (0xff / 8)

#  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                true
#  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         true
```

6. Document Revision History

Doc. Rev.	Date	Comments
42344B	12/2015	Fixed typos
42344A	12/2014	Initial release

Atmel | Enabling Unlimited Possibilities®

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