Application Note

AM261x での VBUS ベースの USB ホスト検出を実装

Shaunak Deshpande, Tejas Kulakarni

概要

AM261x Sitara™ Arm® マイクロコントローラは、Sitara AM26x リアルタイム MCU ファミリに属し、次世代の産業および自動車組み込みシステムの厳しい処理ニーズに対応するように設計されています。スケーラブルな Arm® Cortex®-R5F コアをベースとする AM261x は、包括的なペリフェラルセット、統合された安全機能、フレキシブルな通信インターフェイスを含め、高性能のリアルタイム制御を実現します。この中で、USB (ユニバーサル シリアル バス) はデバイス接続のために広く採用されている高速シリアル インターフェイスを提供します。

USB は、USB ホストと USB デバイス間の電力供給とデータ転送を標準化することで、広範なコンシューマ、産業用、車載デバイスを接続するための設計を実現します。正しいホスト検出機能は、有効なホストが存在するときのみ USB デバイスを確実に初期化するための重要です。これにより、準拠の問題を防止し、システムの堅牢性を向上させることができます。

AM261x では、USB コントローラは外部 VBUS 電圧での動作をサポートしています。ただし、デフォルトの実装では、ホスト検出のために VBUS 電圧をアクティブに監視することはできません。代わりに、USB ドライバーコードは、VBUS 電圧を供給するホストが存在するかどうかに関係なく、コントローラーレジスターを PHY に電源を付け,USB モジュールを有効に設定します。このアプローチにより初期化が簡素化されますが、スプリアス列挙の試行、バスの競合、消費電力の増加、USB 仕様の違反など、いくつかの欠点があります。

このアプリケーション ノートでは、1 つの GPIO を使用した単純な実装により、VBUS 監視を使用した真のホスト検出を組み込むように AM261x USB のデフォルトの動作を変更するリファレンス実装を紹介します。この実装では、デバイスは PHY のみを有効にし、有効なホストと VBUS 電圧の存在を確認した後、列挙を開始します。

このアプリケーション ノートで説明されている設計は、リファレンス実装として提供されています。これについては、AM261x での広範な USB ストレステストを受けていません。お客様は、最終アプリケーションに応じてアプローチを適応させ、検証し、認定することをお勧めします。

目次

1 はじめに.....2
2 USB ホスト検出.....3
2.1 一般的な USB ホスト検出フロー.....3
2.2 AM261x USB ホスト検出.....3
3 ハードウェアの変更点.....4
4 ソフトウェアの変更.....5
4.1 USB Synp ドライバの変更.....5
4.2 USB アプリケーションの変更.....7
4.3 USB ドライバとアプリケーションを再構築する手順.....9
4.4 新しいアプリケーションのテスト.....11
5 まとめ.....12
6 参考資料.....12

商標

1 はじめに

AM261x デバイスの USB サブシステムの詳細な説明については、AM26x MCU+ Academy USB モジュール、テクニカル リファレンス マニュアル (TRM) 、および MCU+ SDK の資料を参照してください。図 1-1 に、AM261x 上の USB 2.0 サブシステム (USB2SS) の機能ブロック図を示します。このサブシステムには、ラッパー モジュール、USB コントローラ モジュール、PHY モジュール、外部インターフェイス、内部インターフェイスが含まれます。

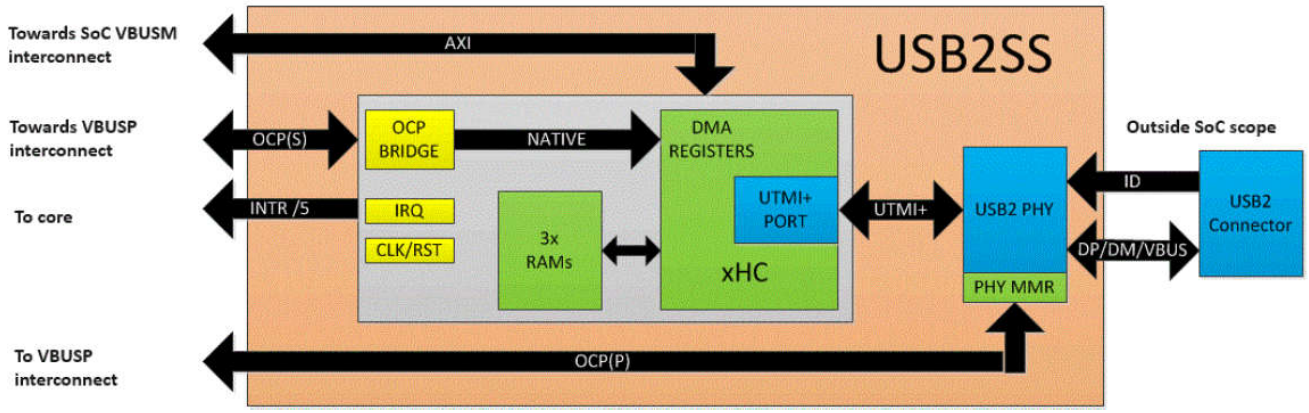

図 1-1. AM261x USB サブシステム

AM261x USB2SS は、外部電源を使用して 5V VBUS を生成するために、外部電源ロジックに依存します。このデバイスは専用の出力信号 (USB0_DRVVBUS) を提供します。この信号はアクティブ high で、外部 VBUS 電源をイネーブルまたはディセーブルにするために使用されます。

ホスト モードで動作している場合、USB コントローラは自動的に USB0_DRVVBUS を high に駆動し、外部チャージ ポンプが有効になり、VBUS ラインで 5V を供給します。

デバイス モードで動作している場合、コントローラは USB0_DRVVBUS を low に駆動し、外部電源をディセーブルにして、AM261x が VBUS をソースしないようにします。

この制御はすべて USB コントローラによって処理され、ハードウェア接続とソフトウェアの初期化が正しく構成されていれば、ユーザーは透過的です。

USB コントローラには、(AM261x デバイスの電源ピン経由で) 自己電源が供給されていることに注意してください。デバイス モードでは、コントローラは外部から供給される VBUS 電圧に依存せずに動作を維持します。代わりに、VBUS の存在がレジスタを介して内部で伝達されます。
MSS_CTRL.CONTROL_USBOTGHS_CONTROL.CONTROL_USBOTGHS_CONTROL_VBUSVALID (0x50D00894)

このレジスタ ビットは、VBUSVALID 信号を USB コントローラに駆動するために使用されます。AM261x はデバイス モードで物理的な VBUS 入力ピンを露出しないため、このビットをソフトウェアで構成する必要があります。デフォルトでは、AM261x USB ドライバは、USB ホストの実際の存在を検証せずに、USB_init () 中に VBUSVALID ビットを設定します。これによりコントローラと PHY が有効になりますが、これは、USB 仕様で定義されている通常のホスト検出メカニズムをバイパスします。

このデフォルトの動作では、次のようないくつかのリスクが発生します。

- スプリアス列挙の試行:ホストが接続されていない場合でも、デバイスは列挙を開始できます。
- バスの競合:デバイスは、USB プロトコルのルールに違反する、有効なホストの VBUS をソースせずに D+/D- ラインを駆動できます
- 消費電力の増加:PHY は不要に電力を供給され続け、低消費電力かつバッテリに制約のあるアプリケーションに影響を与えます
- USB 仕様への違反:USB 規格では、VBUS が検出されるまでデバイスは非アクティブのままであることが明示的に要求されています。

信頼性が高く仕様に準拠した動作を検証するには、デバイスは VBUS 信号の監視に基づいて真のホスト検出を実装する必要があります。適切に検出されると、本デバイスは PHY のみを有効にし、有効なホストが存在するときに列挙を開始するため、前述の問題を防止します。

2 USB ホスト検出

2.1 一般的な USB ホスト検出フロー

USB (ユニバーサル シリアル バス) は、ホストとデバイスが接続されたことを検出し、列挙が実行される前に必要な電気的条件を確立するための堅牢なメカニズムを定義しています。従来型の実装では、この検出は主に VBUS ラインと D+/D– 差動ペアの監視によって達成され、ホストとデバイスの有効な関係が確認された後にのみコントローラがデータ通信をイネーブルにすることを保証します。VBUS ラインは、ホストから供給される 1 次電源レールです。標準の USB 2.0 実装では、ホストは USB 仕様で定義されている電流能力 (通常、High-Speed USB 2.0 では 500mA、USB 3.x ではそれより高い電流) で VBUS 上に 5V を供給します。デバイスがこの回線を駆動することは想定されていません。

従来型の USB デバイス コントローラには、5V 信号の存在を検出する VBUS 検出回路が内蔵されています。検出は、次の 2 つのアプローチのいずれかを使用して実行されます。

- コンパレータ ベースの検出:内蔵コンパレータは VBUS を継続的に監視します。電圧が特定のスレッショルド (たとえば、仕様で定義された最小 4.4V) を上回ると、コントローラは、有効なホスト接続が存在することを示すステータス フラグを設定します。
- 専用の VBUSVALID 入力:多くのコントローラは、USB PHY に直接接続する VBUS ピンを露出しています。VBUS 入力が有効範囲を超えると、PHY は内部 VBUSVALID 信号をアサートします。

この VBUSVALID 信号は重要です。この信号は、有効なホストが接続されるまで USB デバイスのロジックを有効にすることを妨げるからです。このバスがないと、デバイスが誤ってバスを駆動しようとし、ライン上で未定義の状態や競合が発生する可能性があります。USB 仕様では、信頼性の高い検出を検証するために、一連のデバウンス間隔を規定しています。次に例を示します。

- VBUS が 4.4V を上回ると、デバイスは 100mA を超える電流を引き込む前に、少なくとも 100ms 待機する必要があります。
- ホストは、D+ または D- プルアップを有効な接続と解釈する前に、デバウンス期間 (少なくとも 100ms) を適用します。

このようなタイミング要件により、ノイズ、ホットプラグ イベント、不安定な電源による誤検出を防止できます。従来の USB PHY はこのデバウンス ロジックを内蔵しており、条件が安定するまで内部イネーブル信号を自動的にゲーティングします。

従来の実装フロー:

VBUS 検出を利用した代表的な USB デバイス初期化シーケンスは、次のように進みます。

1. アイドル状態: デバイス PHY に電力が供給されていますが、トランシーバは無効のままです。
2. VBUS 検出:ホストは 5V を供給し、コントローラ内部で VBUSVALID がアサートされます。
3. デバイス イネーブル:USB コントローラは、トランシーバとプルアップ抵抗を適切なライン (D+ または D–) に有効にします。
4. ホストがデバイスを検出:ホストがプルアップを検出し、リセット信号を開始します。
5. 列挙:デバイスはリセットに応答し、速度をネゴシエートし、記述子交換で列挙シーケンスを開始します。

このシーケンスにより、接続と USB 仕様への準拠を適宜検証できます。

2.2 AM261x USB ホスト検出

AM261x の場合、前のセクションで説明したように、VBUS ラインはコントローラの構成に使用しません。代わりに、ソフトウェア ドライバはコントローラの VBUSVALID ビットを構成します。これを変更するには、次のセクションで説明する手順に従います。

3 ハードウェアの変更点

デフォルトの TI の AM261x launchpad は、ホスト存在を USB コントローラに通知するために VBUS 信号に依存しないため、真のホスト検出を実装するために、シンプルなハードウェアの変更を導入して、VBUS 信号をデバイス ファームウェアによって観測できるようにすることができます。この変更には、5V VBUS 信号を AM261x の汎用入出力 (GPIO) ピンに配線することが含まれます。本デバイスの I/O は 3.3V のロジック レベルで動作するため、5V 入力を 3.3V 互換のレベルに安全にスケーリングするには、抵抗分圧器が必要です。分圧器は、直列に接続された 2 つの抵抗で構成され、接合部は抵抗値の比に比例する出力電圧を供給します。この場合、分圧器により、VBUS が存在するときに GPIO ピンが安全なロジック high 信号を受信するようになります。

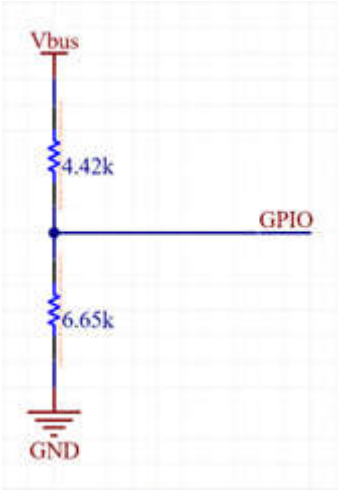

図 3-1. VBUS 電圧を AM261x GPIO ピンに駆動する

このアプリケーション ノートで説明されているリファレンス実装では、次のようになります。

- VBUS 監視用には、AM261x LaunchPad 上の GPIO6 を選択します。
- GPIO6 は J6/J8 BoosterPack ヘッダのピン 53 でアクセスできるため、GPIO6 は外付け分圧器回路とのインターフェイスを行うのに便利です。
- 図 3-1 に示すように単純な分圧器回路が実装されています。AM261x GPIO がロジック high として検出されるように、USB の 5V を約 3V まで降圧します。
- 配線された VBUS 信号は GPIO6 に接続されており、立ち上がりエッジと立ち下がりエッジの両方で割り込みをトリガするよう構成されています。
  - 立ち上がりエッジ:ホスト接続イベントを示します。
  - 立ち下がりエッジ:ホストの切断イベントを示します。

これに基づいて、標準 MCU_PLUS_SDK ソフトウェアを変更して、GPIO 入力に基づいて USB ドライバを初期化し、ホスト接続が検出されたときにのみ USBSS PHY とコントローラを初期化します。同様に、ホストが切断されたときに de-init シーケンスを実行します。このやり方により、AM261x は有効なホストが検出されたときのみ USB サブシステムをアクティブにするようになり、スプリアス動作を防止し、デバイスの動作を USB 仕様に整合させることができます。

4 ソフトウェアの変更

USB 用 MCU_PLUS_SDK のデフォルトの USB ドライバは、USB アプリケーションから USB_init () 関数を呼び出します。この関数は、example.syscfg ti_drivers_open_close.c ファイルから自動生成されたコードから呼び出されます。Drivers_usbOpen() 関数。

この関数は、USB dwc_usb3_dev ハンドラのリセット、USB PHY およびコントローラ クロックの構成、コントローラの制御レジスタの設定、PHY のオン、UTMI OTG レジスタの構成を行い、PHY の起動を完了します。デフォルトドライバは、USB ポート上に本当に存在するかどうかを検出せず、USB コントローラの VBUSVALID ビットをプログラムしてホスト接続を強制的に実行します。これは機能には影響しません。

4.1 USB Synp ドライバの変更

Mcu_plus_sdk/source/usb/synp/soc/ パスにある device_wrapper.c ファイルのデフォルトの USB_init () 関数を変更します。USB_init () は、GPIO6 の割り込みを登録し、VBUS ラインのステータスに基づいてドライバの初期化を続行することができます。

```
#include <drivers/gpio.h>
#include <kernel/dpl/AddrTranslateP.h>
#include <kernel/dpl/ClockP.h>

#define HOST_CONNECTED    (1U)
#define HOST_DISCONNECTED    (0U)

uint8_t gHostConnected = false;
uint32_t gGpioBaseAddr = CSL_GPIO0_U_BASE;
uint32_t gHostDetectionPin = 6U; /* GPIO Pin number */
uint32_t gIntrNum = CSLR_R5FSS0_CORE0_INTR_GPIO_INTRXBAR_OUT_14;
HwiP_Object gGpioHwiObject;

void USB_hostDetectGpioIsrFxn(void *args);

void USB_init()
{
    usb_handle.cfg_base = USB_DWC_3;
    usb_handle.dwc_usb3_dev = NULL;
    /* Register GPIO interrupt */
    HwiP_Params     hwiPrms;
    HwiP_Params_init(&hwiPrms);
    hwiPrms.intNum = gIntrNum;
    hwiPrms.callback = &USB_hostDetectGpioIsrFxn;
    hwiPrms.args = (void *)gHostDetectionPin;
    hwiPrms.isPulse = TRUE;
    uint8_t retVal = HwiP_construct(&gGpioHwiObject, &hwiPrms);
    DebugP_assert(retVal == SystemP_SUCCESS );
    /* Check initial state of GPIO - if host is already connected, GPIO will be
high */
    if(GPIO_pinRead(gGpioBaseAddr, gHostDetectionPin) == HOST_CONNECTED)
    {
        gHostConnected = true;
        if (usb_phy_power_sequence() == USB_PHY_OK )
        {
            usbdIntrConfig();
        }
        else
        {
            DebugP_assert(FALSE);
        }
        tusb_init(); /* By default, this is a part of Drivers_usbOpen() in
ti_drivers_open_close.c */
    }
}
```

図 4-1. ドライバに GPIO 割り込みを登録しています

```
void USB_hostDetectGpioIsrFxn(void *args)
{
    uint32_t    pinNum = (uint32_t)args;
    uint32_t    bankNum =  GPIO_GET_BANK_INDEX(pinNum);
    uint32_t    intrStatus;

    /* Get and clear bank interrupt status */
    intrStatus = GPIO_getBankIntrStatus(gGpioBaseAddr, bankNum);
    GPIO_clearBankIntrStatus(gGpioBaseAddr, bankNum, intrStatus);

    if(gHostConnected==true && (GPIO_pinRead(gGpioBaseAddr, pinNum) ==
HOST_DISCONNECTED))
    {
        gHostConnected = false;
        /* VBUSVALID. set 0
        */
        HW_WR_FIELD32_RAW(MSS_CTRL + MSS_CTRL_CONTROL_USBOTGHS_CONTROL,
0x00000004,0,0x0);
    }
    else if((GPIO_pinRead(gGpioBaseAddr, pinNum) == HOST_CONNECTED) &&
gHostConnected==false)
    {
        gHostConnected = true;
        if (usb_phy_power_sequence() == USB_PHY_OK )
        {
            usbdIntrConfig();
        }
        else
        {
            DebugP_assert(FALSE);
        }
        tusb_init(); /* By default, this is a part of Drivers_usbOpen() in
ti_drivers_open_close.c */
    }
}
```

図 4-2. ホスト検出用の GPIO ISR 機能

4.2 USB アプリケーションの変更

まず、自動生成された syscfg コードから TinyUSB 初期化を無効にし、ホスト接続が検出された場合にのみ実行します。

tusb_init() を無効にするには、*mcu_plus_sdk/source/sysconfig/usb/.meta/tinyusb/templates/tinyusb_open_close_config.c.xdt* ファイルを変更し、「tusb_init()」関数をコメントアウトします。代わりに、この初期化は USB ドライバの初期化シーケンスに移動されます。

```
void Drivers_usbOpen(void)
{
    /* initialize USB HW for TI SOC */
    % if(common.getSocName() == "am64x" || common.getSocName() == "am243x") {
    usb_init(&gUsbInitParam);
    % } else {
    USB_init();
    %}
    /* Comment out tiny USB init and move it to the USB driver */
    /* tusb_init(); */

    return;
}
```

図 4-3. Syscfg テンプレート ファイルから TinyUSB init を削除しています

ここで、USB アプリケーションで example.syscfg を開き、次の手順を実行します。

1. GPIO ピンを追加。この実装では、AM261x Launchpad のブースタパックテストヘッダ内で J6 ピン 53 としても利用できる GPIO6 を使用します。ホスト接続で VBUS ラインが high にプルアップされ、切断時に low になるため、トリガタイプを立ち上がりと立ち下がりの両方としてマークします。そのため、どちらの場合も ISR がトリガされることが予想されます。

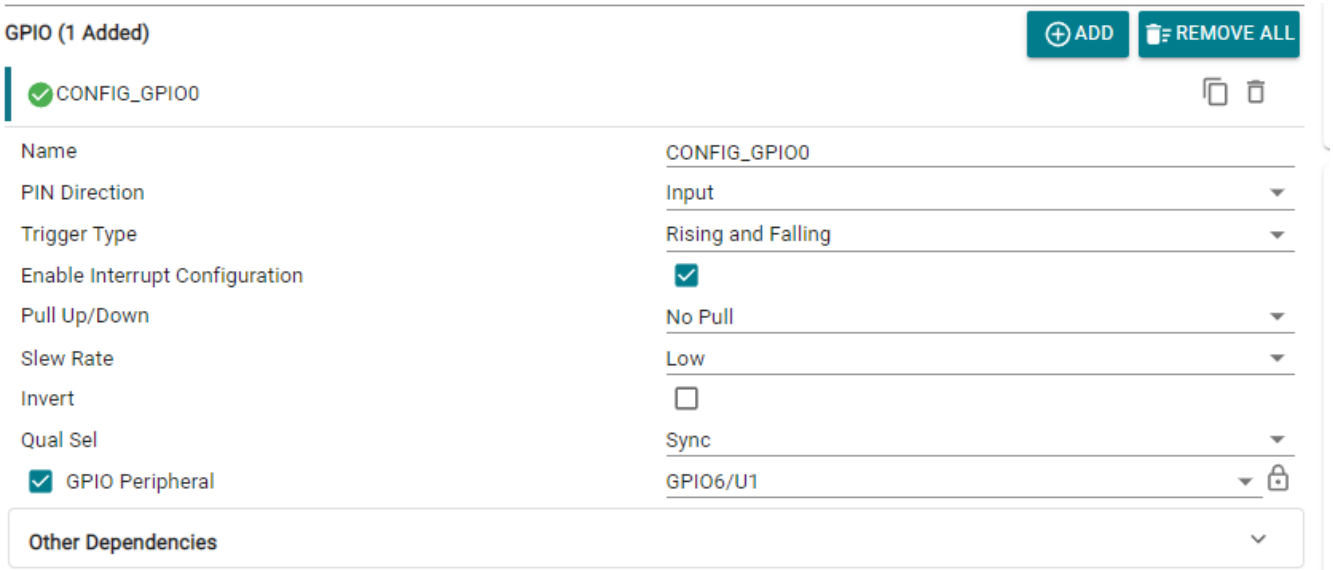

図 4-4. SysCfg GPIO 構成

2. GPIO クロスバー割り込み配線を構成します。GPIO0 ～ GPIO15 ピンは、INTR_0 クロスバー出力を使用します。そのため、GPIO6 の場合、INTR_0 と VIM_MODULE0_0 を構成します。他の GPIO ピンを使用する場合は、割り込み構成が正しく処理されていることを確認してください。

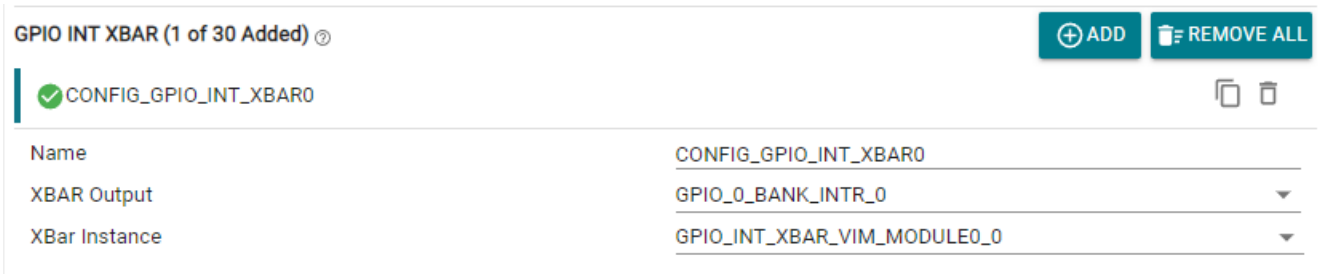

図 4-5. SysCfg GPIO INT XBAR の構成

3. USB モジュールを設定します

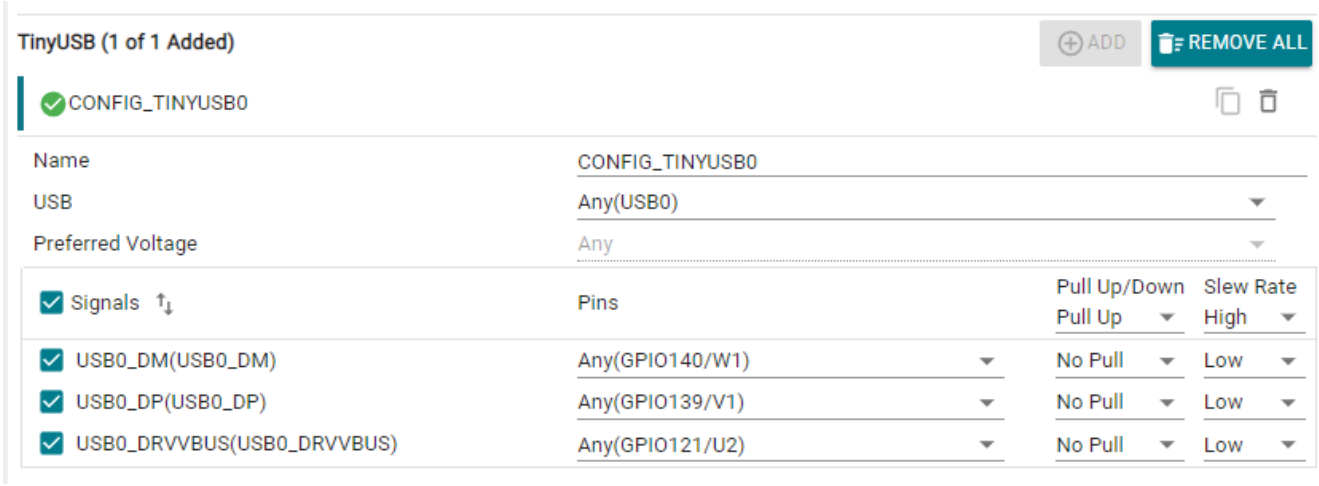

図 4-6. SysCfg USB 構成

4.3 USB ドライバとアプリケーションを再構築する手順

ドライバとアプリケーションは構成済みです。次のステップは、USB ライブラリと USB アプリケーションをリビルドしてテストすることです。

4.3.1 USB ライブラリの再構築

MCU_PLUS_SDK ライブラリは、CCS から再ビルドすることはできません。次のコマンドを実行して、MCU_PLUS_SDK ディレクトリから TINYUSB スタックと USB ドライバを再ビルドします。

```
# re-building NoRTOS USB Lib
gmake -sj -f makefile.am261x usbd_synp_nortos_r5f.ti-arm-clang PROFILE=debug

# re-building FreeRTOS USB Lib
gmake -sj -f makefile.am261x usbd_synp_freertos_r5f.ti-arm-clang PROFILE=debug

# re-building TUSB stack for CDC (NoRTOS)
gmake -sj -f makefile.am261x usbd_tusb_cdc_nortos_r5f.ti-arm-clang PROFILE=debug

# re-building TUSB stack for CDC (FreeRTOS)
gmake -sj -f makefile.am261x usbd_tusb_cdc_freertos_r5f.ti-arm-clang
PROFILE=debug

# re-building TUSB stack for DFU (NoRTOS)
gmake -sj -f makefile.am261x usbd_tusb_dfu_nortos_r5f.ti-arm-clang PROFILE=debug

# re-building TUSB stack for DFU (FreeRTOS)
gmake -sj -f makefile.am261x usbd_tusb_dfu_freertos_r5f.ti-arm-clang
PROFILE=debug

# re-building TUSB stack for NCM (NoRTOS)
gmake -sj -f makefile.am261x usbd_tusb_ncm_nortos_r5f.ti-arm-clang PROFILE=debug

# re-building TUSB stack for NCM (FreeRTOS)
gmake -sj -f makefile.am261x usbd_tusb_ncm_freertos_r5f.ti-arm-clang PROFILE=de-
bug
```

図 4-7. Makefiles を使用して USB ライブラリを再構築しています

4.3.2 USB アプリケーションの再構築

4.3.2.1 CCS ビルド

アプリケーションがすでに CCS ワークスペースに存在している場合は、アプリケーションを右クリックし、*「Re-build」(ビルド)* をクリックします。これにより、更新された USB ドライバが再構築され、再利用されます。

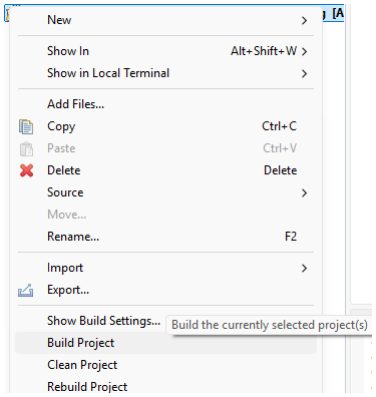

図 4-8. CCS 例のビルド ステップ

4.3.2.2 コマンド ライン ビルド

コマンド ライン *make/gmake* コマンドを使用してアプリケーションをビルドするには、次のコマンドを実行します。

```
gmake -sj -C examples/usb/device/cdc_echo/am261x-lp/r5fss0-0_nortos/ti-arm-clang/
PROFILE=debug all
```

図 4-9. CDC アプリケーション

```
gmake -sj -C examples/usb/device/dfu/am261x-lp/r5fss0-0_nortos/ti-arm-clang/ PRO-
FILE=debug all
```

図 4-10. DFU アプリケーション

```
gmake -sj -C examples/usb/device/ncm/am261x-lp/r5fss0-0_nortos/ti-arm-clang/
PROFILE=debug all
```

図 4-11. NCM アプリケーション

4.4 新しいアプリケーションのテスト

上記のすべてのハードウェアとソフトウェアを変更した後で、USB ホストの検出が VBUS ラインに依存するようになったかどうかをテストします。次の方法を試して、USB が正常に動作していることを確認します。

1. ロジック アナライザまたはオシロスコープを使用して、AM261x-LP の GPIO6 ピン (または使用されている GPIO) にプローブを当て、Launchpad に電源を投入して、ケーブルを数回接続 / 切断し、実際の GPIO 信号が USB VBUS ラインでドライバになっていることを確認します。
2. AM261x-LP を適切なブートモードに構成します。テストには、AM261x デバイスに SBL がヌルにフラッシュ書き込みされた、開発ブートモードまたは OSPI ブートモードを使用することを TI の推奨します。
3. Code Composer Studio で、USB アプリケーション向けのデバッグ セッションを開始し、R5F コアに接続して、USB アプリケーション用のデバッグ ビルド バイナリをロードします。

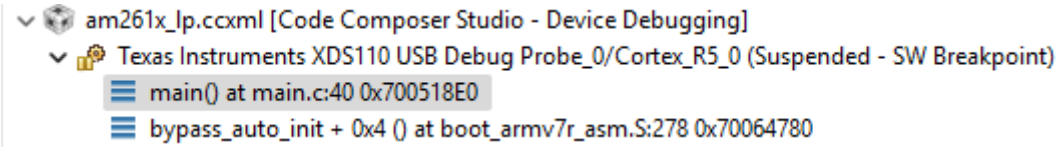

図 4-12. R5_0 コアへのアプリケーションのロード

4. USB_hostDetectGpioIsrFxn () にブレークポイントを設定して、割り込みが生成され、ISR がトリガされていることを確認します。
5. アプリケーションの実行中は、USB ケーブルを数回接続して取り外し、ISR が一貫してヒットしているかどうかを確認します。

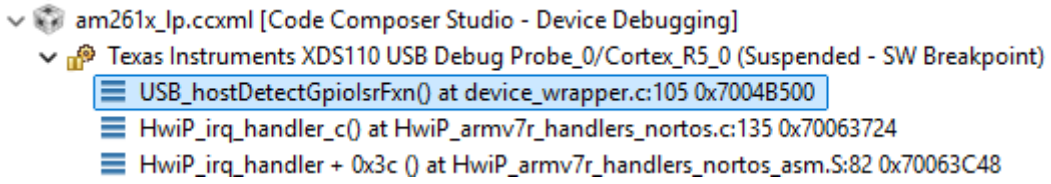

図 4-13. ホストが切断して接続したとき、アプリケーションは GPIO ISR で停止します

6. USB を再接続した後、USB 転送が機能するかどうかを確認してください。(CDC アプリケーションの場合は、COM ポート通信が機能しているかどうかを確認します。DFU アプリケーションが機能しない場合は、ファイル転送が期待どおりに機能するかどうかを確認します)

5 まとめ

これまでの手順に従うと、真の信頼性の高い USB ホスト検出を実装できます。この検出機能は VBUS 電圧ラインを使用してホストの存在を検出し、AM261x USB での信頼性や通信の潜在的な問題を回避し、真のホスト検出を検証することができます。

6 参考資料

1. テキサス インスツルメンツ、『*AM261x Sitara™ マイクロコントローラ*』、データ シート。
2. テキサス インスツルメンツ、*『AM261x テクニカル リファレンス マニュアル』*、テクニカル リファレンス マニュアル。
3. テキサス インスツルメンツ、*AM261x MCU+ SDK 10.02.00*、ドキュメント。
4. テキサス インスツルメンツ、*AM261x MCU+ SDK 10.02.00*、ドキュメント。
5. テキサス インスツルメンツ、*『AM261x MCU+ SDK USB サンプル』*、ドキュメント。
6. USB、*USB 2.0 仕様*、ドキュメント。
7. テキサス・インスツルメンツ、*『AM2612：AM261x に関する FAQ のリスト*、FAQ。

重要なお知らせと免責事項

TI は、技術データと信頼性データ (データシートを含みます)、設計リソース (リファレンス デザインを含みます)、アプリケーションや設計に関する各種アドバイス、Web ツール、安全性情報、その他のリソースを、欠陥が存在する可能性のある「現状のまま」提供しており、商品性および特定目的に対する適合性の黙示保証、第三者の知的財産権の非侵害保証を含むいかなる保証も、明示的または黙示的にかかわらず拒否します。

これらのリソースは、TI 製品を使用する設計の経験を積んだ開発者への提供を意図したものです。(1) お客様のアプリケーションに適した TI 製品の選定、(2) お客様のアプリケーションの設計、検証、試験、(3) お客様のアプリケーションに該当する各種規格や、その他のあらゆる安全性、セキュリティ、規制、または他の要件への確実な適合に関する責任を、お客様のみが単独で負うものとします。

上記の各種リソースは、予告なく変更される可能性があります。これらのリソースは、リソースで説明されている TI 製品を使用するアプリケーションの開発の目的でのみ、TI はその使用をお客様に許諾します。これらのリソースに関して、他の目的で複製することや掲載することは禁止されています。TI や第三者の知的財産権のライセンスが付与されている訳ではありません。お客様は、これらのリソースを自身で使用した結果発生するあらゆる申し立て、損害、費用、損失、責任について、TI およびその代理人を完全に補償するものとし、TI は一切の責任を拒否します。

TI の製品は、TI の販売条件 、 TI の総合的な品質ガイドライン 、 ti.com または TI 製品などに関連して提供される他の適用条件に従い提供されます。TI がこれらのリソースを提供することは、適用される TI の保証または他の保証の放棄の拡大や変更を意味するものではありません。 TI がカスタム、またはカスタマー仕様として明示的に指定していない限り、TI の製品は標準的なカタログに掲載される汎用機器です。

お客様がいかなる追加条項または代替条項を提案する場合も、TI はそれらに異議を唱え、拒否します。

最終更新日：2025 年 10 月