参考資料

Application Note

MCU+SDK を使用した SysConfig による開発の迅速化

Tushar Thakur, Anil Swargam, Soumya Tripathy

概要

このアプリケーション ノートでは、AM243x、AM275x、AM6x デバイス向けに SysConfig ツールを MCU+SDK と統合する役割について説明します。SysConfig では、ピン マルチプレクシング、クロック構成、電力ドメイン構成、ドライバ構成、ボード ペリフェラル構成、領域アドレス変換 (RAT) 、メモリ管理ユニット (MMU) 、メモリ保護ユニット (MPU) の構成におけるソース ファイルを自動生成することで、開発を簡素化できます。

また、このアプリケーション ノートは、開発者が TI の SOC で SysConfig を効果的に使用できるように、ステップバイステップのガイダンスや使用事例、トラブルシューティングのヒントも掲載しています。

目次

商標

1 はじめに

SysConfig は、MCU+SDK と統合された対話型の構成ツールであり、TI の SoC 向けのデバイス初期化とドライバ設定を自動化することができます。これにより、構成の競合が検出され、初期化ファイルが生成され、カスタム ソフトウェア プロジェクトや MCU+SDK プロジェクトへの統合が簡素化されます。デベロッパーは SysConfig を使用して、直観的な GUI またはコマンド ライン インターフェイスを経由して、クロック、PinMux、MPU/MMU/RAT 領域、ドライバ インスタンスを構成できます。

このツールでサポートされている機能は以下のとおりです。

- **システムの初期化:**SysConfig (CodeGen) ツールは、AM243x、AM275x、AM6x の各デバイス向けの初期化コードを生成し、ペリフェラルのセットアップ、クロック構成、割り込み処理、PinMux 構成、MPU、MMU、RAT の各設定をカバーします。詳細は、「*システムの初期化*」を参照してください。
- **PinMux の視覚化:**このツールによって、デバイスとピンのグラフィック表示、利用可能なすべての PinMux オプションの表示、各ピンに対してユーザーが選択するモードの強調表示が可能です。詳細については、*CCS の SysConfig 例* (5)を参照してください。
- **エラー検出:**SysConfig は構成を検証し、設定が正しくない場合のエラーを報告します。これにより、ピン構成間の競合が自動的に検出されます。詳細については、「*ピンの競合*」を参照してください。
- **依存関係の識別:**このツールは、デバイス内のモジュール間の依存関係を特定し、必要なペリフェラルが一貫して構成されるよう保証します。
- **リソース競合の検出:**モジュールが他のペリフェラルに依存する場合、SysConfig は競合をチェックします。依存関係のペリフェラルがすでに使用されている場合、ツールはリソース競合エラーをフラグ付けします。詳細については、「*リソースの競合*」を参照してください。

注記: サポート対象のデバイス ファミリは次のとおりです。

- AM243x、AM64x
- AM62Lx
- AM62Ax
- AM62Dx、AM275x
- AM62Px、AM62x

1.1 SysConfig CodeGen ツール

SysConfig CodeGen ツールは、ソース ファイルとヘッダー ファイルを生成します。これらのファイルを MCU SDK サンプルとともに使用して、上記の機能を実現します。CodeGen ツールは、MCU+SDK が提供する Sciclient (TISCI) API を内部的に使用して、DMSC ファームウェアとの通信を通じて、クロック、リセット、電源ドメインを管理します。

SysConfig CodeGen ツールを開くために必要な SysConfig CodeGen Tool のパラメータを参照してください。

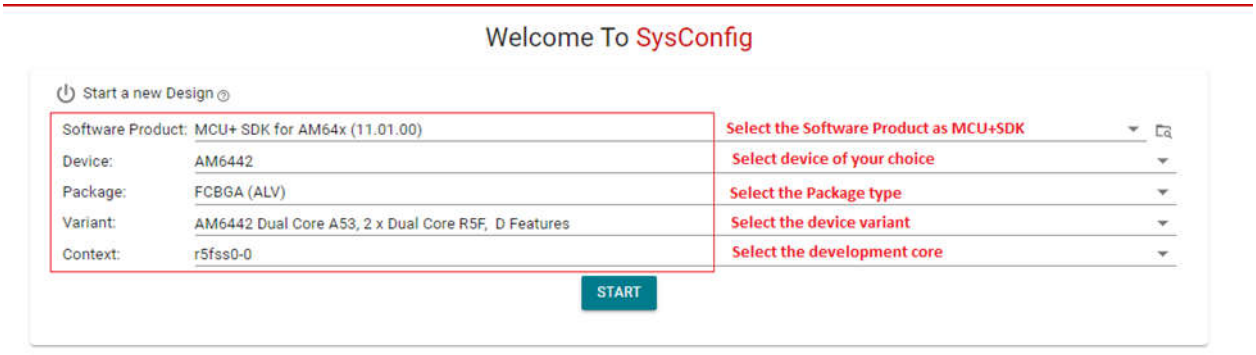

図 1-1. SysConfig CodeGen ツール

CodeGen Tool Generated Files に表示される、SysConfig CodeGen ツールのビュー。

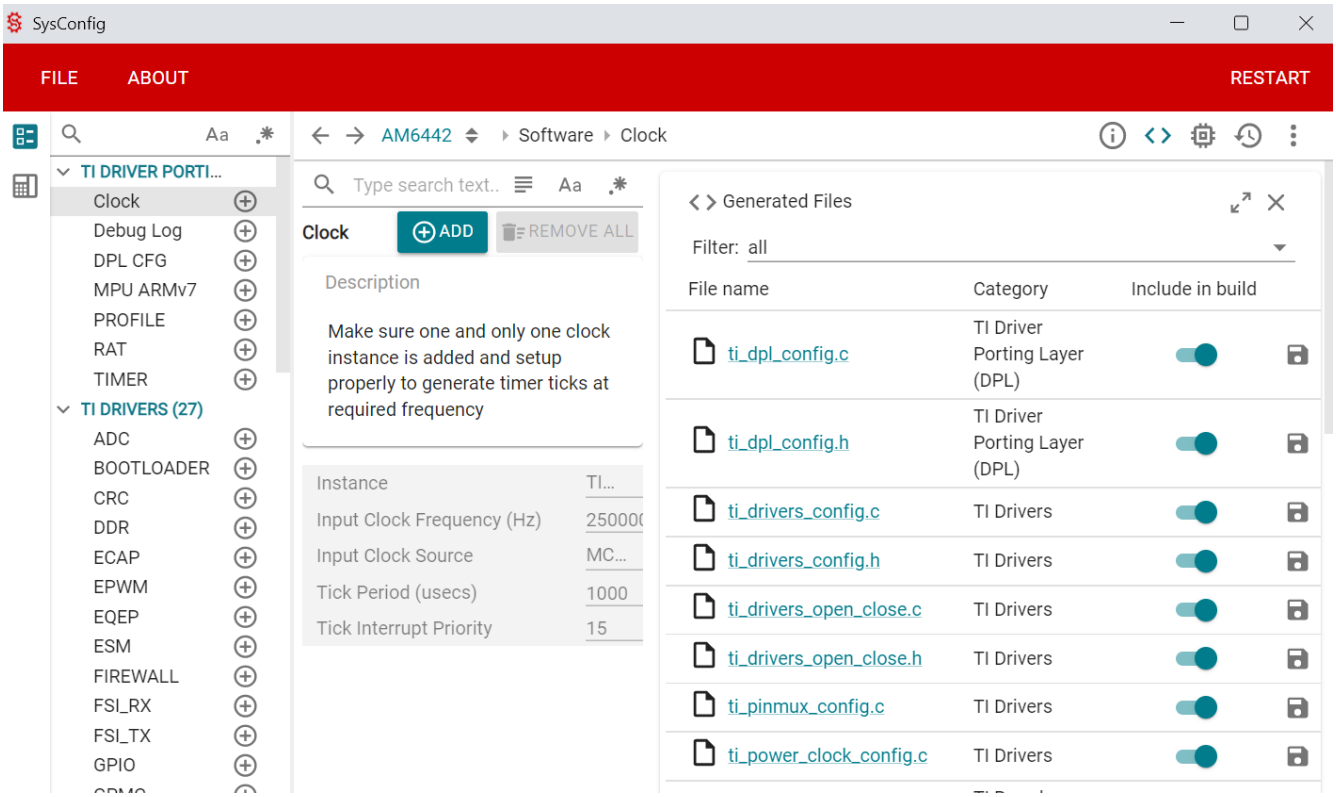

図 1-2. CodeGen ツールで生成されたファイル

2 入門ガイド

SysConfig ツールは、オンライン開発とオフライン開発の両方で利用できます。

ツールにアクセスするには、以下のリンクを参照してください。

1. SysConfig ツールのオフライン バージョンをダウンロードするには、https://www.ti.com/tool/download/SYSCONFIG を参照してください。
2. このツールのオンライン バージョンは、https://dev.ti.com/sysconfig/#/start からアクセスできます。

注

このツールのオンライン バージョンは、SysConfig の最新バージョンです。オンラインの SysConfig ツールを使用するときに、MCU SDK のリリース ノートで互換性の問題を確認してください。

2.1 SysConfig の起動方法 (GUI およびコマンドライン)

SysConfig ツールは、GUI (グラフィカル ユーザー インターフェイス) または CLI (コマンド ライン インターフェイス) を使用して起動できます。

- GUI でツールを開くには、sysconfig ディレクトリに移動し、**sysconfig_gui.bat** ファイルをダブルクリックします。
- CLI を使用して SysConfig CodeGen ツールを開くには、以下の手順に従います。
  - makefile が表示されるまで、サンプル ディレクトリに移動します。
  - 次のコマンドを実行します。Linux 用の *make* と Windows 用の *gmake* を使用します。

```
> {gmake|make} -s syscfg-gui
```

- CLI から SysConfig ツールを実行するには、次のコマンドを使用します。次のコマンドは、SysConfig CodeGen ツールによって生成されたすべてのファイルを出力します。

```
> cd ${SysConfig_root} > sysconfig_cli.bat -s ${MCU+SDK_root}\.metadata\product.json -d AM6442 -o $
{Project_path}\debug ${Project_path}\example.syscfg
```

2.2 CCS や Makefile ビルドとの統合

SysConfig ツールは、CodeGen ツールに関するすべての情報を含む **product.json** ファイルで開始されます。

ここで説明する情報は、SysConfig スタンドアロン ツールと CCS と統合されているものの両方に適用できます。

CCS プロジェクトで SysConfig プロジェクトのプロパティを表示します。

1. プロジェクト名を右クリックし、**Properties** を選択します。
2. **Build** オプションで **Sysconfig** を選択し、すべての SysConfig オプションを表示します。

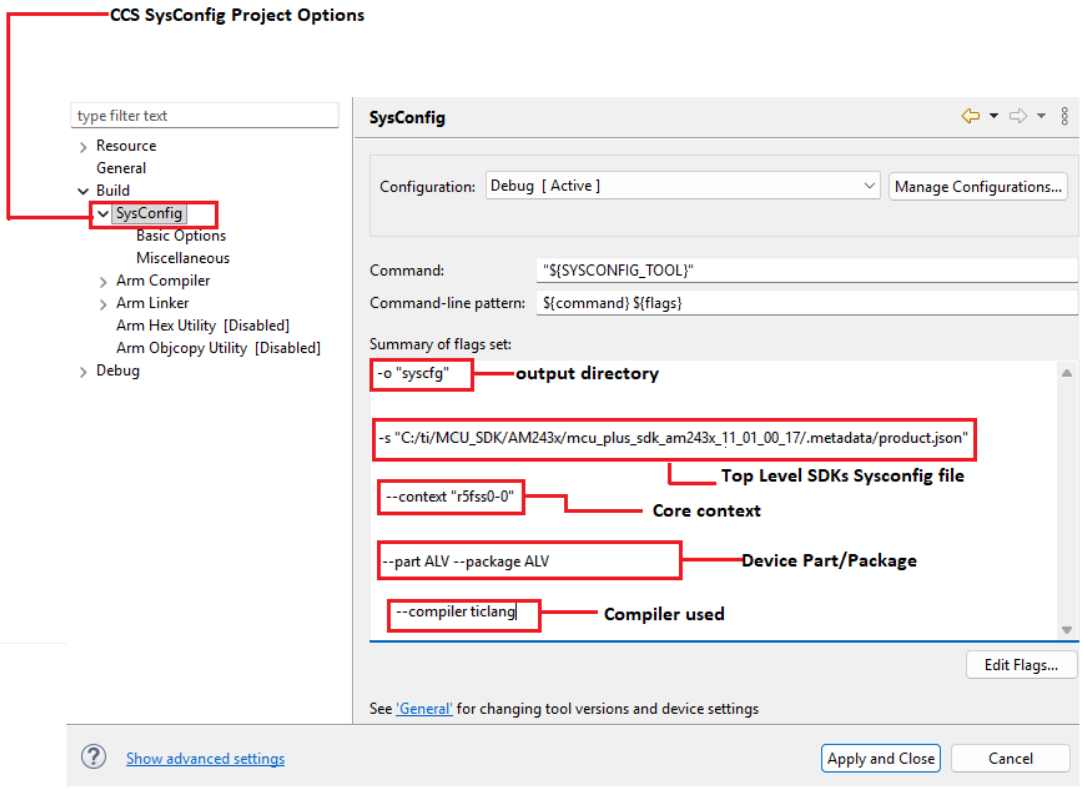

図 2-1. CCS SysConfig プロジェクトのプロパティ

3. デバイス ファミリとトップ レベルの SysConfig **product.json** ファイルを変更 / 表示するには、**Basic Options** を選択します。

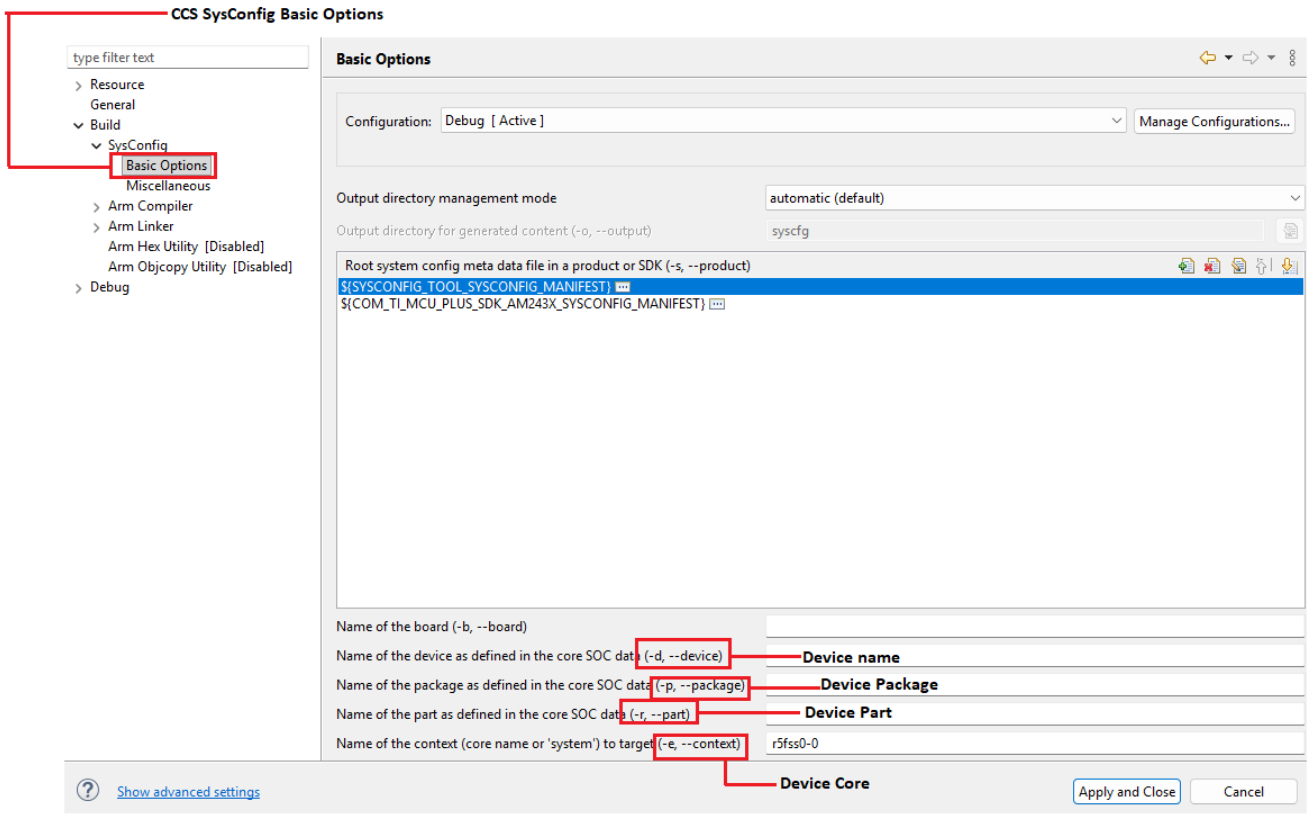

図 2-2. SysConfig 基本オプション

4. デバイスのパッケージ / パーツを変更 / 表示するには、**Miscellaneous** を選択します。

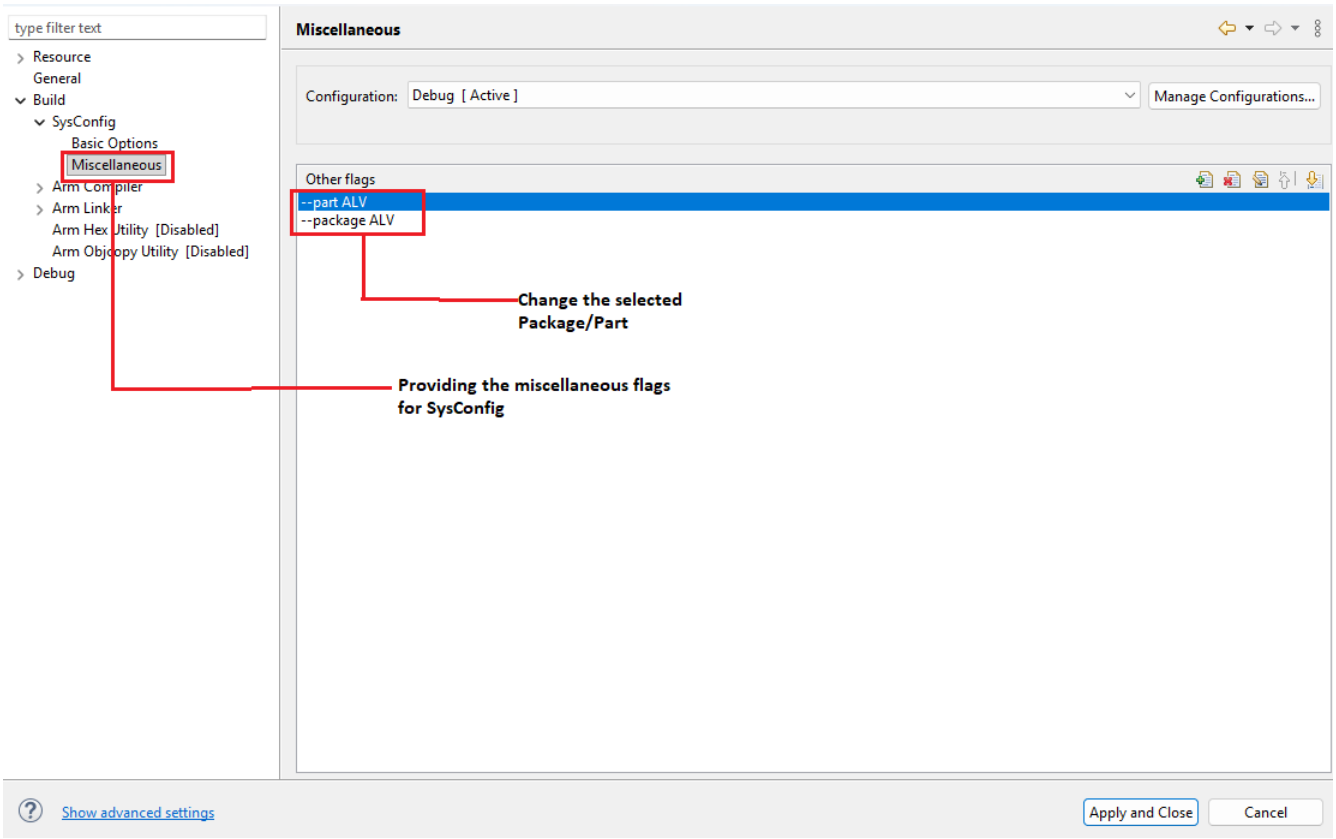

図 2-3. SysConfig のその他のオプション

2.3 MCU SDK 内の SysConfig ファイルの場所

2.3.1 既存の SysConfig ファイルの使用

MCU+SDK に含まれるすべてのサンプルには、SysConfig CodeGen ツールを使用して構成および初期化を行うペリフェラルの詳細が含まれている **example.syscfg** ファイルが含まれています。このツールはこのファイルを入力として受け取り、構成済みのペリフェラルに必要な出力ファイルを生成します。

example.syscfg ファイルは、**${MCU+SDK}/examples/${name}/${device}/${core}/example.sysconfig** にあります。

2.3.2 新しい SysConfig ファイルの作成

example.syscfg ファイルは、SysConfig CodeGen Tool で指定されているように CodeGen ツールを開くことで、最初から生成できます。

このツールは、**untitled.syscfg** ファイルを自動的に生成します。このファイルは保存後、MCU SDK で使用できます。

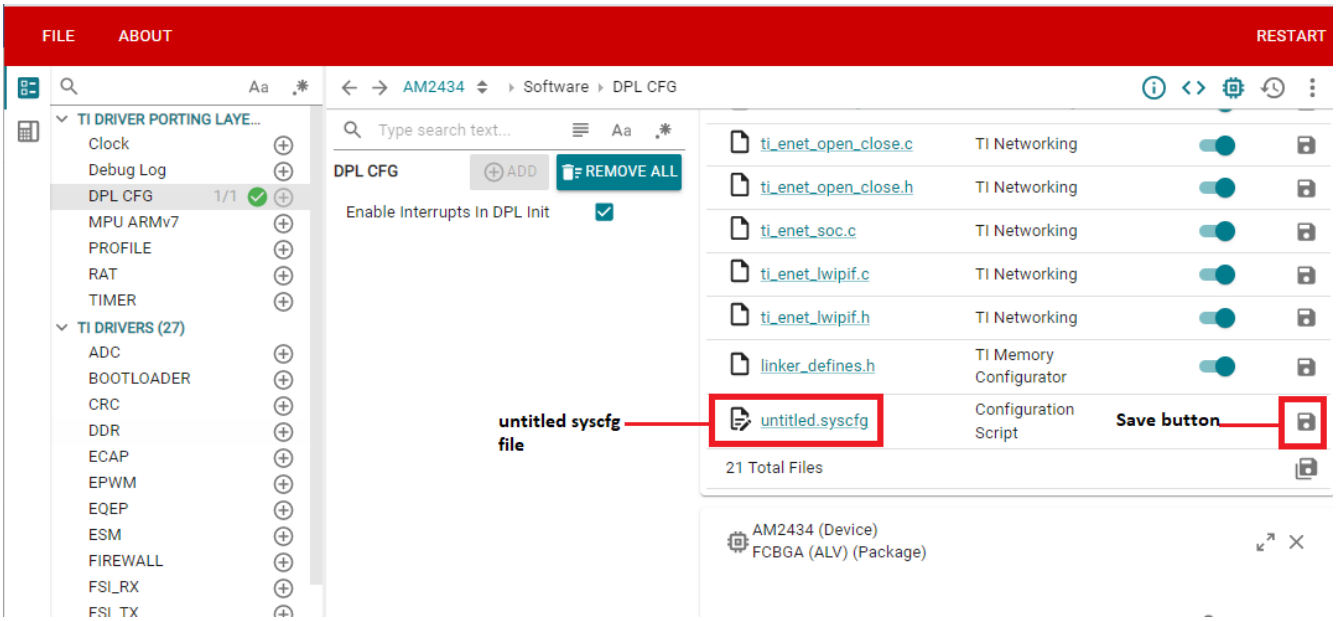

図 2-4. 生成された syscfg ファイル

3 CCS での SysConfig 例

3.1 I2C 読み取りの例

SysConfig CodeGen ツールを使用して開発を開始するには、SysConfig をサポートする MCU SDK に含まれている既存のサンプルをインポートしてください。

1. CCS を起動し、例：**i2c_read_r5fss0-0_nortos** をインポートします。
   a. **Project → Import CCS Project** を選択します。
   b. **${MCU+SDK}\examples\drivers\i2c\i2c_read\am64x-evm\r5fss0-0_nortos** を参照します。
   c. プロジェクトを選択し、インポートします。
2. CCS プロジェクト内で、ユーザーは残りのアプリケーション ファイルとともに syscfg ファイルを確認できます。

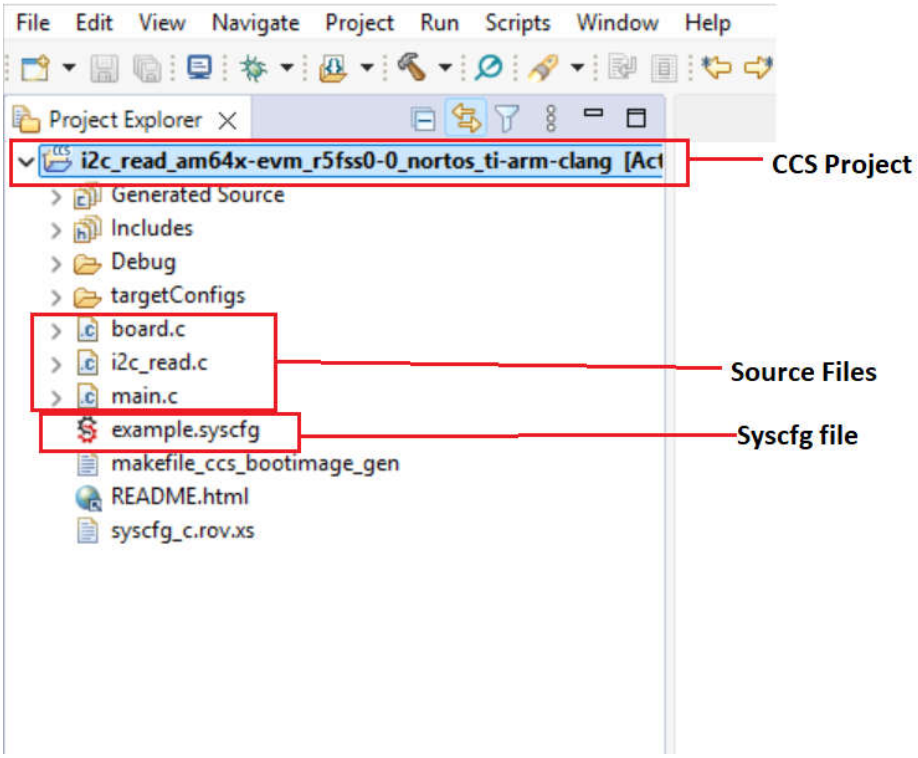

図 3-1. サンプル プロジェクト

3. **example.syscfg** ファイルをダブルクリックすると、SysConfig GUI が起動します。

注

syscfg ファイルを右クリックし、「Open With」→「SysConfig Editor」を選択します。

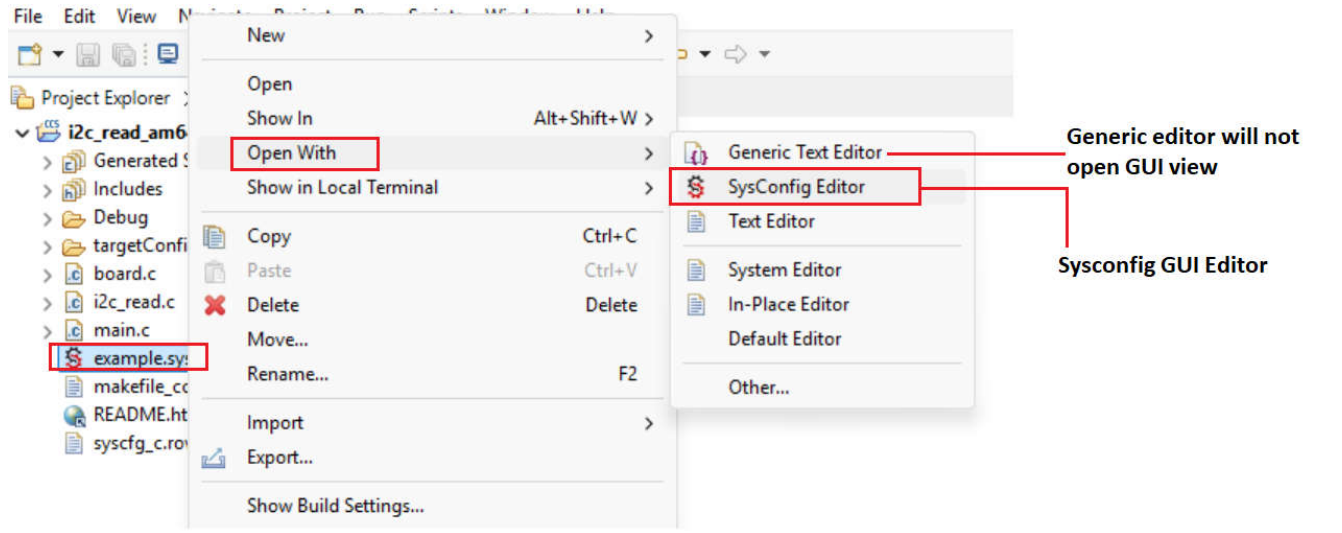

図 3-2. SysConfig CCS GUI エディタ

4. SysConfig GUI は CCS 内で起動する必要があります。表示例を図 3-3 に示します。

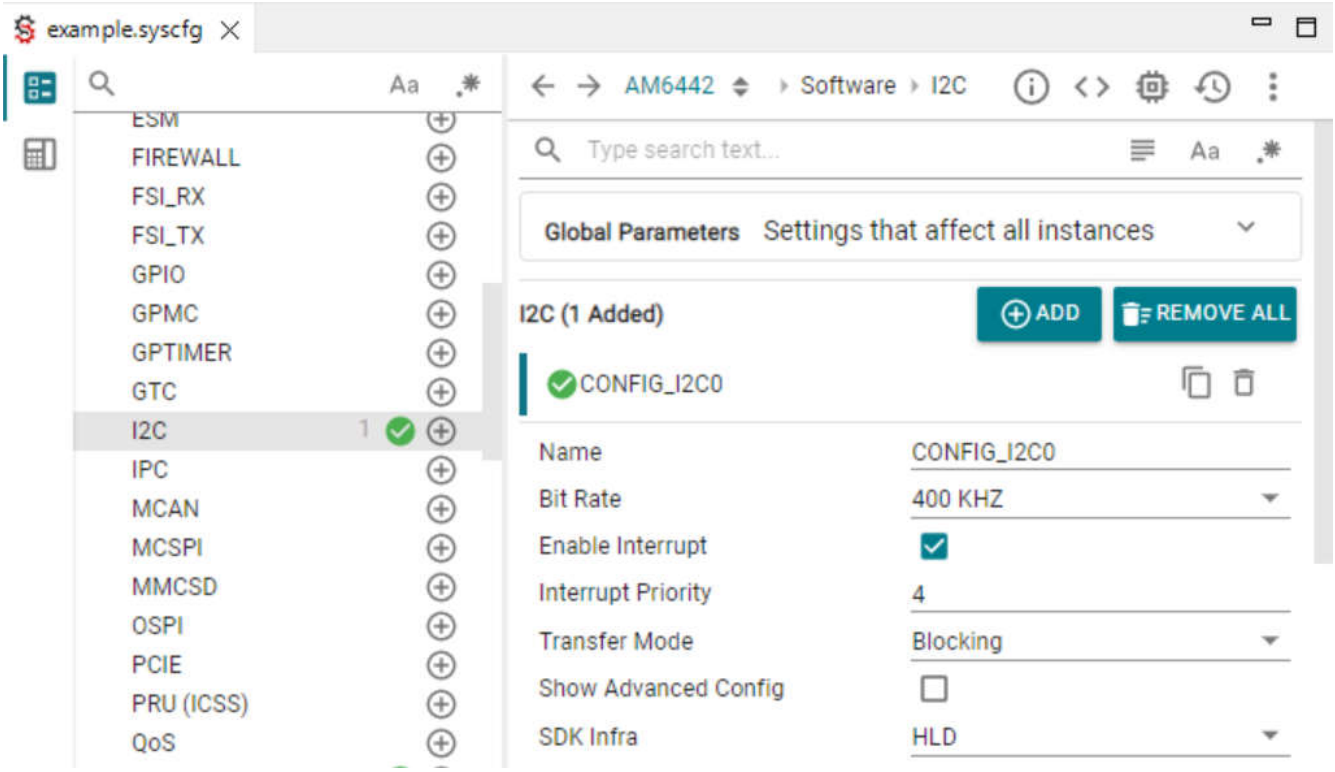

図 3-3. SysConfig CCS GUI ビュー

5. SysConfig GUI の右上隅にある **Device View** ボタンをクリックすると、プロジェクトで使用されているデバイスとパッケージを表示できます。

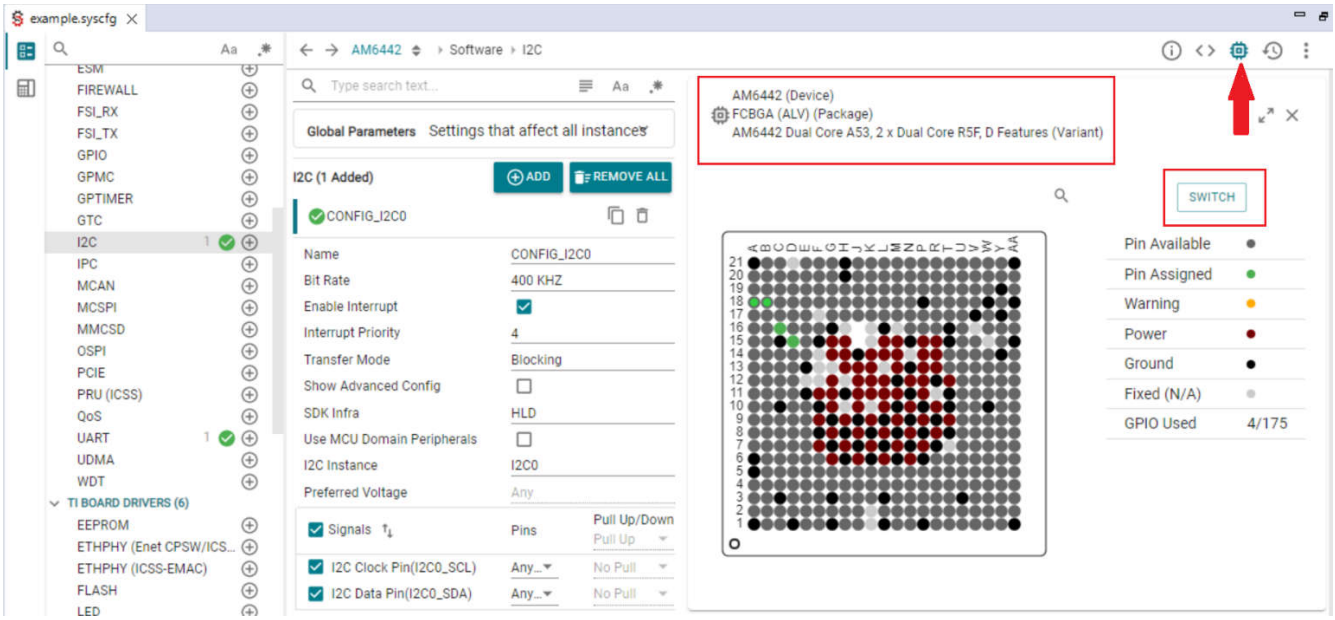

図 3-4. SysConfig デバイス ビュー

SysConfig のサポートがプロジェクト プロパティに追加されます。デフォルトでは、このプロジェクトは AM64x ファミリのデバイス用に構成されており、選択されたデバイス パッケージは FCBGA (ALV) パッケージに設定されています。CCS プロジェクトにおいてデフォルトで AM64x SysConfig サポートのプロジェクト プロパティが設定されていない場合、syscfg ファイルは GUI を正常に起動しません。

スタンドアロン バージョンの CodeGen ツール (CLI で開く) を使用する場合、モジュールの設定にも同じ手順が適用されます。

4 一般的なアプリケーションの構成

4.1 RAT の構成

RAT は、Region Based Address Translation (地域ベースのアドレス変換) の略称です。RAT モジュールは、32 ビットの入力アドレスを 48 ビットの出力アドレスに変換します。AM243x および AM6x デバイス ファミリでは、MPU サブ システムの R5F/M4F コアが 32 ビット メモリ アドレスのみにアクセスできます。この制限を克服し、SoC の全メモリ ビューにアクセスするため、RAT 領域は 32 ビットよりも高いメモリ領域にアクセスするように構成されています。

図 4-1 に、RAT 構成ビューを示します。

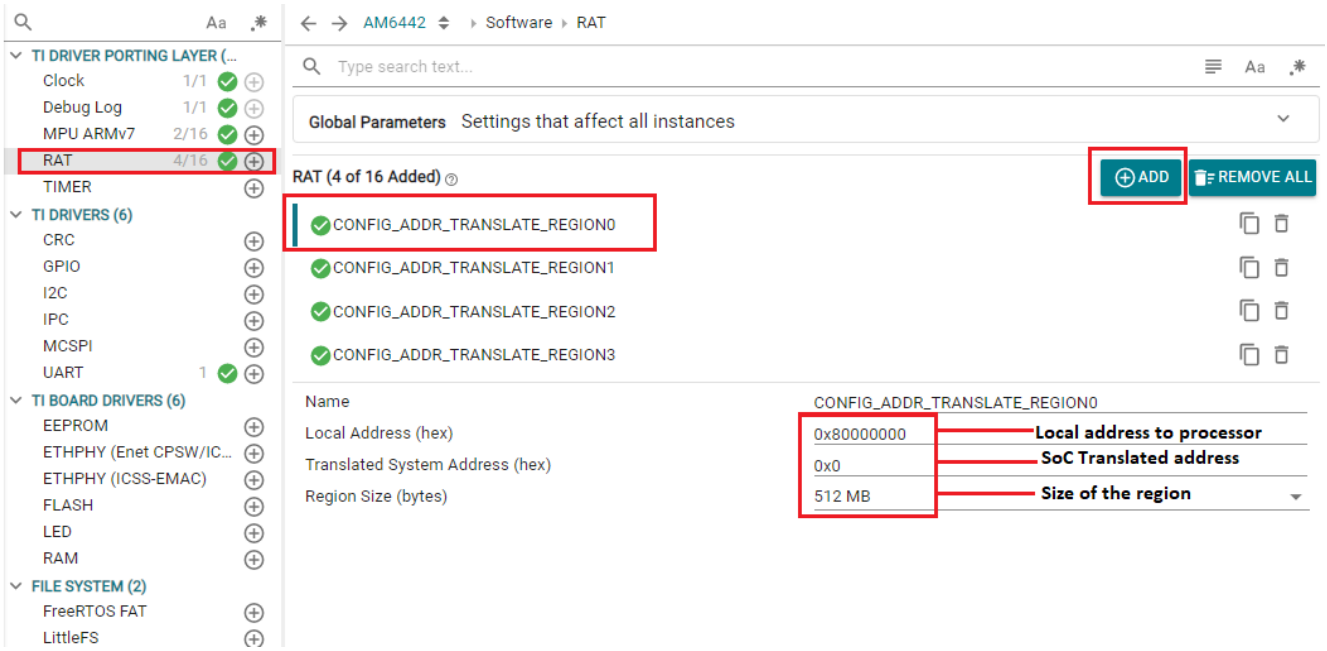

図 4-1. RAT の構成

4.2 MPU の構成

MPU は、メモリ保護ユニット (Memory Protection Unit) の略称です。MPU 構成を使用すると、メモリアクセス許可 (読み取り / 書き込み / 実行) をさまざまな権限レベルで構成できます。また、設定されたメモリ領域に格納できる属性 (キャッシュ可能 / 共有可能 / バッファ可能など) をユーザーが指定することもできます。

図 4-2 に、MPU 構成ビューを示します。

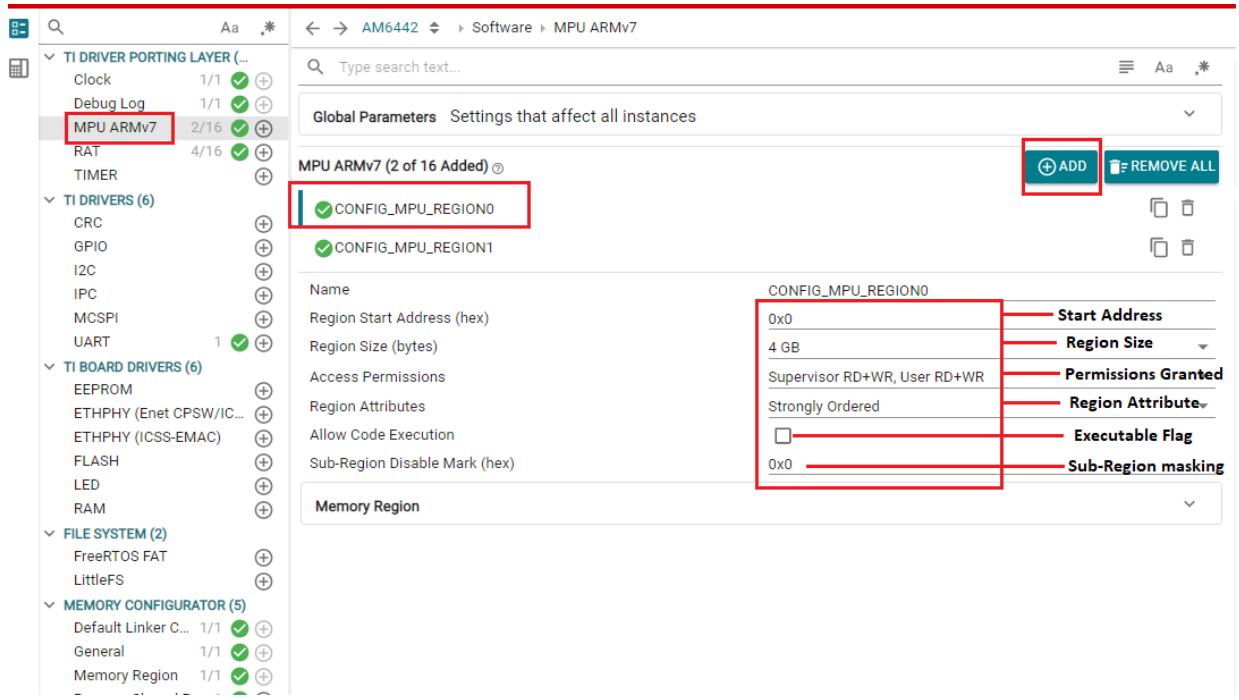

図 4-2. MPU の構成

4.3 MMU の構成

MMU は、メモリ管理ユニット (Memory Management Unit) の略称です。MMU 構成では、設定された領域の仮想メモリ変換、メモリ保護、およびキャッシュ管理が可能です。

MPU は R5F または M4F コアに適用でき、MMU は A53 コアに適用でき、仮想アドレス変換とキャッシュ管理をサポートします。

SysConfig では、選択したコア コンテキストに応じて、MPU と MMU のビューと構成を個別に表示できます。

図 4-3 に、MPU 構成ビューを示します。

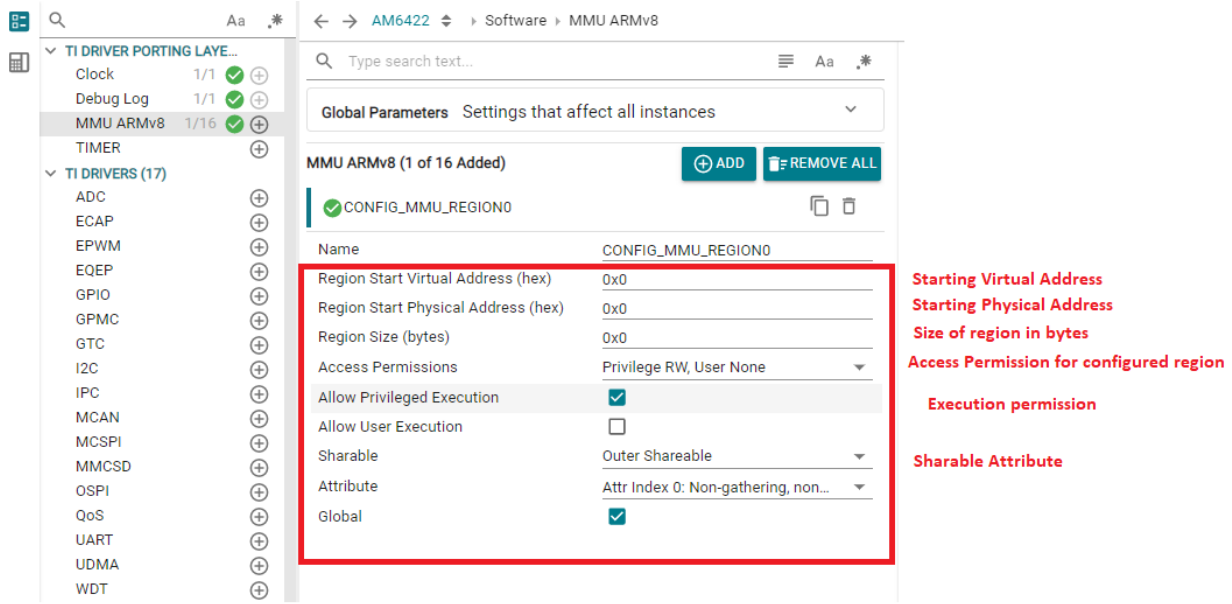

図 4-3. MMU の構成

4.4 システムの初期化

SysConfig CodeGen ツールは、クロックを有効にし、pinmux 設定を実行し、構成済みペリフェラルのドライバの初期化を行うためのコードを生成します。

図 4-4 に、初期化コードを示します。

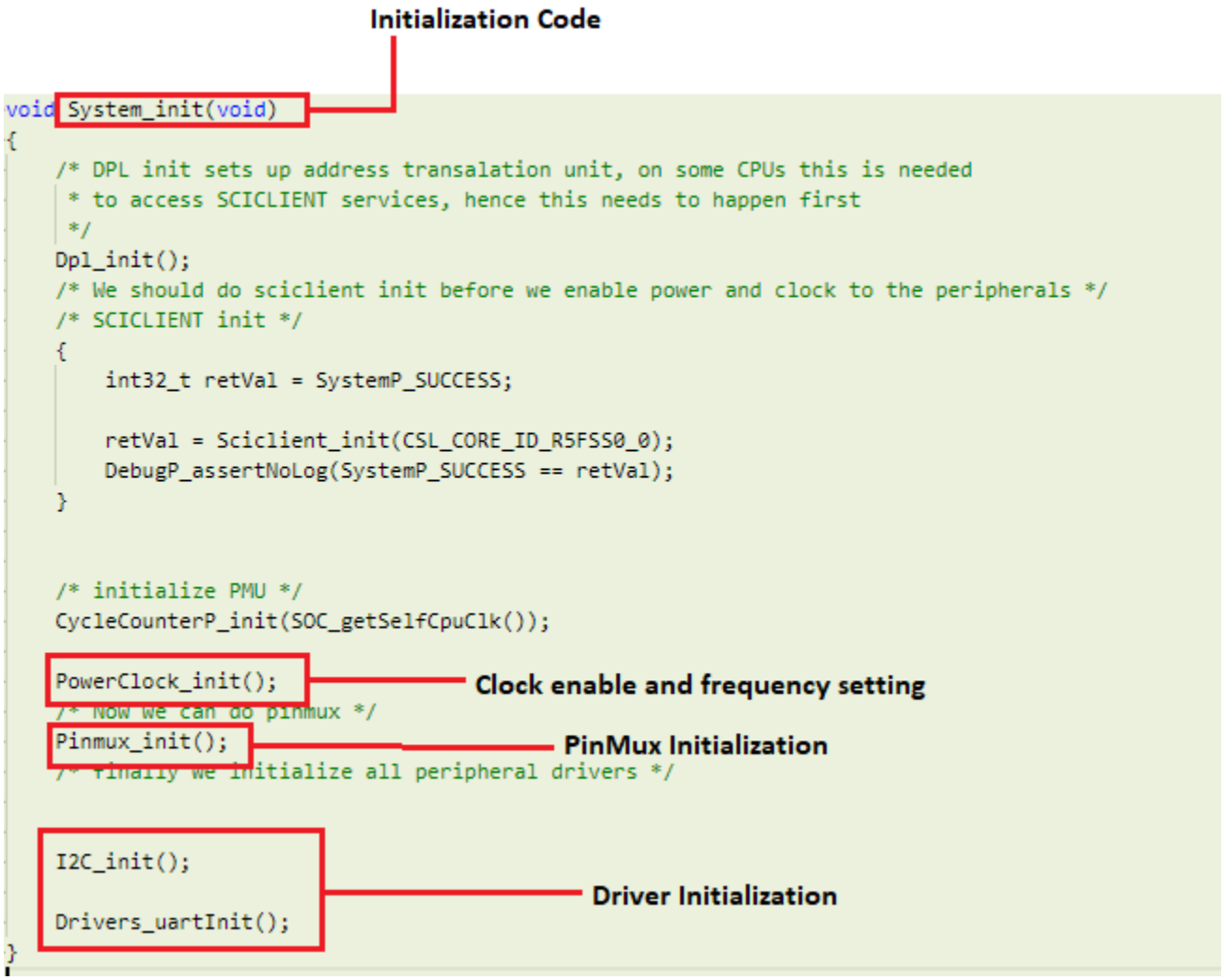

図 4-4. システムの初期化

以降のセクションでは、それぞれについて詳しく説明します。

4.4.1 DPL の初期化

CodeGen コードは、DPL を初期化するためのコードを生成します。DPL は、Driver Porting Layer (ドライバ ポーティング レイヤ) の略称です。DPL の初期化により、割り込みが有効になり、システム クロックが初期化され、タイマの初期化が行われます。

ti_dpl_config.c

```
144 144 void __mmu_init()
145 145 {
146 146     MmuP_init();
147 147     CacheP_enable(CacheP_TYPE_ALL);
148 148 }
149 149
150 150 void Dpl_init(void)
151 151 {
152 152     /* initialize Hwi but keep interrupts disabled */
153 153     HwiP_init();
154 154
155 155     /* init debug log zones early */
156 156 #if defined(SMP_FREERTOS)
157 157     /* Initialize Debug module from Core0 only */
158 158     if(0 == Armv8_getCoreId())
159 159     {
160 160
161 161     /* Debug log init */
162 162     DebugP_logZoneEnable(DebugP_LOG_ZONE_ERROR);
163 163     DebugP_logZoneEnable(DebugP_LOG_ZONE_WARN);
164 164
165 165     }
166 166 #else
167 167     /* Debug log init */
168 168     DebugP_logZoneEnable(DebugP_LOG_ZONE_ERROR);
169 169     DebugP_logZoneEnable(DebugP_LOG_ZONE_WARN);
170 170 #endif
171 171
172 172 #if defined(SMP_FREERTOS)
173 173     /* Initialize Clock from Core0 only */
174 174     if(0 == Armv8_getCoreId())
175 175     {
176 176
177 177     /* initialize Clock */
178 178     ClockP_init();
179 179
180 180
181 181     }
182 182 #else
183 183
184 184     /* initialize Clock */
185 185     ClockP_init();
186 186
187 187 #endif
188 188
189 189     TimerP_init();
190 190
191 191
192 192 }
193 193
```

図 4-5. DPL の初期化

4.4.2 クロック初期化

アプリケーションでペリフェラルを使用する前に、それぞれのクロック設定を有効化および構成して、ペリフェラルを適切に初期化する必要があります。TISCI API を呼び出すことで、クロックが初期化されます。クロックの構成に必要なこのコードは、ペリフェラルを追加するとツールによって自動生成されます。

ユーザーが CodeGen ツールにモジュール / ペリフェラルを追加すると、構成されたクロック構成は、TISCI API が周波数を設定するために使用する構造体 (gSocModulesClockFrequency など) に自動的に入力されます。Module_clockEnable() API は、LPSC ゲートをクロック用に有効化することで、モジュールの電力構成を実行します。

図 4-6 に、ペリフェラル用のクロックを有効化および構成するためにツールによって生成されるコードを示します。

```
typedef struct {

    uint32_t moduleId;
    uint32_t clkId;
    uint32_t clkRate;
    uint32_t clkParentId;

} SOC_ModuleClockFrequency;

uint32_t gSocModules[] = {
    TISCI_DEV_MCU_UART0,

    SOC_MODULES_END,
};

SOC_ModuleClockFrequency gSocModulesClockFrequency[] = {
    { TISCI_DEV_MCU_UART0, TISCI_DEV_MCU_UART0_FCLK_CLK, 48000000, SOC_MODULES_END},

    { SOC_MODULES_END, SOC_MODULES_END, SOC_MODULES_END, SOC_MODULES_END },
};

void Module_clockEnable(void)
{
    int32_t status;
    uint32_t i = 0;

    while(gSocModules[i]!=SOC_MODULES_END)
    {
        status = SOC_moduleClockEnable(gSocModules[i], 1);
        DebugP_assertNoLog(status == SystemP_SUCCESS);
        i++;
    }
}
```

```
void Module_clockSetFrequency(void)
{
    int32_t status;
    uint32_t i = 0;

    while(gSocModulesClockFrequency[i].moduleId!=SOC_MODULES_END)
    {
        if (gSocModulesClockFrequency[i].clkParentId != SOC_MODULES_END)
        {
            /* Set module clock to specified frequency and with a specific par
            status = SOC_moduleSetClockFrequencyWithParent(
                        gSocModulesClockFrequency[i].moduleId,
                        gSocModulesClockFrequency[i].clkId,
                        gSocModulesClockFrequency[i].clkParentId,
                        gSocModulesClockFrequency[i].clkRate
                        );
        }
        else
        {
            /* Set module clock to specified frequency */
            status = SOC_moduleSetClockFrequency(
                        gSocModulesClockFrequency[i].moduleId,
                        gSocModulesClockFrequency[i].clkId,
                        gSocModulesClockFrequency[i].clkRate
                        );
        }
        DebugP_assertNoLog(status == SystemP_SUCCESS);
        i++;
    }
}


void PowerClock_init(void)
{
    Module_clockEnable();
    Module_clockSetFrequency();
}
```

図 4-6. クロック構成

4.4.3 PinMux 構成

AM243x および AM6x ファミリのデバイスは、複数のペリフェラル (UART、SPI、I2C、GPIO など) 間で限られたピンを共有します。ピン マルチプレクシング (PinMux) は、どのペリフェラルがどの物理ボールおよびピンに接続されているかを選択します。PinMux を正しく設定しない場合、ペリフェラルが外部デバイスと通信できなくなります。競合するアサインメント (たとえば、UART と I2C が同じピンを共有している) が発生すると、ブートまたはランタイム障害が発生します。CodeGen ツールを使用すると、これらの競合を簡単に回避できます。

CodeGen ツールにモジュールとペリフェラルを追加すると、モジュールに必要なピンが SysConfig ツールによって公開され、対応するコードが生成されます。また、ユーザーはこのツールを使用して、入力を有効化 / 無効化するピン設定を選択することも、プルアップ / ダウンするようにピンを構成することもできます。

MCU と MAIN ドメイン ペリフェラル用に構成された、独立した構造またはピンセットがあります。

図 4-7 に、Pinmux 初期化が生成したコードを示します。

```c
#include "ti_drivers_config.h"
#include <drivers/pinmux.h>

static Pinmux_PerCfg_t gPinMuxMainDomainCfg[] = {

    {PINMUX_END, PINMUX_END}
};

static Pinmux_PerCfg_t gPinMuxMcuDomainCfg[] = {
            /* MCU_USART0 pin config */
    /* MCU_UART0_RXD -> MCU_UART0_RXD (A9) */
    {
        PIN_MCU_UART0_RXD,
        ( PIN_MODE(0) | PIN_INPUT_ENABLE | PIN_PULL_DISABLE )
    },
    /* MCU_USART0 pin config */
    /* MCU_UART0_TXD -> MCU_UART0_TXD (A8) */
    {
        PIN_MCU_UART0_TXD,
        ( PIN_MODE(0) | PIN_PULL_DISABLE )
    },

    {PINMUX_END, PINMUX_END}
};

/*
 * Pinmux
 */


void Pinmux_init(void)
{



    Pinmux_config(gPinMuxMainDomainCfg, PINMUX_DOMAIN_ID_MAIN);

    Pinmux_config(gPinMuxMcuDomainCfg, PINMUX_DOMAIN_ID_MCU);
}
```

図 4-7. PinMux の初期化

4.4.4 ドライバの初期化

アプリケーションでペリフェラルを使用するには、ドライバを初期化して適切に機能させる必要があります。SysConfig CodeGen ツールは、構成済みペリフェラル用のドライバ構成用コードを生成します。

このツールは、実際のドライバ初期化コードのラッパー関数として Drivers_Init/Deinit()、Drivers_Open/Close()API を使用します。ドライバ初期化コードは SDK のドライバから取得されたものであり、CodeGen ツールによって自動生成されることはありません。

CodeGen ツールは、必要な構造体に設定済みの値を入力します。入力された構造体は、SDK のドライバ API で使用されます。

```
UART_DmaChConfig gUartDmaChConfig[CONFIG_UART_NUM_INSTANCES] =
{
                NULL,
};

/* UART Driver Parameters */
UART_Params gUartParams[CONFIG_UART_NUM_INSTANCES] =
{
        {
            .baudRate           = 115200,
            .dataLength         = UART_LEN_8,
            .stopBits           = UART_STOPBITS_1,
            .parityType         = UART_PARITY_NONE,
            .readMode           = UART_TRANSFER_MODE_BLOCKING,
            .readReturnMode     = UART_READ_RETURN_MODE_FULL,
            .writeMode          = UART_TRANSFER_MODE_BLOCKING,
            .readCallbackFxn    = NULL,
            .writeCallbackFxn   = NULL,
            .hwFlowControl      = FALSE,
            .hwFlowControlThr   = UART_RXTRIGLVL_16,
            .transferMode       = UART_CONFIG_MODE_INTERRUPT,
            .skipIntrReg         = FALSE,
            .uartDmaIndex = -1,
            .intrNum            = 211U,
            .intrPriority       = 4U,
            .operMode           = UART_OPER_MODE_16X,
            .rxTrigLvl          = UART_RXTRIGLVL_8,
            .txTrigLvl          = UART_TXTRIGLVL_32,
            .rxEvtNum           = 0U,
            .txEvtNum           = 0U,
        },
};

void Drivers_uartOpen(void)
{
    uint32_t instCnt;
    int32_t  status = SystemP_SUCCESS;

    for(instCnt = 0U; instCnt < CONFIG_UART_NUM_INSTANCES; instCnt++)
    {
        gUartHandle[instCnt] = NULL;    /* Init to NULL so that we can exit gracefully */
    }

    /* Open all instances */
    for(instCnt = 0U; instCnt < CONFIG_UART_NUM_INSTANCES; instCnt++)
    {
        gUartHandle[instCnt] = UART_open(instCnt, &gUartParams[instCnt]);
        if(NULL == gUartHandle[instCnt])
        {
            DebugP_logError("UART open failed for instance %d !!!\r\n", instCnt);
            status = SystemP_FAILURE;
            break;
        }
    }
```

図 4-8. Driver Configuration (デバイス構成)

4.4.5 ボード ペリフェラルの初期化

CodeGen ツールは、ボード ペリフェラルの初期化向けのソース ファイルも生成します。このファイルには、構成済みボード ドライバの初期化を実行するための API の定義が含まれています。また、このファイルは、ドライバを開いたり閉じたりするための API も提供します。

生成される API の定義は、**ti_board_open_close.c** ファイルにあります。

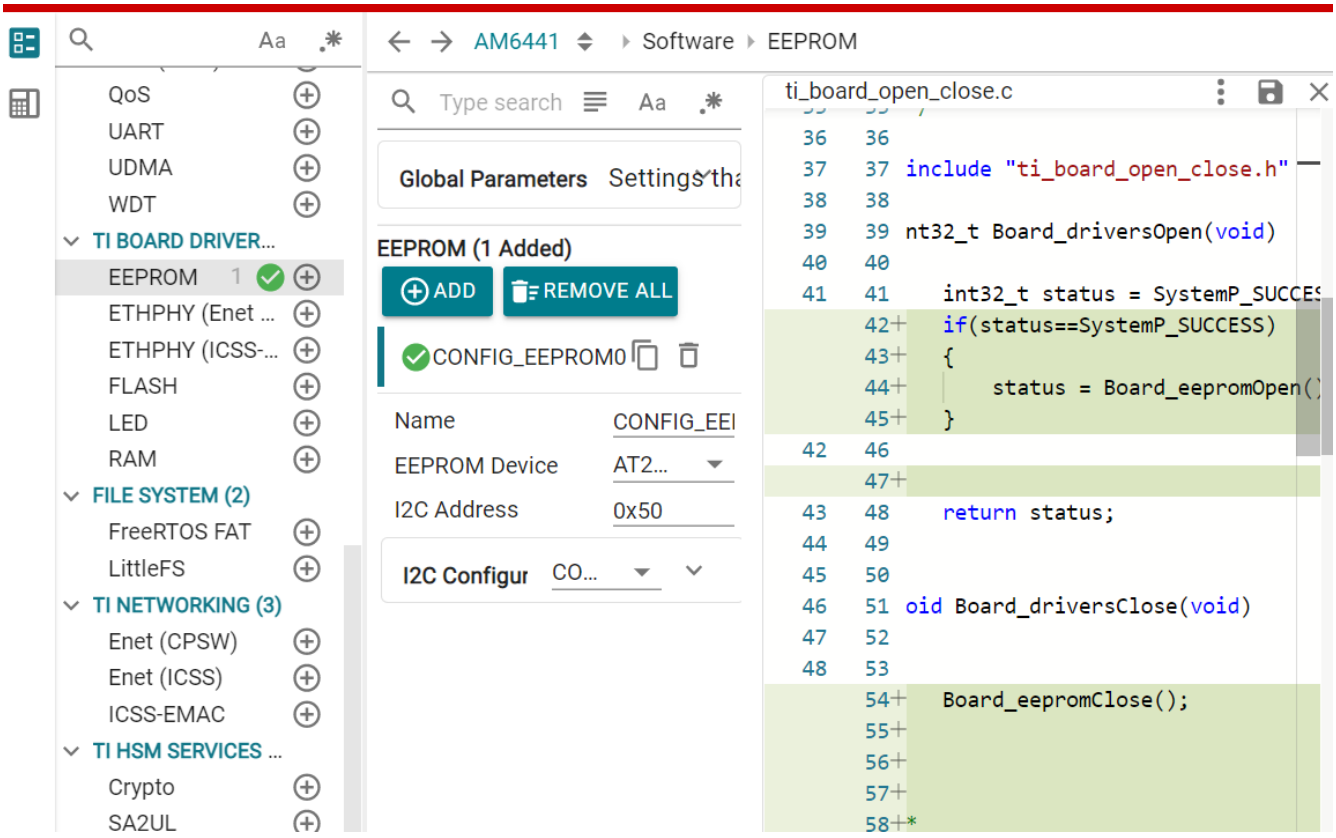

図 4-9. ボード ペリフェラル ドライバ

5 出力ファイル

5.1 CodeGen ツールによって生成されたファイル

SysConfig 向けの CodeGen ツールは、C ソース ファイルとヘッダー ファイルを生成します。これらのファイルをアプリケーション開発で使用することで、エラーを防止し、生産性を向上させることができます。この生成済みファイルは、ドライバの初期化と構成用のアプリケーションに直接インポートすることができます。

生成されたファイルを以下に示します。

1. *ti_dpl_config.h* には、DPL (Driver Porting Layer) 初期化 API の宣言が含まれています。
2. *ti_dpl_config.c* には、DPL を初期化するためのコードが含まれています。DPL の初期化には、割り込みコントローラ、MMU および RAT 構成、デバッグ ログ、システム タイマの初期化が含まれます。DPL の初期化は、生成されたコードによるカーネル レベル API 呼び出しを使用して行われます。
3. *ti_drivers_config.h* には、設定されたすべてのドライバのドライバ初期化 API の宣言が含まれています。
4. *ti_drivers_config.c* には、構成済みのペリフェラル ドライバの初期化、クロックの初期化、PinMux 設定、ドライバの初期化に使用するコードが含まれています。このファイルには、構成されたペリフェラル用のグローバル ハンドルも含まれています。
5. *ti_drivers_open_close.h* には、構成済みのペリフェラルに対するドライバ open/close API の宣言と必要なハンドラが含まれています。
6. *ti_drivers_open_close.c* には、構成済みのペリフェラル用にドライバを開く / 閉じるためのコードが含まれています。このファイルには、追加されたペリフェラルに必要な構成済みのパラメータを持つハンドラも含まれています。
7. *ti_pinmux_config.c* には、GUI 経由で構成された必要な機能を実現するために、構成済みのペリフェラルが必要とするピンマルチプレクサ構成が含まれています。
8. *ti_power_clock_config.c* には、クロックを有効にし、構成済みのペリフェラルのクロック周波数を変更するためのコードが含まれています。生成されたコードでは、TISCI 呼び出しを使用してクロック周波数を構成します。
9. *ti_board_config.h* には、ボード固有のドライバ構成に関する宣言が含まれています。
10. *ti_board_config.c* には、ボード固有のドライバ構成の定義が含まれています。
11. *ti_board_open_close.h* には、ボード固有のドライバ open/close API の宣言が含まれています。
12. *ti_board_open_close.c* には、ボード固有のドライバ open/close API の定義が含まれています。
13. *ti_enet_config.h* には、enet モジュールで使用されるすべてのマクロの定義が含まれています。
14. *ti_enet_config.c* には、グローバル構造の定義と、enet 機能を提供するために必要な API が含まれています。
15. *ti_enet_open_close.h* には、enet open/close API の宣言と必要なユーティリティ API が含まれています。
16. *ti_enet_open_close.c* には、enet open/close API の定義と必要な構造定義が含まれます。
17. *ti_enet_soc.c* には、enet 割り込みのセットアップ、クロック周波数の設定、他の必要な構成の設定 / 取得に必要な構造と API の定義が含まれています。
18. *ti_enet_lwipif.h* には、ドライバ コールバック用の enet Lwip インターフェイス レイヤの宣言が含まれています。
19. *ti_enet_lwipif.c* には、ドライバ コールバック用の enet Lwip インターフェイス レイヤの実装を含んでいます。

5.1.1 デバッグとトラブルシューティング

SysConfig アプリケーションを使用するときに、誤った cliArgs 引数が渡されると、SysConfig ツールはエラー メッセージを報告します。エラー メッセージを確認することで、エラーの原因を特定できます。cliArgs エラーとは別に、ツールの使用中に発生する可能性のある他の問題があります。

以下のセクションでは、SysConfig を使用しているときに表示されるいくつかの一般的な問題について説明します。それらの修正手順を参照してください。

5.2 バージョンの不一致

図 5-1 では、MCU SDK バージョンと SysConfig ツールのバージョンが不一致のため、「*Update Required*」というエラー メッセージが表示されます。syscfg ファイルで使用されている cliArgs が正しくないため、GUI ビューを開くときにエラーが報告されます。

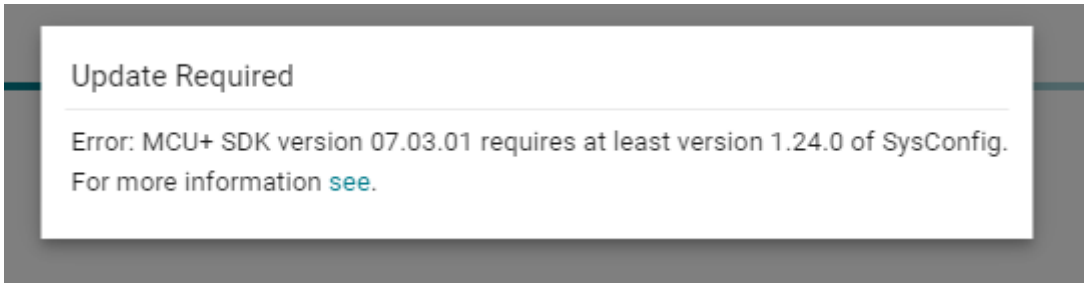

図 5-1. バージョンの不一致

上記の問題を解決するには、MCU SDK の資料で説明しているように、適切なバージョンの SysConfig が使用されていることを確認してください。バージョンの詳細については、MCU SDK に付属されている **product.json** ファイルを確認してください。

前の例では、SysConfig v1.23.0 と MCU+SDK v11.00 が使用されます。example.syscfg ファイルで使用されている cliArgs は次のとおりであり、MCU SDK のバージョンが正しくありません。

```
/** * These arguments were used when this file was generated. They will be automatically applied on
subsequent loads * via the GUI or CLI. Run CLI with '--help' for additional information on how to
override these arguments. *
                @cliArgs --device "AM64x" --part "Default" --package "ALV" --context "r5fss0-0" --
product "MCU_PLUS_SDK@07.03.01" * @v2CliArgs --device "AM6442" --package "FCBGA (ALV)" --variant
"AM6442-D" --context "r5fss0-0" --product
                "MCU_PLUS_SDK@07.03.01" * @versions {"tool":"1.21.2+3837"} */
```

example.syscfg ファイルで上記の cliArgs を変更し、MCU SDK の正しいバージョンに修正すると、このツールは期待どおりに動作します。

```
/** * These arguments were used when this file was generated. They will be automatically applied on
subsequent loads * via the GUI or CLI. Run CLI with '--help' for additional information on how to
override these arguments. *
                @cliArgs --device "AM64x" --part "Default" --package "ALV" --context "r5fss0-0" --
product "MCU_PLUS_SDK_AM64x@11.00.00" * @v2CliArgs --device "AM6442" --package "FCBGA (ALV)" --
variant "AM6442-D" --context "r5fss0-0" --product
                "MCU_PLUS_SDK_AM64x@11.00.00" * @versions {"tool":"1.21.2+3837"} */
```

その他の問題では、*「デバイスのバリエーションが見つかりません」*、*「パッケージ / 型番が見つかりません」*などが表示される場合があります。詳細は、図 5-2、図 5-3、および図 5-4 を参照してください。

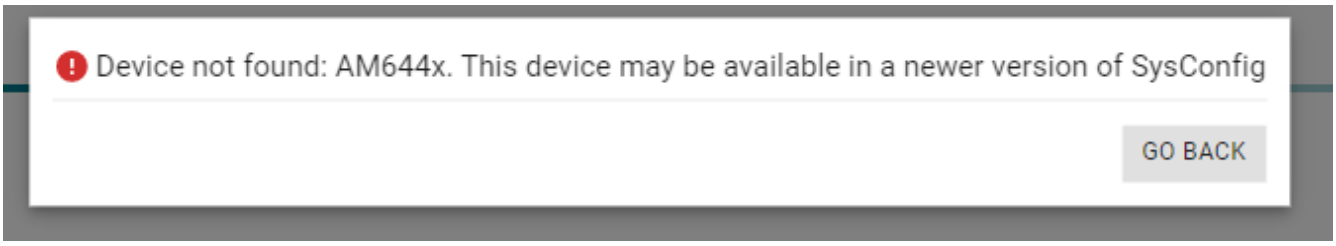

図 5-2. デバイスが見つかりません

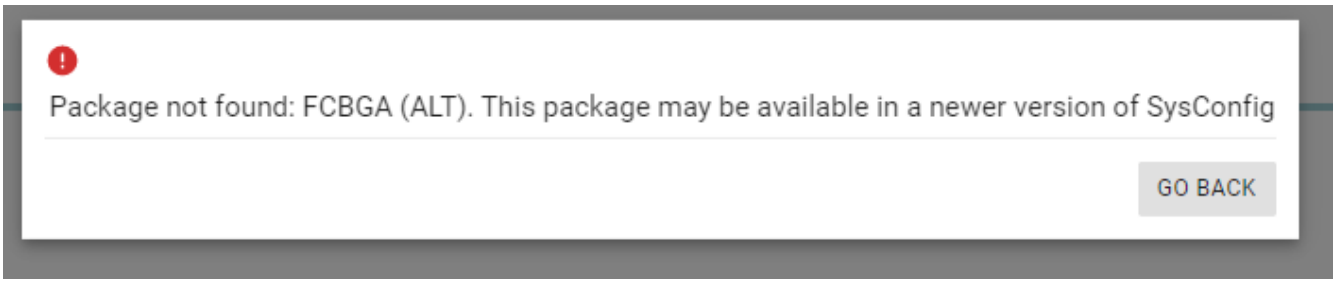

図 5-3. パッケージが見つかりません

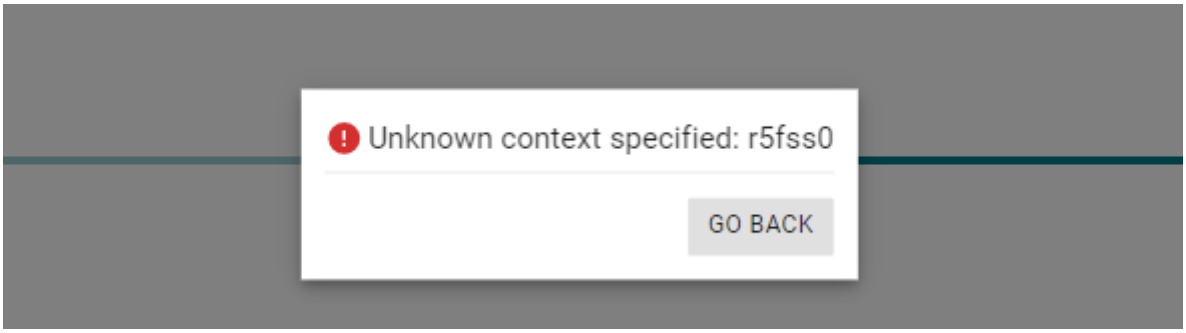

図 5-4. 不明なコンテキストが指定されました

上記のすべてのパラメータは、**example.syscfg** ファイルの cliArgs で正しく渡されなければなりません。間違った引数を渡すと、上記の問題のいずれかが発生します。

cliArgs でどのパラメータを使用するかわからない場合は、ソフトウェア製品として MCU SDK を選択した状態で CodeGen ツールを開き、生成された **untitled.syscfg** ファイルから cliArgs をコピーします。

図 5-5. SysConfig CodeGen cliArgs

5.3 リソースの競合

開発に CodeGen ツールを使用すると、競合を特定して解決することが簡単です。CodeGen ツールは、ユーザーが手動設定を行う際に発生する可能性のあるさまざまな種類の競合を検出し、ツールは同時にエラーメッセージをポップアップで表示します。以降のセクションでは、ツールを使用して識別および解決できる競合の種類について説明します。

5.3.1 ピンの競合

ピンのいずれかが複数の機能に構成されている場合、SysConfig ツールはエラーを報告します。

たとえば、ユーザーが GPIO ピン (ボール T20) と同じピン (ボール T20) を GPMC に構成している場合です。SysConfig ツールは、T20 ピンが複数の機能向けに構成されているため、リソースの競合の問題を報告します。

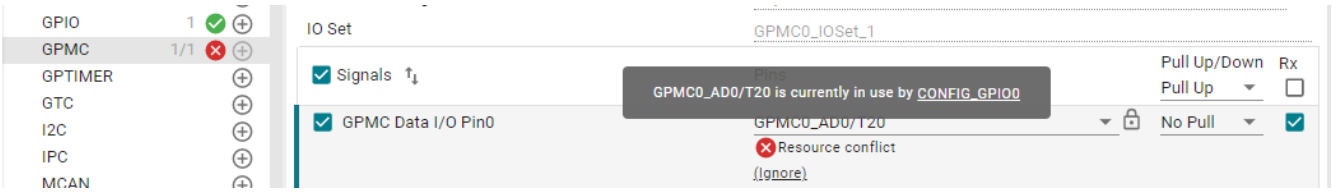

図 5-6. ピンの競合

GPIO または GPMC ペリフェラルから T20 ピンを取り除くことで、この問題を簡単に解決できます。

5.3.2 モジュール インスタンスの競合

特定のモジュールの下にあるインスタンスの 1 つが複数回構成されている場合、SysConfig ツールはエラーを報告します。

たとえば、ユーザーが UART モジュールを構成し、UART の 2 つのインスタンスを追加した場合です。UART モジュールの下に 2 つ以上のインスタンスが同じ UART (UART0 など) ペリフェラルを構成しようとすると、このツールはインスタンスの競合エラーを報告します。

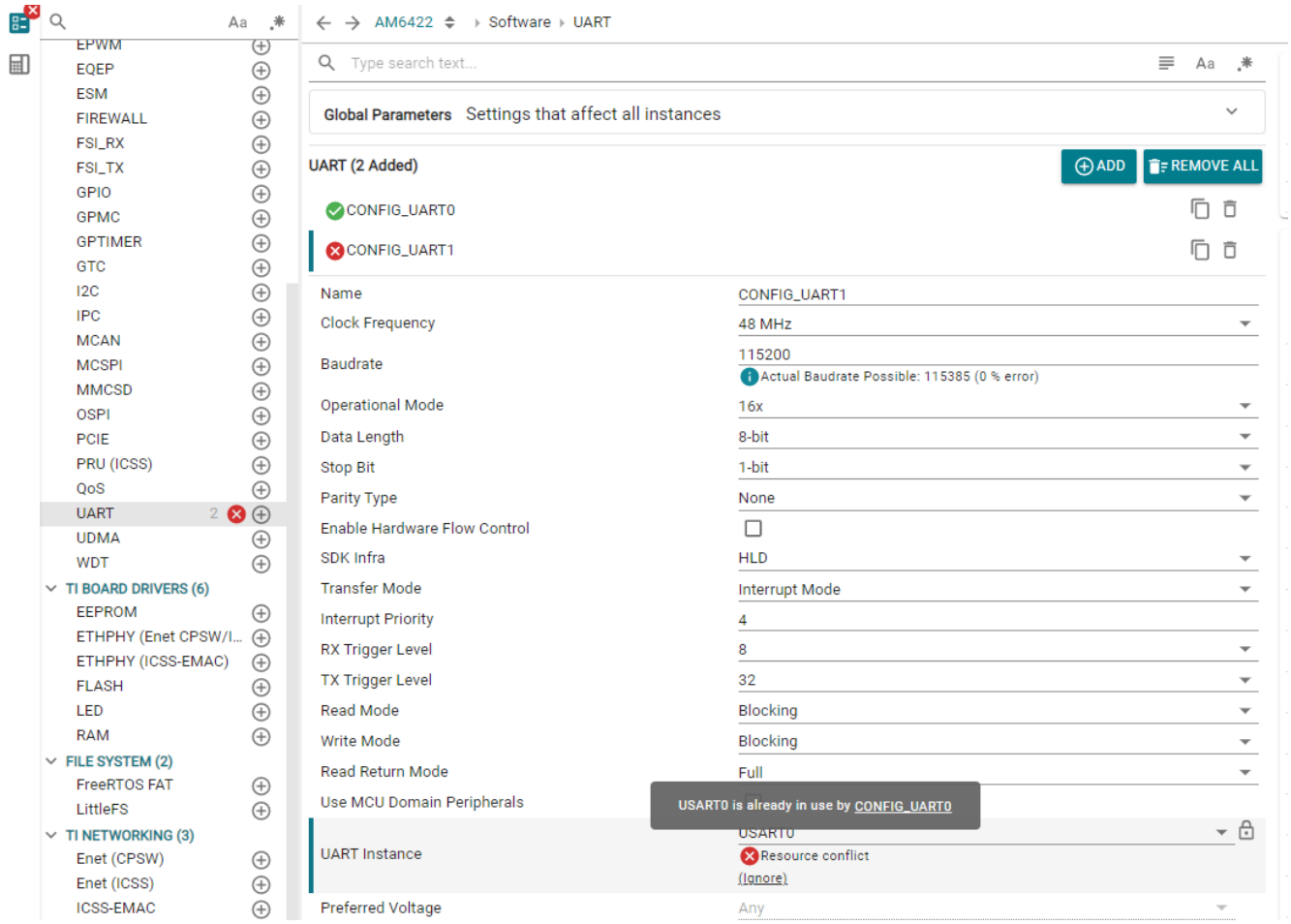

図 5-7. インスタンスの競合

この問題は、異なるインスタンスに異なる UART を構成することで簡単に解決できます。

5.3.3 マルチコア リソースの競合

マルチコア プロジェクトを使用する場合、同じリソースを 2 つの異なるコアに構成できます。このようなシナリオでは、ツールはコアとポップアップ エラー メッセージ間の競合の原因を自動的に検出します。

たとえば、GPIO ピンが R5F0-0 コア用に構成されており、マルチコア プロジェクトの R5F0-1 コア用に同じピンが再度構成されている場合、ツールによってリソース競合エラーが発生します。

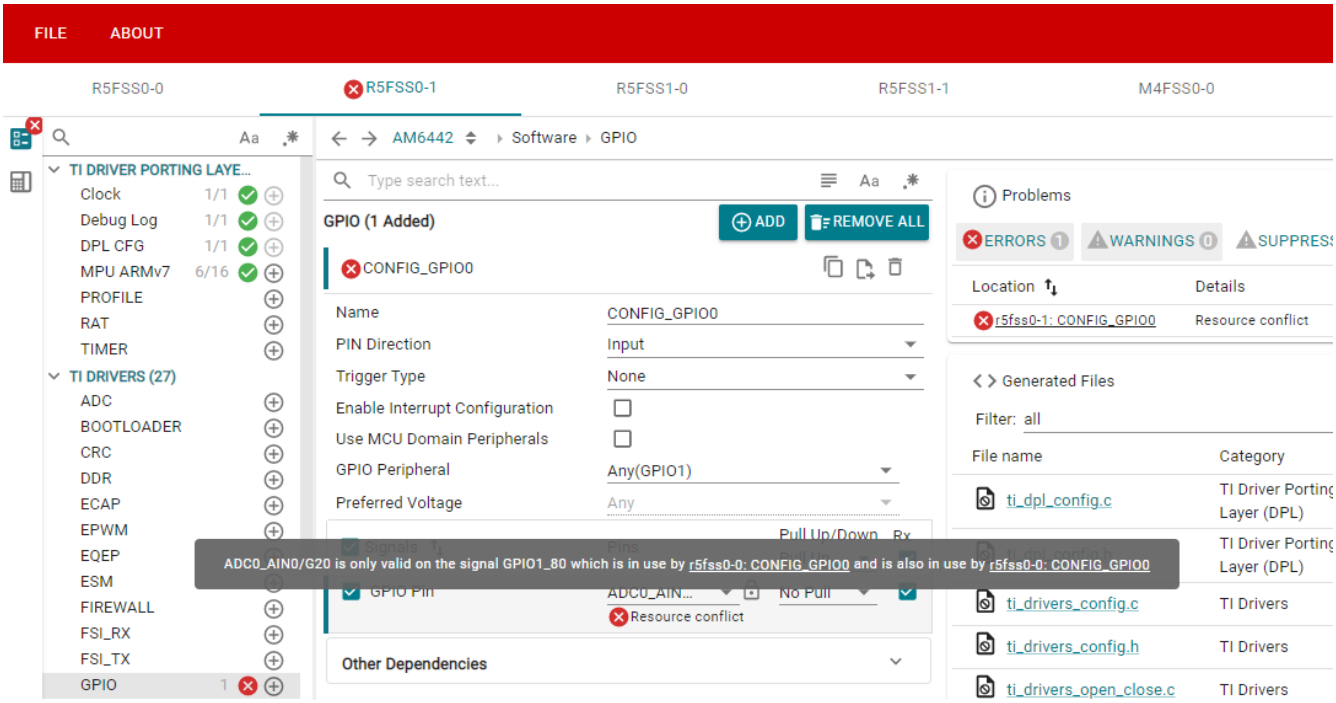

図 5-8. マルチコア リソースの競合

この問題は、異なるコア用に異なる GPIO ピンを構成することで簡単に解決できます。

5.4 サポートされていないドライバ

以下の例では、アプリケーションで OSPI ドライバを使用する必要がありますが、ツールの *TI ドライバ*リストには OSPI モジュールがありません。詳しくは、図 5-9 を参照してください。

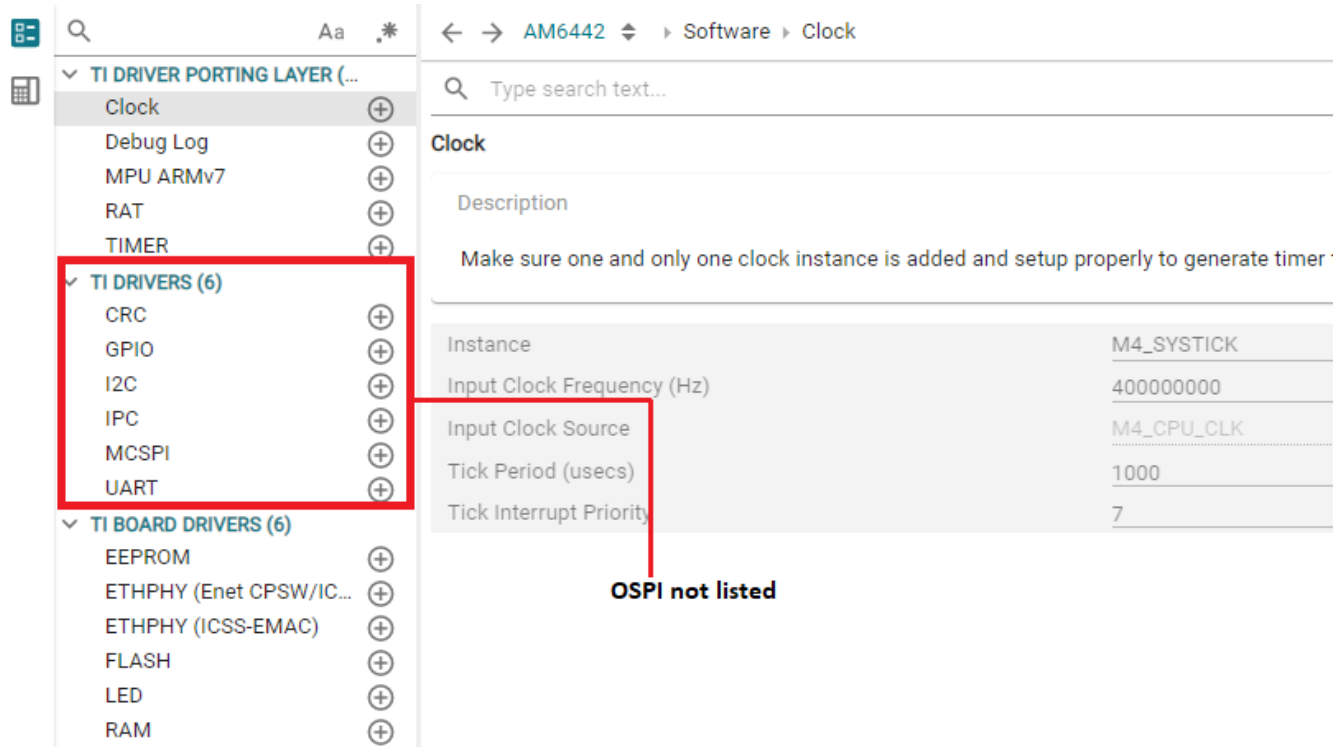

図 5-9. TI ドライバ

ツールの *TI ドライバ* セクションに表示されていないドライバは、MCU SDK ではサポートされていません。ツールのドライバ リストは、コアのさまざまな組み合わせに応じて変更できます。

サポート対象ドライバ リストの詳細については、MCU SDK のリリース ノートを参照してください。

5.5 予約済みペリフェラルの使用

「Reserved Peripherals」タブは、カスタム コードが使用できるハードウェア リソースを予約するために使用され、このタブには SysConfig ツールにそのペリフェラルを使用しないよう表示されます。SysConfig ツールは、予約済みペリフェラルの下に構成済みペリフェラル用のコードを生成しません。このタブは、ツールで設定する必要のある周辺機器には使用しないでください。

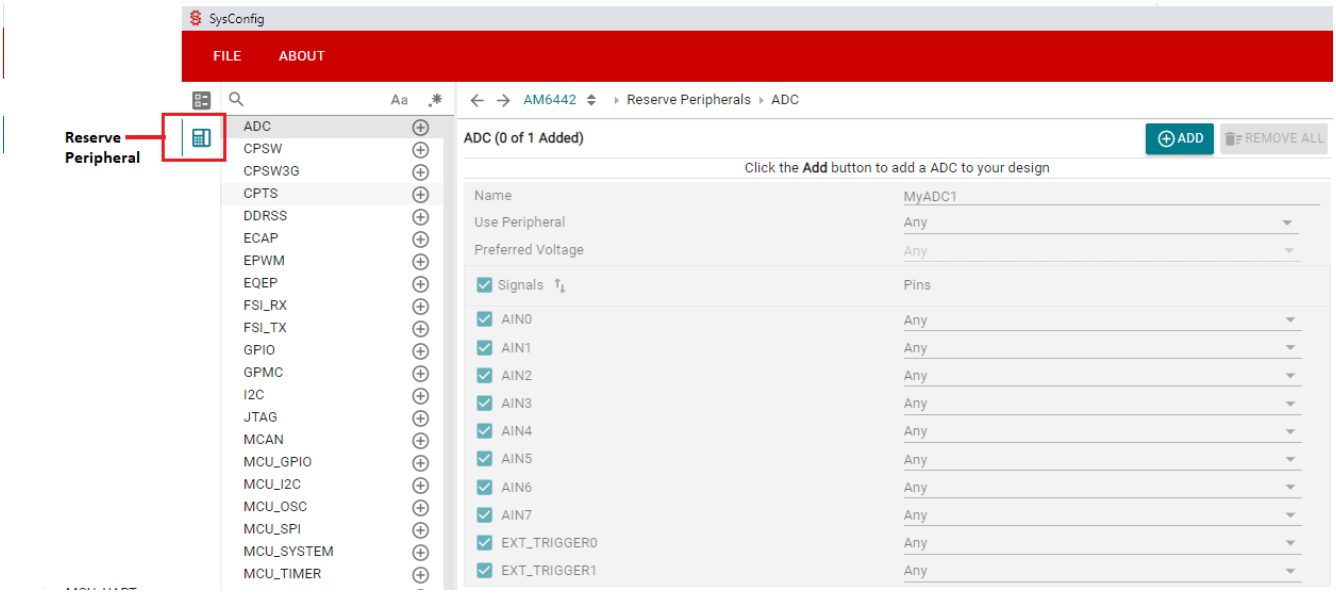

図 5-10. SysConfig Reserve Peripheral タブ

6 免責事項と使用目的

SysConfig コア ツールは、TI のベースライン品質開発プロセスに準拠しています。つまり、SysConfig を使用して生成したコードに対して、車載または機能安全の要求を実施できないことを意味します。特定の規格の要件に従って、生成されたコードの標準的な適格性確認を実行するのは、お客様の責任であることが期待されています。

7 まとめ

SysConfig により、TI の SoC 用の初期化コードと構成コードを自動生成することで、ソフトウェアの開発期間が大幅に短縮されます。

GUI および CLI インターフェイスは、手動作業を最小限に抑え、構成の一貫性を検証し、マルチコア プロジェクト全体の生産性を向上させます。

SysConfig を MCU+SDK ワークフローに統合することで、開発者は構成エラーのリスクを低減し、組込みシステムのプロトタイプ製作とスケール化を迅速に実施できます。

8 参考資料

- TI クラウド ツール
  - SysConfig
  - Resource Explorer

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