

序説

供給者クラス装置用USBホスト インターフェース(UHI)はUSB供給者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ホスト供給者単位部\(UHI供給者\)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装置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](#)

| | |
|------------------------------------------------|----|
| 序説 | 1 |
| 1. ソフトウェア許諾契約 | 3 |
| 2. API概要 | 4 |
| 2.1. マクロ定義 | 4 |
| 2.1.1. USBホストコア(UHC)とのインターフェース | 4 |
| 2.2. 関数定義 | 4 |
| 2.2.1. UHCによって必要とされる関数 | 4 |
| 2.2.2. 供給者クラス用UHI | 4 |
| 3. USBホスト供給者単位部(UHI供給者)用の即時開始の手引き | 7 |
| 3.1. 基本的な使用事例 | 7 |
| 3.1.1. 構成設定段階 | 7 |
| 3.1.2. 使用段階 | 7 |
| 3.2. 高度な使用事例 | 8 |
| 3.3. USB高速(HS)支援許可 | 8 |
| 3.3.1. 構成設定段階 | 8 |
| 3.3.2. 使用段階 | 8 |
| 3.4. 複数クラス支援 | 8 |
| 3.4.1. 構成設定段階 | 8 |
| 3.4.2. 使用段階 | 9 |
| 3.5. 二重役割支援 | 9 |
| 3.5.1. 構成設定段階 | 9 |
| 3.5.2. 使用段階 | 9 |
| 4. 形態設定ファイル例 | 10 |
| 4.1. conf_usb_host.h | 10 |
| 4.1.1. 単一UHI供給者 | 10 |
| 4.1.2. 複数UHI供給者(複合) | 11 |
| 4.2. conf_clock.h | 12 |
| 4.2.1. SAM3X, SAM3Aデバイス(UOTGHS:USB OTG 高速(HS)) | 12 |
| 4.2.2. SAM4Lデバイス(USBC) | 13 |
| 4.3. conf_clocks.h | 14 |
| 4.3.1. SAM D21デバイス(USB) | 14 |
| 4.4. conf_board.h | 16 |
| 4.4.1. SAM3X, SAM3Aデバイス(UOTGHS:USB OTG 高速(HS)) | 16 |
| 4.4.2. SAM4Lデバイス(USBC) | 18 |
| 4.4.3. SAM D21デバイス(USB) | 18 |
| 5. USBホスト基本構成設定 | 19 |
| 5.1. USBホスト使用者形態設定 | 19 |
| 5.2. USBホスト使用者呼び戻し | 19 |
| 5.3. USBホスト構成設定段階 | 19 |
| 5.3.1. USBホスト制御部(UHC) - 事前必要条件 | 19 |
| 5.3.2. USBホスト制御部(UHC) - コード例 | 20 |
| 5.3.3. USBホスト制御部(UHC) - 作業の流れ | 20 |
| 5.4. conf_clock.h例 | 20 |
| 6. 資料改訂履歴 | 21 |

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、これに限らず)、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_VENDOR マクロ

```
#define UHI_VENDOR
```

UHC用の標準UHI APIを含む全域定義。これは`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_vendor_install()関数

インターフェース取り付け

```
uhc_enum_status_t uhi_vendor_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_vendor_enable()関数

インターフェース許可

```
void uhi_vendor_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_vendor_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_vendor_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 |

2.2.2. 供給者クラス用UHI

USBホスト クラスを使用する上位応用によって使用される共通的なAPI。

この供給者クラス実装は制御IN、制御OUT、割り込みIN、割り込みOUT、大量(バルク)IN、大量OUT、等時(アイクロナス)IN、等時OUTの全方向の全エンドポイント形式に対して1つのエンドポイントを支援します。

この実装は一例で、2つの大量INエンドポイントのように多くのエンドポイントを支援する別の供給者クラスを作成するための基にすることができます。

2.2.2.1. uhi_vendor_control_in_run()関数

制御INで転送開始

```
bool uhi_vendor_control_in_run( uint8_t * buf, iram_size_t buf_size,  
    uhd_callback_setup_end_t callback)
```

転送が終了または失敗(中断停止、リセットなど)された時に呼び戻しが呼ばれます。この呼び戻しは転送状態と結果的に転送されたバイト数を返します。

表2-4. パラメータ

| パラメータ名 | データ方向 | 説明 |
|----------|-------|---------------------------------------------------------------------|
| buf | [出力] | 送るまたは満たすための内部RAM上の緩衝部。これは整列でなければならず、COMPILER_WORD_ALIGNEDを使用してください。 |
| buf_size | [入力] | 送るまたは満たすための緩衝部の大きさ |
| callback | [入力] | NULLまたは転送の最後で呼ぶ関数 |

戻り値：関数が成功裏に終了した場合に1、さもなければ0

2.2.2.2. uhi_vendor_control_out_run()関数

制御OUTで転送を開始

```
bool uhi_vendor_control_out_run( uint8_t * buf, iram_size_t buf_size,
    uhd_callback_setup_end_t callback )
```

転送が終了または失敗(中断停止、リセットなど)された時に呼び戻しが呼ばれます。この呼び戻しは転送状態と結果的に転送されたバイト数を返します。

表2-5. パラメータ

| パラメータ名 | データ方向 | 説明 |
|----------|-------|---------------------------------------------------------------------|
| buf | [入力] | 送るまたは満たすための内部RAM上の緩衝部。これは整列でなければならず、COMPILER_WORD_ALIGNEDを使用してください。 |
| buf_size | [入力] | 送るまたは満たすための緩衝部の大きさ |
| callback | [入力] | NULLまたは転送の最後で呼ぶ関数 |

戻り値：関数が成功裏に終了した場合に1、さもなければ0

2.2.2.3. uhi_vendor_bulk_in_run()関数

大量(バルク)INで転送開始

```
bool uhi_vendor_bulk_in_run( uint8_t * buf, iram_size_t buf_size, uhd_callback_trans_t callback )
```

転送が終了または失敗(中断停止、リセットなど)された時に呼び戻しが呼ばれます。この呼び戻しは転送状態と結果的に転送されたバイト数を返します。

表2-6. パラメータ

| パラメータ名 | データ方向 | 説明 |
|----------|-------|---------------------------------------------------------------------|
| buf | [出力] | 送るまたは満たすための内部RAM上の緩衝部。これは整列でなければならず、COMPILER_WORD_ALIGNEDを使用してください。 |
| buf_size | [入力] | 送るまたは満たすための緩衝部の大きさ |
| callback | [入力] | NULLまたは転送の最後で呼ぶ関数 |

戻り値：関数が成功裏に終了した場合に1、さもなければ0

2.2.2.4. uhi_vendor_bulk_out_run()関数

大量(バルク)OUTで転送を開始

```
bool uhi_vendor_bulk_out_run( uint8_t * buf, iram_size_t buf_size, uhd_callback_trans_t callback )
```

転送が終了または失敗(中断停止、リセットなど)された時に呼び戻しが呼ばれます。この呼び戻しは転送状態と結果的に転送されたバイト数を返します。

表2-7. パラメータ

| パラメータ名 | データ方向 | 説明 |
|----------|-------|---------------------------------------------------------------------|
| buf | [入力] | 送るまたは満たすための内部RAM上の緩衝部。これは整列でなければならず、COMPILER_WORD_ALIGNEDを使用してください。 |
| buf_size | [入力] | 送るまたは満たすための緩衝部の大きさ |
| callback | [入力] | NULLまたは転送の最後で呼ぶ関数 |

戻り値：関数が成功裏に終了した場合に1、さもなければ0

2.2.2.5. uhi_vendor_int_in_run()関数

割り込みINで転送開始

```
bool uhi_vendor_int_in_run( uint8_t * buf, iram_size_t buf_size, uhd_callback_trans_t callback )
```

転送が終了または失敗(中断停止、リセットなど)された時に呼び戻しが呼ばれます。この呼び戻しは転送状態と結果的に転送されたバイト数を返します。

表2-8. パラメータ

| パラメータ名 | データ方向 | 説明 |
|----------|-------|---------------------------------------------------------------------|
| buf | [出力] | 送るまたは満たすための内部RAM上の緩衝部。これは整列でなければならず、COMPILER_WORD_ALIGNEDを使用してください。 |
| buf_size | [入力] | 送るまたは満たすための緩衝部の大きさ |
| callback | [入力] | NULLまたは転送の最後で呼ぶ関数 |

戻り値：関数が成功裏に終了した場合に1、さもなければ0

2.2.2.6. uhi_vendor_int_out_run()関数

割り込みOUTで転送を開始

```
bool uhi_vendor_int_out_run( uint8_t * buf, iram_size_t buf_size, uhd_callback_trans_t callback )
```

転送が終了または失敗(中断停止、リセットなど)された時に呼び戻しが呼ばれます。この呼び戻しは転送状態と結果的に転送されたバイト数を返します。

表2-9. パラメータ

| パラメータ名 | データ方向 | 説明 |
|----------|-------|---------------------------------------------------------------------|
| buf | [入力] | 送るまたは満たすための内部RAM上の緩衝部。これは整列でなければならず、COMPILER_WORD_ALIGNEDを使用してください。 |
| buf_size | [入力] | 送るまたは満たすための緩衝部の大きさ |
| callback | [入力] | NULLまたは転送の最後で呼ぶ関数 |

戻り値：関数が成功裏に終了した場合に1、さもなければ0

2.2.2.7. uhi_vendor_iso_in_run()関数

等時(アイソクロナス)INで転送開始

```
bool uhi_vendor_iso_in_run( uint8_t * buf, iram_size_t buf_size, uhd_callback_trans_t callback )
```

転送が終了または失敗(中断停止、リセットなど)された時に呼び戻しが呼ばれます。この呼び戻しは転送状態と結果的に転送されたバイト数を返します。

表2-10. パラメータ

| パラメータ名 | データ方向 | 説明 |
|----------|-------|---------------------------------------------------------------------|
| buf | [出力] | 送るまたは満たすための内部RAM上の緩衝部。これは整列でなければならず、COMPILER_WORD_ALIGNEDを使用してください。 |
| buf_size | [入力] | 送るまたは満たすための緩衝部の大きさ |
| callback | [入力] | NULLまたは転送の最後で呼ぶ関数 |

戻り値：関数が成功裏に終了した場合に1、さもなければ0

2.2.2.8. uhi_vendor_iso_out_run()関数

等時(アイソクロナス)OUTで転送を開始

```
bool uhi_vendor_iso_out_run( uint8_t * buf, iram_size_t buf_size, uhd_callback_trans_t callback )
```

転送が終了または失敗(中断停止、リセットなど)された時に呼び戻しが呼ばれます。この呼び戻しは転送状態と結果的に転送されたバイト数を返します。

表2-11. パラメータ

| パラメータ名 | データ方向 | 説明 |
|----------|-------|---------------------------------------------------------------------|
| buf | [入力] | 送るまたは満たすための内部RAM上の緩衝部。これは整列でなければならず、COMPILER_WORD_ALIGNEDを使用してください。 |
| buf_size | [入力] | 送るまたは満たすための緩衝部の大きさ |
| callback | [入力] | NULLまたは転送の最後で呼ぶ関数 |

戻り値：関数が成功裏に終了した場合に1、さもなければ0

2.2.2.9. uhi_vendor_bulk_is_available()関数

大量(バルク)での転送が可能か調査

```
bool uhi_vendor_bulk_is_available( void )
```

戻り値：可能ならば1、さもなければ0

2.2.2.10. uhi_vendor_int_is_available()関数

割り込みでの転送が可能か調査

```
bool uhi_vendor_int_is_available( void )
```

戻り値：可能ならば1、さもなければ0

2.2.2.11. uhi_vendor_iso_is_available()関数

等時(アイソクロナス)での転送が可能か調査

```
bool uhi_vendor_int_is_available( void )
```

戻り値：可能ならば1、さもなければ0

3. USBホスト供給者単位部(UHI供給者)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ホスト供給者単位部\(UHI供給者\)用の即時開始の手引き](#)です。

使用事例は様々なコードの断片を強調します。構成設定に関する段階でのコード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 Vendor (Single Class support)”単位部が使用されます。

”USB Host Vendor (Composite)”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. コード例

`conf_usb_host.h`の内容

```
#define USB_HOST_UHI          UHI_VENDOR
#define UHI_VENDOR_CHANGE(dev, b_plug) my_callback_vendor_change(dev, b_plug)
extern void my_callback_vendor_change(uhc_device_t* dev, bool b_plug);
#define UHI_VENDOR_VID_PID_LIST {USB_VID_ATMEL, USB_PID_ATMEL_ASF_VENDOR_CLASS}
#include "uhi_vendor.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_vendor_test_start = false;
void my_callback_vendor_change(uhc_device_t* dev, bool b_plug)
{
    // USB装置供給者接続
    my_flag_vendor_test_start = b_plug;
}

static void my_callback_bulk_in_done (usb_add_t add,
    usb_ep_t ep, uhd_trans_status_t status, iram_size_t nb_transferred)
{
    if (status != UHD_TRANS_NOERROR) {
        return; // 転送中に異常
    }
    // データ受信後に検査再開
    my_flag_vendor_test_start = true;
}

#define MESSAGE "Hello bulk"
#define HELLO_SIZE 5
#define HELLO_BULK_SIZE 10
uint8_t my_out_buffer[MESSAGE_SIZE+1] = MESSAGE;
uint8_t my_in_buffer[MESSAGE_SIZE+1];
void my_task(void)
{
```



```

if (!my_flag_vendor_test_start) {
    return;
}
my_flag_vendor_test_start = false;

// 制御エンドポイントを通してデータ送出
uhi_vendor_control_out_run(my_out_buffer, HELLO_SIZE, NULL);

// 大量(バルク)エンドポイントが利用可能か調査
if (uhi_vendor_bulk_is_available()) {
    // 大量(バルク)INエンドポイントを通してデータ送出
    uhi_vendor_bulk_out_run(my_out_buffer, HELLO_BULK_SIZE, NULL);
    // 大量(バルク)INエンドポイントを通してデータ受信
    uhi_vendor_bulk_in_run(my_in_buffer, sizeof(my_in_buffer), my_callback_bulk_in_done);
}
}

```

3.1.2.2. 作業の流れ

1. `conf_usb_host.h`が利用可能でUSBホスト供給者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_HID_VENDOR
```

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_VENDOR_CHANGE(dev, b_plug) my_callback_vendor_change(dev, b_plug)
extern bool my_callback_vendor_change(uhc_device_t* dev, bool b_plug);
```

注：この呼び戻しはUSB装置供給者が接続または切断された時に呼ばれます。

```
#define UHI_VENDOR_VID_PID_LIST {USB_VID_ATMEL, USB_PID_ATMEL_ASF_VENDOR_CLASS}
```

注：これは(VIDとPIDによって定義され、)USBホストによって支援される装置の一覧を定義します。

2. 供給者データ転送関数は`uhi_vendor_group`で記述されます。

```
uhi_vendor_control_out_run(), uhi_vendor_bulk_out_run(), ~
```

3.2. 高度な使用事例

UHI供給者単位部のも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

この事例の実装に先だって、既に”基本的な使用事例”のUHI単位部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. コード例

`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(HS,480Mbps)に必要な以下のパ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

この事例の実装に先だって、既に”基本的な使用事例”のUHI単位部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. コード例

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パ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)コネクタとそのUSB IDピンにより、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

この事例の実装に先だって、既に”基本的な使用事例”のUHI単位部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. コード例

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ないが、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供給者

```
/*
 * 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_VENDOR

#define USB_HOST_POWER_MAX    500

// #define USB_HOST_HUB_SUPPORT

#if (UC3A3 || UC3A4)
# define USB_HOST_HS_SUPPORT
#elif (SAM3XA)
# 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_VENDOR_CHANGE(dev, b_plug)
#define UHI_VENDOR_VID_PID_LIST {USB_VID_ATMEL, USB_PID_ATMEL_ASF_VENDOR_CLASS}
```

```
#include "uhi_vendor.h"

#endif // _CONF_USB_HOST_H_
```

4.1.2. 複数UHI供給者 (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. SAM3X, SAM3Aデバイス (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)ハードウェア */

/* ===== 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 */
```

4.2.2. SAM4Lデバイス (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

// #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_OSCO
// #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周波数選択: 4MHzに対して0、8MHzに対して1、12MHzに対して2 */
// #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

// ===== 全ての必須でない周辺機能のクロックを禁止
// #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任意選択
#define CONFIG_PLL0_SOURCE PLL_SRC_OSCO
// #define CONFIG_PLL0_SOURCE PLL_SRC_GCLK9

/* Fp110 = (Fclk * PLL_mul) / PLL_div */
#define CONFIG_PLL0_MUL (48000000UL / BOARD_OSCO_HZ)
#define CONFIG_PLL0_DIV 1
// #define CONFIG_PLL0_MUL (192000000 / FOSCO) /* Fp11 = (Fclk * PLL_mul) / PLL_div */
// #define CONFIG_PLL0_DIV 4 /* Fp11 = (Fclk * PLL_mul) / PLL_div */

// ==== DFLL任意選択
// #define CONFIG_DFLL0_SOURCE GENCLK_SRC_OSCO
// #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

/* Fdf11 = (Fclk * DFLL_mul) / DFLL_div */
```

```

// #define CONFIG_DFLLO_FREQ          48000000UL
// #define CONFIG_DFLLO_MUL          ((4 * CONFIG_DFLLO_FREQ) / BOARD_OSC32_HZ)
// #define CONFIG_DFLLO_DIV          4
// #define CONFIG_DFLLO_MUL          (CONFIG_DFLLO_FREQ / BOARD_OSC32_HZ)
// #define CONFIG_DFLLO_DIV          1

// ===== USBクロック元任意選択
#define CONFIG_USBCLK_SOURCE          USBCLK_SRC_PLLO
// #define CONFIG_USBCLK_SOURCE          USBCLK_SRC_DFLLO

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

// ===== GCLK9任意選択
// #define CONFIG_GCLK9_SOURCE          GENCLK_SRC_GCLKINO
// #define CONFIG_GCLK9_DIV              1

#endif /* CONF_CLOCK_H_INCLUDED */

```

4.3. conf_clocks.h

4.3.1. SAM D21デバイス (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

/* システムクロックバス形態設定 */
# 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_APBBA_DIVIDER                SYSTEM_MAIN_CLOCK_DIV_1
# define CONF_CLOCK_APBBC_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水晶/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_OUTPUT        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_DFLLL形態設定 - デジタル周波数固定化閉路 */
# define CONF_CLOCK_DFLLL_ENABLE               true
# define CONF_CLOCK_DFLLL_LOOP_MODE            SYSTEM_CLOCK_DFLLL_LOOP_MODE_CLOSED
# define CONF_CLOCK_DFLLL_ON_DEMAND            true

/* DFLLL開路動作形態設定 */
# define CONF_CLOCK_DFLLL_FINE_VALUE           (512)

/* DFLLL閉路動作形態設定 */
# define CONF_CLOCK_DFLLL_SOURCE_GCLK_GENERATOR GCLK_GENERATOR_1
# define CONF_CLOCK_DFLLL_MULTIPLY_FACTOR      (48000000/32768)
# define CONF_CLOCK_DFLLL_QUICK_LOCK           true
# define CONF_CLOCK_DFLLL_TRACK_AFTER_FINE_LOCK true
# define CONF_CLOCK_DFLLL_KEEP_LOCK_ON_WAKEUP true
# define CONF_CLOCK_DFLLL_ENABLE_CHILL_CYCLE   true
# define CONF_CLOCK_DFLLL_MAX_COARSE_STEP_SIZE (0x1f / 8)
# define CONF_CLOCK_DFLLL_MAX_FINE_STEP_SIZE   (0xff / 8)

/* SYSTEM_CLOCK_SOURCE_DPPLL形態設定 - デジタル位相固定化閉路 */
# define CONF_CLOCK_DPPLL_ENABLE               false
# define CONF_CLOCK_DPPLL_ON_DEMAND            false
# define CONF_CLOCK_DPPLL_RUN_IN_STANDBY        true
# define CONF_CLOCK_DPPLL_LOCK_BYPASS           false
# define CONF_CLOCK_DPPLL_WAKE_UP_FAST          false
# define CONF_CLOCK_DPPLL_LOW_POWER_ENABLE      true

# define CONF_CLOCK_DPPLL_LOCK_TIME              SYSTEM_CLOCK_SOURCE_DPPLL_LOCK_TIME_DEFAULT
# define CONF_CLOCK_DPPLL_REFERENCE_CLOCK        SYSTEM_CLOCK_SOURCE_DPPLL_REFERENCE_CLOCK_XOSC32K
# define CONF_CLOCK_DPPLL_FILTER                SYSTEM_CLOCK_SOURCE_DPPLL_FILTER_DEFAULT

# define CONF_CLOCK_DPPLL_REFERENCE_FREQUENCY    32768
# define CONF_CLOCK_DPPLL_REFERENCE_DIVIDER      1
# define CONF_CLOCK_DPPLL_OUTPUT_FREQUENCY       48000000

/* DPPLL GCLK基準形態設定 */
# define CONF_CLOCK_DPPLL_REFERENCE_GCLK_GENERATOR GCLK_GENERATOR_1
/* DPPLL GCLK固定化計時器形態設定 */
# define CONF_CLOCK_DPPLL_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_DFLLL
# 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. SAM3X, SAM3Aデバイス (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ピン)に対応するピンの記述 */
//#define CONSOLE_PINS          {PINS_UART}

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

/* UARTピン形態設定 */
//#define CONF_BOARD_UART_CONSOLE

/* A/D変換器のピン形態設定 */
//#define CONF_BOARD_ADC

/* PWM LED0ピン形態設定 */
//#define CONF_BOARD_PWM_LED0

/* PWM LED1ピン形態設定 */
//#define CONF_BOARD_PWM_LED1

/* PWM LED2ピン形態設定 */
//#define CONF_BOARD_PWM_LED2

/* SPI0ピン形態設定 */
//#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ピン形態設定 */
//#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ピン形態設定 */
//#define CONF_BOARD_USART_RXD

/* USART TXDピン形態設定 */
//#define CONF_BOARD_USART_TXD

/* USART CTSピン形態設定 */
//#define CONF_BOARD_USART_CTS

/* USART RTSピン形態設定 */
//#define CONF_BOARD_USART_RTS

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

/* ADM3312許可ピン形態設定 */
//#define CONF_BOARD_ADM3312_EN

```

```

/* IrDA送受信部停止ピン形態設定 */
//#define CONF_BOARD_TFDU4300_SD

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

//#define CONF_BOARD_SMC_PSRAM

/* LCD EBIピン形態設定 */
//#define CONF_BOARD_HX8347A

/* 背面灯制御ピン形態設定 */
//#define CONF_BOARD_AAT3194

/* USBピン形態設定 */
#define CONF_BOARD_USB_PORT

#endif /* CONF_BOARD_H_INCLUDED */

```

4.4.2. SAM4Lデバイス (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

/* board_init()が呼ばれる時にUSART用汎用入出力を自動初期化 */
//#define CONF_BOARD_COM_PORT

/* USBインターフェース(USB)を許可 */
#define CONF_BOARD_USB_PORT
/* ID検出許可、PB05/USBジャンパ設定ならば注釈を外してください。 */
#define CONF_BOARD_USB_ID_DETECT
/* ホストVBUS制御許可、PC08/USBジャンパ設定ならば注釈を外してください。 */
#define CONF_BOARD_USB_VBUS_CONTROL
/* ホストVBUS制御許可、PC08/USBジャンパ設定ならば注釈を外してください。 */
#define CONF_BOARD_USB_VBUS_ERR_DETECT

/* 基板監視を制御するためにUSARTを許可 */
//#define CONF_BOARD_BM_USART

/* LCD背面灯を初期化 */
#define CONF_BOARD_BL

#endif /* CONF_BOARD_H_INCLUDED */

```

4.4.3. SAM D21デバイス (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い場合は、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。
 - USB-C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。

sleepmgr(休止管理)サービ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。

- ・ 休止管理サービスを初期化してください。
- ・ 応用がアイドル状態の時に休止動作形態を活性化にしてください。

conf_clock.h例

AVRとSAM3/4デバイスについては以下の初期化コード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デバイスについては初期化コード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
```

主アイド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) – コード例

全てのUSBホストに対する共通的なコード例

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デバイス(USB)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_OSCO
#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デバイス(高速(HS)支援付きUSB)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_OSCO
#define CONFIG_USBCLK_DIV 1 // Fusb = Fsys/(2 ^ USB_div)
```

AT32UC3Cデバイス(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_OSCO
#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デバイス(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デバイス(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 - デジ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       (4800000/32768)
# 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 |
|--------|----------|--------|
| 42346A | 2014年12月 | 初版資料公開 |
| 42346B | 2015年12月 | 誤植修正 |



Atmel® | Enabling Unlimited Possibilities®



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

© 2015 Atmel Corporation. / 改訂:Atmel-42346B-USB-Host-Interface-UHI-for-Vendor-Class-Device_AT09341_Application Note-12/2015

Atmel®, Atmelロゴとそれらの組み合わせ、Enabling Unlimited Possibilities®, AVR®, XMEGA®とその他は米国と他の国に於けるAtmel Corporationの登録商標または商標です。ARM®, ARM Connected®ロゴとその他はARM Ltd.の登録商標です。他の用語と製品名は一般的に他の商標です。

お断り: 本資料内の情報はAtmel製品と関連して提供されています。本資料またはAtmel製品の販売と関連して承諾される何れの知的所有権も禁反言あるいはその逆によって明示的または暗示的に承諾されるものではありません。Atmelのウェブサイト¹に位置する販売の条件とAtmelの定義での詳しい説明を除いて、商品性、特定目的に関する適合性、または適法性の暗黙保証に制限せず、Atmelはそれらを含むその製品に関連する暗示的、明示的または法令による如何なる保証も否認し、何ら責任がないと認識します。たとえAtmelがそのような損害賠償の可能性を進言されたとしても、本資料を使用できない、または使用以外で発生する(情報の損失、事業中断、または利益と損失に関する制限なしの損害賠償を含み)直接、間接、必然、偶然、特別、または付随して起こる如何なる損害賠償に対しても決してAtmelに責任がないでしょう。Atmelは本資料の内容の正確さまたは完全性に関して断言または保証を行わず、予告なしでいつでも製品内容と仕様の変更を行う権利を保留します。Atmelはここに含まれた情報を更新することに対してどんな公約も行いません。特に別の方法で提供されなければ、Atmel製品は車載応用に対して適当ではなく、使用されるべきではありません。Atmel製品は延命または生命維持を意図した応用での部品としての使用に対して意図、認定、または保証されません。

安全重視、軍用、車載応用のお断り: Atmel製品はAtmelが提供する特別に書かれた承諾を除き、そのような製品の機能不全が著しく人に危害を加えたり死に至らしめることがかなり予期されるどんな応用("安全重視応用")に対しても設計されず、またそれらとの接続にも使用されません。安全重視応用は限定なしで、生命維持装置とシステム、核施設と武器システムの操作の装置やシステムを含みます。Atmelによって軍用等級として特に明確に示される以外、Atmel製品は軍用や航空宇宙の応用や環境のために設計も意図もされていません。Atmelによって車載等級として特に明確に示される以外、Atmel製品は車載応用での使用のために設計も意図もされていません。

© HERO 2016.

本応用記述はAtmelのAT09341応用記述(Rev.42346B-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識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。