Atmel マイクロ コントローラ

AT09333：通信クラス装置(CDC)用USBホスト インターフェース(UHI)

応用記述

序説

通信クラス装置(CDC)用USBホスト インターフェース(UHI)はUSB CDC直列ホストの形態設定と管理に関するインターフェースを提供します。

この資料の概要は以下の通りです。

- API概要
- USBホスト通信クラス装置単位部(UHI CDC)用の即時開始の手引き
- 形態設定ファイル例

Atmel®ソフトウェア枠組み(ASF) USBホスト階層に関するより多くの詳細については以下の応用記述を参照してください。

- AVR4950：ASF – USBホスト階層

目次

1. ソフトウェア許諾契約

変更の有りまたはなしでソースと2進の形式での使用と再配布は以下の条件に合っていれば許されます。

1. ソース コードの再配布は上の著作権通知、この条件一覧、それと以下のお断りを維持しなければなりません。
2. 2進形式での再配布はこの配布で提供された資料や他の素材で、上の著作権通知、この条件一覧、それと以下のお断りを再現しなければなりません。
3. Atmelの名称は先に書かれた許諾の指定なしにこのソフトウェアから派生した販促製品や裏書(保証)に使用できないかもしれません。
4. このソフトウェアはAtmelのマイクロ コントローラ製品との関係でだけ再分配して使用することができます。

このソフトウェアは特定目的のための市場性と適合性の暗黙的な保証が明白且つ明確に放棄されることを含みますが、これに限らず、何等かの明示的または暗示的な保証と"現状そのまま"でAtmelによって提供されます。例えそのような損害賠償の可能性を通知されたとしても、このソフトウェアの使用の外の何処でも生じる契約、厳密な責任、(不注意やその他を含む)不法行為かどうかに拘わらず、発生する(代替物またはサービスの獲得、使用、データ、または利益の損失、または事業中断を含みますが、それに限らず)、如何なる直接的、間接的、偶発的、特別的、典型的、または間接的な損害に関して決してAtmelに責任がないでしょう。

2. API概要

2.1. マクロ定義

2.1.1. USBホスト コア(UHC)とのインターフェース

UHCによって必要とされる定義と関数

2.1.1.1. UHI_CDC マクロ

```
#define UHI_CDC
```

UHC用の標準UHI APIを含む全域定義。conf_usb_host.hファイルからUSB_HOST_UHI定義で追加されなければなりません。

2.2. 関数定義

2.2.1. UHCによって必要とされる関数

2.2.1.1. uhi_cdc_install()関数

インターフェースを取り付け

```
uhc_enum_status_t uhi_cdc_install( uhc_device_t * dev )
```

支援されるなら、インターフェース エンドポイントを割り当て

表2-1. パラメータ

パラメータ名	データ方向	説明
dev	[入力]	要求する装置

戻り値：取り付けの状態

2.2.1.2. uhi_cdc_enable()関数

インターフェースを許可

```
void uhi_cdc_enable( uhc_device_t * dev )
```

UHIに対応するUSBインターフェースを許可

表2-2. パラメータ

パラメータ名	データ方向	説明
dev	[入力]	要求する装置

2.2.1.3. uhi_cdc_uninstall()関数

(取り付けられていれば)インターフェースを取り外し

```
void uhi_cdc_uninstall( uhc_device_t * dev )
```

表2-3. パラメータ

パラメータ名	データ方向	説明
dev	[入力]	要求する装置

2.2.1.4. uhi_cdc_sof()関数

SOFが発生した合図

```
void uhi_cdc_sof( bool b_micro )
```

2.2.2. 通信装置クラス用UHI

このUSBホスト クラスを使用するために高位応用によって使用される共通的なAPI。これらのルーチンはUSB CDCインターフェースとそのデータを転送するのにメモリによって使用されます。

2.2.2.1. uhi_cdc_open()関数

UHI CDCインターフェースのポートを開きます。

```
bool uhi_cdc_open( uint8_t port, usb_cdc_line_coding_t * configuration )
```

表2-4. パラメータ

パラメータ名	データ方向	説明
port	[入力]	通信ポート番号
configuration	[入力]	ポート形態設定上のポインタ

戻り値：ポートが利用可能ならばtrue

2.2.2.2. uhi_cdc_close()関数

ポートを閉じます。

```
void uhi_cdc_close( uint8_t port )
```

表2-5. パラメータ

パラメータ名	データ方向	説明
port	[入力]	通信ポート番号

2.2.2.3. uhi_cdc_is_rx_ready()関数

この関数はCDC線で文字が受信されたかを調べます。

```
bool uhi_cdc_is_rx_ready( uint8_t port )
```

表2-6. パラメータ

パラメータ名	データ方向	説明
port	[入力]	通信ポート番号

戻り値：読むべきバイトが準備可ならばtrue

2.2.2.4. uhi_cdc_get_nb_received()関数

この関数はCDC線で利用可能な文字数を返します。

```
iram_size_t uhi_cdc_get_nb_received( uint8_t port )
```

表2-7. パラメータ

パラメータ名	データ方向	説明
port	[入力]	通信ポート番号

戻り値：受信したデータ数

2.2.2.5. uhi_cdc_getc()関数

待機してCDC線上の値を取得

```
int uhi_cdc_getc( uint8_t port )
```

表2-8. パラメータ

パラメータ名	データ方向	説明
port	[入力]	通信ポート番号

戻り値：CDC線で読んだ値

2.2.2.6. uhi_cdc_read_buf()関数

CDC線のRAM緩衝部を読みます。

```
iram_size_t uhi_cdc_read_buf( uint8_t port, void * buf, iram_size_t size )
```

表2-9. パラメータ

パラメータ名	データ方向	説明
port	[入力]	通信ポート番号
buf	[出力]	読んだ値
size	[入力]	読んだ値の数

戻り値：残りデータ数

2.2.2.7. uhi_cdc_is_tx_ready()関数

この関数は新しい文字の送出が可能かを調べます。

```
bool uhi_cdc_is_tx_ready( uint8_t port )
```

コンパイラLIBからのscanf再指定を支援するためにint型が使用されます。

表2-10. パラメータ

パラメータ名	データ方向	説明
port	[入力]	通信ポート番号

戻り値：新しい文字を送れる場合にtrue

2.2.2.8. uhi_cdc_putc()関数

CDC線にﾊﾞｲﾄを出力。

```
int uhi_cdc_putc( uint8_t port, int value )
```

ｺﾝﾊﾟｲﾗLIBからのprintf再指定を支援するためにint型が使用されます。

表2-11. ﾊﾟﾗﾒｰﾀ

ﾊﾟﾗﾒｰﾀ名	ﾃﾞｰﾀ方向	説明
port	[入力]	管理する通信ﾎﾟｰﾄ番号
value	[入力]	送出する値

戻り値：関数が成功裏に終了した場合にtrue、さもなければfalse

2.2.2.9. uhi_cdc_write_buf()関数

CDC線のRAM緩衝部に書きます。

```
iram_size_t uhi_cdc_write_buf( uint8_t port, const void * buf, iram_size_t size )
```

表2-12. ﾊﾟﾗﾒｰﾀ

ﾊﾟﾗﾒｰﾀ名	ﾃﾞｰﾀ方向	説明
port	[入力]	通信ﾎﾟｰﾄ番号
buf	[入力]	書く値
size	[入力]	書く値の数

戻り値：残りﾃﾞｰﾀ数

3. USBﾎｽﾄ通信装置ｸﾗｽ単位部(UHI CDC)用の即時開始の手引き

これは使用事例の選択に於いて単位部をどう形態設定して使用するかを段階的に指示する、USBﾎｽﾄ通信装置ｸﾗｽ単位部(UHI CDC)用の即時開始の手引きです。

使用事例は様々なｺｰﾄﾞの断片を強調します。構成設定に関する段階でのｺｰﾄﾞの断片は独自初期化関数内に複写することができ、一方使用に関する段階は例えば主応用関数内に複写することができます。

3.1. 基本的な使用事例

この基本的な使用事例では”USB Host CDC (Single Class support)”単位部が使用されます。”USB Host CDC (Multiple Classes support)”単位部の使い方は高度な使用事例で記述されます。

3.1.1. 構成設定段階

USBﾎｽﾄのため、共通USBﾎｽﾄ構成設定段階に従います。USBﾎｽﾄ基本構成設定を参照してください。

3.1.2. 使用段階

3.1.2.1. ｺｰﾄﾞ例

conf_usb_host.hの内容

```
#define USB_HOST_UHI        UHI_CDC
#define UHI_CDC_CHANGE(dev, b_plug) my_callback_cdc_change(dev, b_plug)
extern bool my_callback_cdc_change(uhc_device_t* dev, bool b_plug);
#define UHI_CDC_RX_NOTIFY() my_callback_cdc_rx_notify()
extern void my_callback_cdc_rx_notify(void);
#include "uhi_cdc.h" // conf_usb_host.hﾌｧｲﾙの最後で
```

応用Cﾌｧｲﾙに追加してください。

```
static bool my_flag_cdc_available = false;
bool my_callback_cdc_change(uhc_device_t* dev, bool b_plug)
{
   if (b_plug) {

   // USB装置CDC接続
   my_flag_cdc_available = true;
   // USB CDCﾎﾟｰﾄを開いて形態設定
   usb_cdc_line_coding_t cfg = {
         .dwDTERate   = CPU_TO_LE32(115200),
```

```
            .bCharFormat = CDC_STOP_BITS_1,
            .bParityType = CDC_PAR_NONE,
            .bDataBits   = 8,
    };
    uhi_cdc_open(0, &cfg);

    } else {

    my_flag_cdc_available = false;

    }
}

void my_callback_cdc_rx_notify(void)
{
    // my_task_rx()タスク起動
}
#define MESSAGE "Hello"
void my_task(void)
{
    static bool startup = true;

    if (!my_flag_cdc_available) {
        startup = true;
        return;
    }

    if (startup) {
        startup = false;
        // CDC通信ポートにデータ送出
        uhi_cdc_write_buf(0, MESSAGE, sizeof(MESSAGE)-1);
        uhi_cdc_putc(0,'¥n');
        return;
    }
}

void my_task_rx(void)
{
    while (uhi_cdc_is_rx_ready(0)) {
        int value = uhi_cdc_getc(0);
    }
}
```

3.1.2.2. 作業の流れ

1. conf_usb_host.hが利用可能でUSBホストCDC形態設定である以下の形態設定を含むことを確実にしてください。

```
#define USB_HOST_UHI   UHI_CDC
```

注：これはUSBホストによって支援されるUHIの一覧を定義します。

```
#define UHI_CDC_CHANGE(dev, b_plug) my_callback_cdc_change(dev, b_plug)
extern bool my_callback_cdc_change(uhc_device_t* dev, bool b_plug);
```

注：この呼び戻しはUSB装置CDCが接続または切断される時に呼ばれます。通信ポートはここで開かれて形態設定されます。

```
#define UHI_CDC_RX_NOTIFY() my_callback_cdc_rx_notify()
extern void my_callback_cdc_rx_notify(void);
```

注：この呼び戻しは新しいデータが受信された時に呼ばれます。これは割り込みを通してデータ受信を管理して滞留を避けるのに使用することができます。

2. CDCデータ アクセス関数はUHI CDC API概要で記述されます。

3.2. 高度な使用事例

UHI CDC単位部のもっと高度な使用については以下の使用事例をご覧ください。

- USB高速(HS)支援許可
- 複数ｸﾗｽ支援
- 二重役割支援

3.3. USB高速(HS)支援許可

この使用事例では、USB高速(HS)を支援するのにUSBﾎｽﾄが使用されます。

3.3.1. 構成設定段階

この事例を実装するのに先立って、既にUHI単位部”基本的な使用事例”が適用されていることを確実にしてください。

3.3.2. 使用段階

3.3.2.1. ｺｰﾄﾞ例

conf_usb_host.hの内容

```
#define USB_HOST_HS_SUPPORT
```

3.3.2.2. 作業の流れ

1. conf_usb_host.hが利用可能でUSB装置高速(480Mbps)に必要とされる以下のﾊﾟﾗﾒｰﾀを含むことを確実にしてください。

```
#define  USB_HOST_HS_SUPPORT
```

3.4. 複数ｸﾗｽ支援

この事例では、様々なUSBｸﾗｽを支援するのにUSBﾎｽﾄが使用されます。

3.4.1. 構成設定段階

この事例を実装するのに先立って、既にUHI単位部”基本的な使用事例”が適用されていることを確実にしてください。

3.4.2. 使用段階

3.4.2.1. ｺｰﾄﾞ例

conf_usb_host.hの内容

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

3.4.2.2. 作業の流れ

1. conf_usb_host.hが利用可能で以下のﾊﾟﾗﾒｰﾀを含むことを確実にしてください。

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

注：USB_HOST_UHIはUSBﾎｽﾄによって支援されるUHIの一覧を定義します。ここで、支援を望む全てのｸﾗｽを追加しなければなりません。

3.5. 二重役割支援

この事例では、USBﾎｽﾄとUSB装置が許可され、これは二重役割です。

注：Atmelの基板では、USBの役割がUSB On The Go(OTG)ｺﾈｸﾀとそれのUSB IDﾋﾟﾝにより、USB階層によって自動的に管理されます。更なる情報については以下の応用記述で「二重役割」項を参照してください。

- Atmel AVR4950:ASF – USBﾎｽﾄ階層

3.5.1. 構成設定段階

この事例を実装するのに先立って、既にUHI単位部”基本的な使用事例”が適用されていることを確実にしてください。

3.5.2. 使用段階

3.5.2.1. ｺｰﾄﾞ例

conf_usb_host.hの内容

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

応用Cファイルに以下を追加してください。

```
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) {
    // USBホスト作業呼び出し  } else {
    // USB装置作業呼び出し
  }
}
```

3.5.2.2. 作業の流れ

1. USB二重役割(ホストと装置)の事例では、uhc_start()によってUSB階層が許可されなければならず、udc_start()が呼ばれてはなりません。

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

2. 二重役割では、現在のUSB動作形態を知るために、動作形態変更を通知する呼び戻しを使用することができます。
   - conf_usb_host.hが以下のパラメータを含むことを確実にしてください。

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

   - 応用が以下のコードを含むことを確実にしてください。

```
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) {
    // USBホスト作業呼び出し
  } else {
    // USB装置作業呼び出し
  }
}
```

4. 形態設定ファイル例

4.1. conf_usb_host.h

4.1.1. 単一UHI CDC

```
/*
 * 支援とFAQ: <a href="http://www.atmel.com/design-support/">Atmel Support</a>を尋ねてください。
 */

#ifndef _CONF_USB_HOST_H_
#define _CONF_USB_HOST_H_

#include "compiler.h"

#define USB_HOST_UHI         UHI_CDC

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

#include "uhi_cdc.h"

#endif // _CONF_USB_HOST_H_
```

4.1.2. 複数UHI CDC (複合)

```
/*
 * 支援とFAQ: <a href="http://www.atmel.com/design-support/">Atmel Support</a>を尋ねてください。
 */

#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, AT32UC3Bﾃﾞﾊﾞｲｽ (USBB)

```
/*
 * 支援とFAQ: <a href="http://www.atmel.com/design-support/">Atmel Support</a>を尋ねてください。
 */

#ifndef CONF_CLOCK_H_INCLUDED
#define CONF_CLOCK_H_INCLUDED

/* ===== システム クロック元任意選択 */
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_RCSYS
#define CONFIG_SYSCLK_SOURCE          SYSCLK_SRC_OSC0
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_PLL0

/* ===== PLL0任意選択 */
//#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任意選択 */
#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 */

/* ===== システム クロック バス分周任意選択 */
//#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) */

/* ===== 周辺機能クロック管理任意選択 */
//#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クロック元任意選択 */
//#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, AT32UC3A4ﾃﾞﾊﾞｲｽ (高速(HS)支援付USBB)

```
/*
 * 支援とFAQ: <a href="http://www.atmel.com/design-support/">Atmel Support</a>を尋ねてください。
 */

#ifndef CONF_CLOCK_H_INCLUDED
#define CONF_CLOCK_H_INCLUDED

/* ===== システム クロック元任意選択 */
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_RCSYS
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_OSC0
#define CONFIG_SYSCLK_SOURCE          SYSCLK_SRC_PLL0

/* ===== PLL0任意選択 */
#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任意選択 */
//#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 */

/* ===== システム クロック バス分周任意選択 */
#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) */

/* ===== 周辺機能クロック管理任意選択 */
//#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クロック元任意選択 */
#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, ATUCXXL4Uﾃﾞﾊﾞｲｽ (USBC)

```
/*
 * 支援とFAQ: <a href="http://www.atmel.com/design-support/">Atmel Support</a>を尋ねてください。
 */

#ifndef CONF_CLOCK_H_INCLUDED
#define CONF_CLOCK_H_INCLUDED

// ===== システム クロック元任意選択
//#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任意選択
#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任意選択
//#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 */

// ===== システム クロック バス分周任意選択
//#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) */

// ===== 周辺機能クロック管理任意選択
//#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クロック元任意選択
//#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, SAM3Aﾃﾞﾊﾞｲｽ (UOTGHS:USB OTG 高速(HS))

```
/*
 * 支援とFAQ: <a href="http://www.atmel.com/design-support/">Atmel Support</a>を尋ねてください。
 */

#ifndef CONF_CLOCK_H_INCLUDED
#define CONF_CLOCK_H_INCLUDED

/* ===== ｼｽﾃﾑ ｸﾛｯｸ(MCK)元任意選択 */
//#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

/* ===== ｼｽﾃﾑ ｸﾛｯｸ(MCK)前置分周器任意選択    (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)任意選択    (Fpll = (Fclk * PLL_mul) / PLL_div)
   ここにmulとdivの実効値を使用してください。 */
#define CONFIG_PLL0_SOURCE          PLL_SRC_MAINCK_XTAL
#define CONFIG_PLL0_MUL             14
#define CONFIG_PLL0_DIV             1

/* ===== 480MHzで固定化されたUPLL (UTMI)ﾊｰﾄﾞｳｪｱ */

/* ===== USBｸﾛｯｸ元任意選択    (Fusb = FpllX / USB_div)
   ここにdivの実効値を使用してください。 */
//#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_PLL0
#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_UPLL
#define CONFIG_USBCLK_DIV           1

/*
===== 目的対象周波数 (ｼｽﾃﾑ ｸﾛｯｸ)
- XTAL周波数: 12MHz
- ｼｽﾃﾑ ｸﾛｯｸ元: PLLA
- ｼｽﾃﾑ ｸﾛｯｸ前置分周器: 2 (2分周)
- PLLA供給元: XTAL
- PLLA出力: XTAL * 14 / 1
- ｼｽﾃﾑ ｸﾛｯｸは: 12 * 14 / 1 /2 = 84MHz
===== 目的対象周波数 (USBｸﾛｯｸ)
- USBｸﾛｯｸ元: UPLL
- USBｸﾛｯｸ分周器: 1 (分周なし)
- UPLL周波数: 480MHz
- USBｸﾛｯｸ: 480 / 1 = 480MHz
*/

#endif /* CONF_CLOCK_H_INCLUDED */
```

4.3. conf_clocks.h

4.3.1. SAM D21ﾃﾞﾊﾞｲｽ (USB)

```
/*
 * 支援とFAQ: <a href="http://www.atmel.com/design-support/">Atmel Support</a>を尋ねてください。
 */

#include <clock.h>

#ifndef CONF_CLOCKS_H_INCLUDED
#  define CONF_CLOCKS_H_INCLUDED

/* ｼｽﾃﾑ ｸﾛｯｸ ﾊﾞｽ形態設定 */
#  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形態設定 - 内部8MHz発振器 */
#  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形態設定 - 外部ｸﾛｯｸ/発振器 */
#  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形態設定 - 外部32kHzｸﾘｽﾀﾙ/ｸﾛｯｸ発振器 */
#  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形態設定 - 内部32kHz発振器 */
#  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形態設定 - ﾃﾞｼﾞﾀﾙ周波数固定化閉路 */
#  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開路動作形態設定 */
#  define CONF_CLOCK_DFLL_FINE_VALUE               (512)

/* DFLL閉路動作形態設定 */
```

```
#  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形態設定 - ﾃﾞｼﾞﾀﾙ位相固定化閉路 */
#  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基準形態設定 */
# define CONF_CLOCK_DPLL_REFERENCE_GCLK_GENERATOR GCLK_GENERATOR_1
/* DPLL GCLK固定化計時器形態設定 */
# define CONF_CLOCK_DPLL_LOCK_GCLK_GENERATOR GCLK_GENERATOR_1

/* clocks_init走行時にGCLKを形態設定するためにこれをtrueに設定してください。
 * falseに設定した場合、GCLK発振器はclocks_init()で何も形態設定されません。*/
#  define CONF_CLOCK_CONFIGURE_GCLK               true

/* GCLK発振器0形態設定 (主ｸﾛｯｸ) */
#  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

/* GCLK発振器1形態設定 */
#  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

/* GCLK発振器2形態設定 (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

/* GCLK発振器3形態設定 */
#  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

/* GCLK発振器4形態設定 */
#  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

/* GCLK発振器5形態設定 */
#  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

/* GCLK発振器6形態設定 */
#  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

/* GCLK発振器7形態設定 */
#  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

/* GCLK発振器8形態設定 */
#  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, AT32UC3Bﾃﾞﾊﾞｲｽ (USBB)

```
/*
 * 支援とFAQ: <a href="http://www.atmel.com/design-support/">Atmel Support</a>を尋ねてください。
 */

#ifndef CONF_BOARD_H_INCLUDED
#define CONF_BOARD_H_INCLUDED

/* UARTﾎﾟｰﾄ許可 */
#define CONF_BOARD_COM_PORT

#endif /* CONF_BOARD_H_INCLUDED */
```

4.4.2. AT32UC3A3, AT32UC3A4ﾃﾞﾊﾞｲｽ (高速(HS)支援付USBB)

```
/*
 * 支援とFAQ: <a href="http://www.atmel.com/design-support/">Atmel Support</a>を尋ねてください。
 */

#ifndef CONF_BOARD_H_INCLUDED
#define CONF_BOARD_H_INCLUDED

/* UARTﾎﾟｰﾄ許可 */
#define CONF_BOARD_COM_PORT

#endif /* CONF_BOARD_H_INCLUDED */
```

4.4.3. AT32UC3C, ATUCXXD, ATUCXXL3U, ATUCXXL4Uﾃﾞﾊﾞｲｽ (USBC)

```
/*
 * 支援とFAQ: <a href="http://www.atmel.com/design-support/">Atmel Support</a>を尋ねてください。
 */

#ifndef CONF_BOARD_H_INCLUDED
#define CONF_BOARD_H_INCLUDED

// UARTﾎﾟｰﾄ許可
#define CONF_BOARD_COM_PORT

#endif /* CONF_BOARD_H_INCLUDED */
```

4.4.4. SAM3X, SAM3Aﾃﾞﾊﾞｲｽ (UOTGHS:USB OTG 高速(HS))

```
/*
 * 支援とFAQ: <a href="http://www.atmel.com/design-support/">Atmel Support</a>を尋ねてください。
 */

#ifndef CONF_BOARD_H_INCLUDED
#define CONF_BOARD_H_INCLUDED

/* RXD,TXD(UARTﾋﾟﾝ)に対応するﾋﾟﾝの記述 */
#define CONSOLE_PINS          {PINS_UART}

/* ｺﾝｿｰﾙ(UART0)によって使用されるUSARTﾊｰﾄﾞｳｪｱID */
#define CONSOLE_UART_ID            ID_UART

/* UARTﾋﾟﾝ形態設定 */
#define CONF_BOARD_UART_CONSOLE

/* A/D変換器のﾋﾟﾝ形態設定 */
//#define CONF_BOARD_ADC

/* PWM LED0ﾋﾟﾝ形態設定 */
//#define CONF_BOARD_PWM_LED0

/* PWM LED1ﾋﾟﾝ形態設定 */
//#define CONF_BOARD_PWM_LED1

/* PWM LED2ﾋﾟﾝ形態設定 */
//#define CONF_BOARD_PWM_LED2

/* SPI0ﾋﾟﾝ形態設定 */
//#define CONF_BOARD_SPI0
//#define CONF_BOARD_SPI0_NPCS0
//#define CONF_BOARD_SPI0_NPCS1
//#define CONF_BOARD_SPI0_NPCS2
//#define CONF_BOARD_SPI0_NPCS3

/* SPI1ﾋﾟﾝ形態設定 */
//#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

/* USART RXDﾋﾟﾝ形態設定 */
```

```
//#define CONF_BOARD_USART_RXD

/* USART TXDﾋﾟﾝ形態設定 */
//#define CONF_BOARD_USART_TXD

/* USART CTSﾋﾟﾝ形態設定 */
//#define CONF_BOARD_USART_CTS

/* USART RTSﾋﾟﾝ形態設定 */
//#define CONF_BOARD_USART_RTS

/* USART同期通信SCKﾋﾟﾝ形態設定 */
//#define CONF_BOARD_USART_SCK

/* ADM3312許可ﾋﾟﾝ形態設定 */
//#define CONF_BOARD_ADM3312_EN

/* IrDA送受信部停止ﾋﾟﾝ形態設定 */
//#define CONF_BOARD_TFDU4300_SD

/* RS485送受信部ADM3485 REﾋﾟﾝ形態設定 */
//#define CONF_BOARD_ADM3485_RE

//#define CONF_BOARD_SMC_PSRAM

/* LCD EBIﾋﾟﾝ形態設定 */
//#define CONF_BOARD_HX8347A

/* 背面灯制御ﾋﾟﾝ形態設定 */
//#define CONF_BOARD_AAT3194

/* USBﾋﾟﾝ形態設定 */
#define CONF_BOARD_USB_PORT

#endif /* CONF_BOARD_H_INCLUDED */
```

4.4.5. SAM D21ﾃﾞﾊﾞｲｽ (USB)

```
/*
 * 支援とFAQ: <a href="http://www.atmel.com/design-support/">Atmel Support</a>を尋ねてください。
 */

 #ifndef CONF_BOARD_H_INCLUDED
 #define CONF_BOARD_H_INCLUDED

 /* USB VBUS検出許可 */
 #define CONF_BOARD_USB_VBUS_DETECT
 /* ID検出許可 */
 #define CONF_BOARD_USB_ID_DETECT

 #endif /* CONF_BOARD_H_INCLUDED */
```

5. USBホスト基本構成設定

5.1. USBホスト使用者形態設定

以下のUSBホスト形態設定は応用のconf_usb_host.hファイルでインクルードされなければなりません。

1. USB_HOST_UHI (UHI APIの一覧)
   USBホストによって支援されるUHIの一覧を定義 (例えば:UHI_MSC, UHI_HID_MOUSE)
2. USB_HOST_POWER_MAX (mA)
   VBUSで許される最大電流
3. USB_HOST_HS_SUPPORT (定義のみ)
   高速(HS)での走行をUSBホストに認可
4. USB_HOST_HUB_SUPPORT (定義のみ)
   USBハブ支援を認可

5.2. USBホスト使用者呼び戻し

以下のに任意選択USBホスト呼び戻しは応用のconf_usb_host.hファイルで定義されなければなりません。

1. void UHC_MODE_CHANGE (bool b_host_mode)
   USB動作形態が自動的に切り替わったことを通知。これはIDピンが利用可能な時にだけ可能です。
2. void UHC_VBUS_CHANGE (bool b_present)
   VBUSレベルが変化したことを通知 (VBUS監視を持つUSBハードウェアでだけ利用可能)
3. void UHC_VBUS_ERROR (void)
   VBUS異常が発生したことを通知 (VBUS監視を持つUSBハードウェアでだけ利用可能)
4. void UHC_CONNECTION_EVENT (uhc_device_t* dev, bool b_present)
   装置が接続または切断されたことを通知
5. void UHC_WAKEUP_EVENT (void)
   USB装置またはホストがUSB線で起動される時に呼ばれます。
6. void UHC_SOF_EVENT (void)
   1ms毎でSOFを受信する毎に呼ばれます。高速(HS)と全速(FS)の動作形態で利用可能
7. uint8_t UHC_DEVICE_CONF (uhc_device_t* dev)
   USB装置形態設定が選ばれなければならない時に呼ばれます。故に、応用はこの装置に対する形態設定番号か、またはそれを拒否するために形態設定番号0のどちらかを選ぶことができます。呼び戻しが定義されていなければ、形態設定1が選ばれます。
8. void UHC_ENUM_EVENT (uhc_device_t* dev, uint8_t b_status)
   USB装置列挙(接続認証)が完了されるか、または失敗した時に呼ばれます。

5.3. USBホスト構成設定段階

5.3.1. USBホスト制御部 (UHC) – 事前必要条件

全てのUSBホストに対する共通的な事前必要条件

この単位部は完全な割り込み駆動のUSBホスト階層に基づき、sleepmgrを支援します。AVR®とAtmel®のSMART ARM®に基づくSAM3/4デバイスについてはクロック サービスが支援されます。SAM D21デバイスについてはクロック駆動部が支援されます。

プロジェクトを正しく構成設定するために以下の手続きが実行されなければなりません。

- クロック形態設定を指定してください。
  - USB高速(HS)支援のないUC3とSAM3/4デバイスは48MHzクロック入力が必要です。
    PLLと外部発振器を使用しなければなりません。
  - USB高速(HS)支援付きのUC3とSAM3/4デバイスは12MHzクロック入力が必要です。
    外部発振器を使用しなければなりません。
  - USBCハードウェア付きのUC3デバイスは25MHzよりも高いCPU周波数が必要です。
  - USB高速(HS)支援のないSAM D21デバイスは48MHzクロック入力が必要です。
    DFLLと外部発振器を使用しなければなりません。
- conf_board.hに於いて、USB線を許可するためにCONF_BOARD_USB_PORT定義が追加されなければなりません。(全ての基板に対して必須ではありません。)
- 割り込みを許可してください。
- クロック サービスを初期化してください。

休止管理ｻｰﾋﾞｽの使用は任意選択ですが、消費電力を減らすために推奨されます。

・休止管理ｻｰﾋﾞｽを初期化してください。

・応用がｱｲﾄﾞﾙ状態の時に休止動作形態を活性にしてください。

conf_clock.h例

AVRとSAM3/4ﾃﾞﾊﾞｲｽについては以下の初期化ｺｰﾄﾞを追加してください。

```
sysclk_init();
irq_initialize_vectors();
cpu_irq_enable();
board_init();
sleepmgr_init(); // 任意選択
```

SAM D21ﾃﾞﾊﾞｲｽについては初期化ｺｰﾄﾞに以下を追加してください。

```
system_init();
irq_initialize_vectors();
cpu_irq_enable();
sleepmgr_init(); // 任意選択
```

主ｱｲﾄﾞﾙ繰り返しに以下を追加してください。

```
sleepmgr_enter_sleep(); // 任意選択
```

5.3.2. USBﾎｽﾄ制御部 (UHC) – ｺｰﾄﾞ例

全てのUSBﾎｽﾄに対する共通的なｺｰﾄﾞ例

conf_usb_host.hの内容

```
#define USB_HOST_POWER_MAX   500
```

応用Cﾌｧｲﾙに以下を追加してください。

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

5.3.3. USBﾎｽﾄ制御部 (UHC) – 作業の流れ

全てのUSBﾎｽﾄに対する共通的な作業の流れ

1. conf_usb_host.hが利用可能で主USB装置形態設定である以下の形態設定を含むことを確実にしてください。

```
// 5V生成部に依存してVBUSで許される最大電流(mA)
#define USB_HOST_POWER_MAX   500 // (500mA)
```

2. USBﾎｽﾄ階層を許可するためにUSBﾎｽﾄ階層開始関数を呼んでください。

```
uhc_start();
```

5.4. conf_clock.h例

AT32UC3A0,AT32UC3A1,AT32UC3Bﾃﾞﾊﾞｲｽ(USBB)用のconf_colck.hの内容

```
// 12MHz外部発振器に基づく形態設定
#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)
```

AT32UC3A3とAT32UC3A4ﾃﾞﾊﾞｲｽ(高速(HS)支援付きUSBB)用のconf_colck.hの内容

```
// 12MHz外部発振器に基づく形態設定
#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_OSC0
#define CONFIG_USBCLK_DIV           1 // Fusb = Fsys/(2 ^ USB_div)
```

AT32UC3Cﾃﾞﾊﾞｲｽ(USBC)用のconf_colck.hの内容

```
// 12MHz外部発振器に基づく形態設定
#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)
// USBCで動くために25MHz以上のCPUｸﾛｯｸが必要
#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_PLL1
```

SAM3XとSAM3Aﾃﾞﾊﾞｲｽ(UOTGHS:USB OTG 高速(HS))用のconf_colck.hの内容

```
// UPLLで固定化されるUSBｸﾛｯｸ元
#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_UPLL
#define CONFIG_USBCLK_DIV           1
```

SAM D21ﾃﾞﾊﾞｲｽ(USB)用のconf_colck.hの内容

```
// DFLLで固定化されるUSBｸﾛｯｸ元
// SYSTEM_CLOCK_SOURCE_XOSC32K形態設定 - 外部32kHzｸﾘｽﾀﾙ/ｸﾛｯｸ発振器
#  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形態設定 - ﾃﾞｼﾞﾀﾙ周波数固定化閉路
#  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閉路動作形態設定
#  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

// GCLK発振器0(主ｸﾛｯｸ)形態設定
#  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

// GCLK発振器1形態設定
#  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. 資料改訂履歴

資料改訂	日付	注釈
42338A	2014年12月	初版資料公開
42338B	2015年12月	誤植修正

Atmel Corporation 1600 Technology Drive, San Jose, CA 95110 USA TEL:(+1)(408) 441-0311 FAX: (+1)(408) 436-4200 | www.atmel.com

---

本応用記述はAtmelのAT09333応用記述(Rev.42338B-12/2015)の翻訳日本語版です。日本語では不自然となる重複する形容表現は省略されている場合があります。日本語では難解となる表現は大幅に意訳されている部分もあります。必要に応じて一部加筆されています。頁割の変更により、原本より頁数が少なくなっています。

必要と思われる部分には( )内に英語表記や略称などを残す形で表記しています。

青字の部分はﾘﾝｸとなっています。一般的に赤字の0,1は論理0,1を表します。その他の赤字は重要な部分を表します。