注意: この日本語版文書は参考資料としてご利用ください。
最新情報は必ずオリジナルの英語版をご参照願います。

PIC32CZ CAxマイクロコントローラでMPLAB® Harmony v3とMCC (MPLAB Code Configurator)を使って初めてのアプリケーションを作成する方法

TB3348

はじめに

MPLAB® Harmony v3は相互に互換で相互運用可能なモジュールで構成されたソフトウェア開発フレームワークであり、PLIB(周辺モジュール ライブラリ)、ドライバ、システムサービス、ミドルウェア、サードパーティ ライブラリを含んでいます。MCC (MPLAB Code Configurator)はGUIベースのツールであり、各種MPLAB Harmonyモジュールを有効化して簡単に設定できます。MCCはMPLAB X IDE(統合開発環境)のプラグインです。

本書では、Arm® Cortex®-M7ベースのPIC32CZ CAxマイクロコントローラでMCCとMPLAB Harmony v3モジュールを使ってシンプルなアプリケーションを作成する方法を説明します。このアプリケーションの目的は、LEDをタイムアウト ベースでトグルさせ、LEDトグルレートをシリアル コンソールに表示する事です。このデモではMCCを使って以下のMPLAB Harmony v3モジュールを設定します。

- LED0をトグルさせるPORTピン
- LED0のトグルレートを周期的にサンプリングするRTC(リアルタイム クロック)PLIB
- スイッチ押し下げイベントが発生した時にトグルレートを変更するEIC(外部割り込みコントローラ)PLIB
- PC上で実行されているCOM(シリアル) ポートターミナル アプリケーションにLED0のトグルレートを出力する、SERCOM (USARTとして設定されているシリアル通信インターフェイス)PLIBおよびDMA(ダイレクト メモリアクセス)PLIB
- シリアル ターミナルと通信するポートピン(USARTピン。値を取得した後にターミナルにデータを出力する役割を担う)

1. PIC32CZ CA90 MCUによる初めてのアプリケーション作成

このデモでは以下のソフトウェアおよびハードウェア ツールを使います。

- MPLAB X IDE v6.15
- MCC (MPLAB Code Configurator)プラグインv5.4.1
- MPLAB XC32コンパイラv4.35
- MPLAB Harmony v3リポジトリ:
  - MPLAB Harmony v3 CSP(チップサポート パッケージ)v3.18.1
  - MPLAB Harmony v3 dev_packs v3.18.1
- PIC32CZ CA90 Curiosity Ultra開発ボード

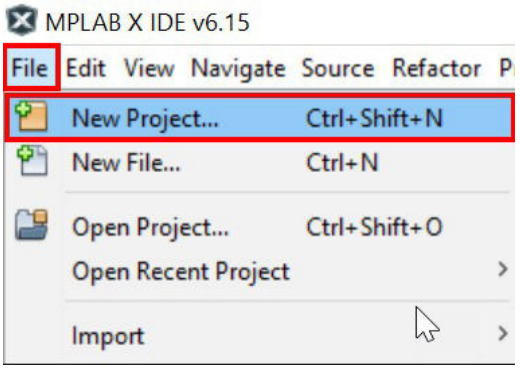
Note: 上記ツールが更新された場合、その新バージョンでアプリケーションを作成できます。古いバージョンを使う必要はありません。

1.1 MPLAB Harmony v3プロジェクトの作成

MPLAB Harmony v3ベースのプロジェクトを作成するには以下のステップを実行します。

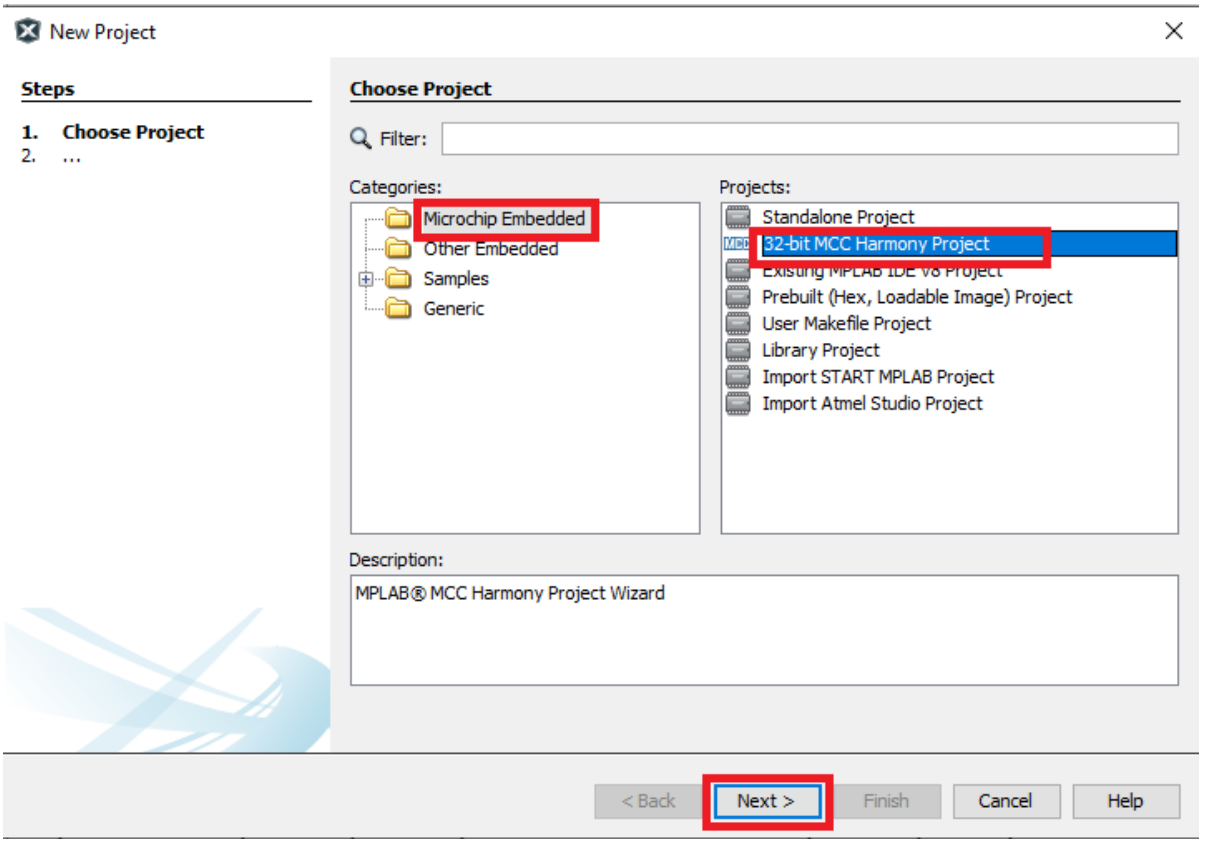
1. スタートメニューからMPLAB X IDEを起動します。
2. MPLAB X IDEの[File]メニューで**[New Project]**を選択します。

図1-1. プロジェクトの新規作成



3. [New Project]ウィンドウの[Steps]ナビゲーション ペインで**[Choose Project]**を選択します。右の[Choose Project]プロパティ セクションの[Categories]で**[Microchip Embedded]**を、[Projects]で**[32-bit MCC Harmony Project]**を選択します。
   Note: [32-bit MCC Harmony Project]が表示されていない場合、[Tools]メニューから[Plugins] > [Available Plugins]を選択し、MPLAB Code Configuratorプラグインをインストールしてからデモを続行してください。詳細はMPLAB Code Configuratorの概要(https://www.microchip.com/en-us/tools-resources/configure/mplab- code-configurator)を参照してください。

図1-2. MPLAB Harmony v3プロジェクトの作成 - Choose Project

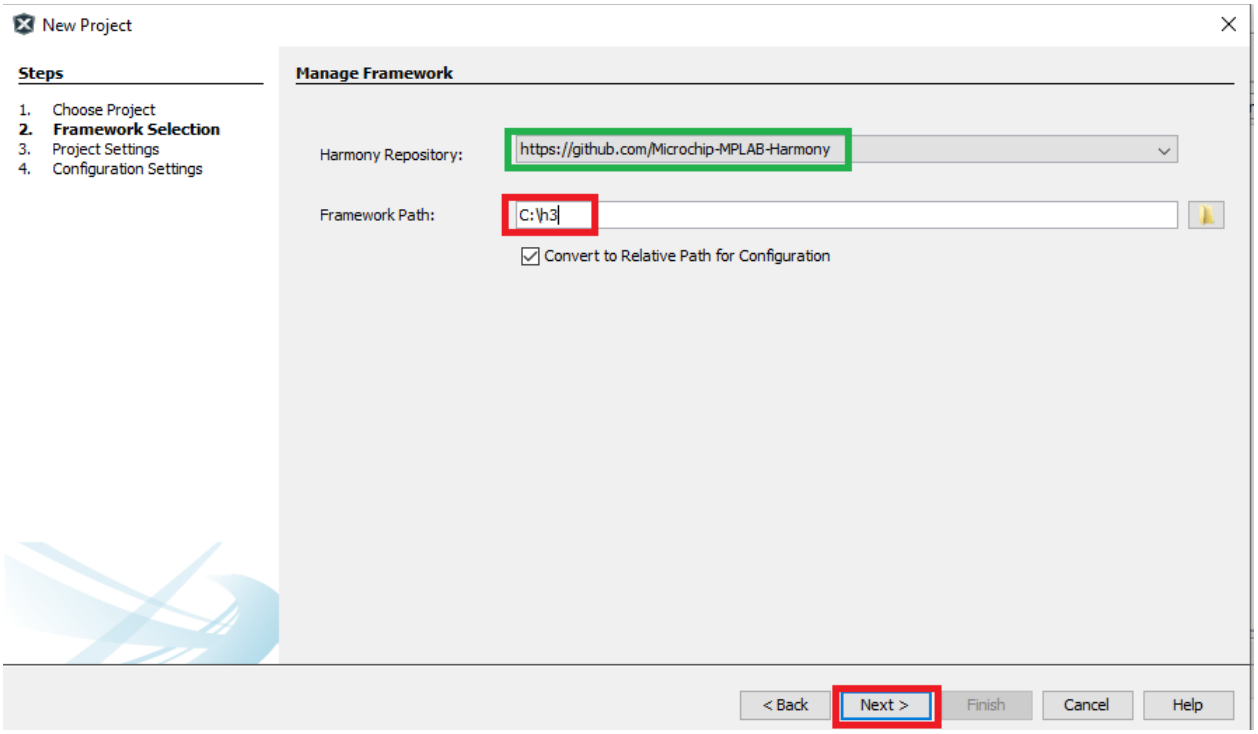


4. **[Next]**をクリックします。
5. 左のナビゲーション ペインで**[Framework Selection]**を選択し、右の[Manage Framework]プロパティ セクションで以下の詳細を入力します。

   a. Harmony Repository: パス「https://github.com/Microchip-MPLAB-Harmony」を入力します。

   b. Framework Path: 「C:\h3」(MPLAB Harmony v3パッケージをダウンロードするパス)を入力します。

図1-3. MPLAB Harmony v3プロジェクトの作成 - Framework Selection

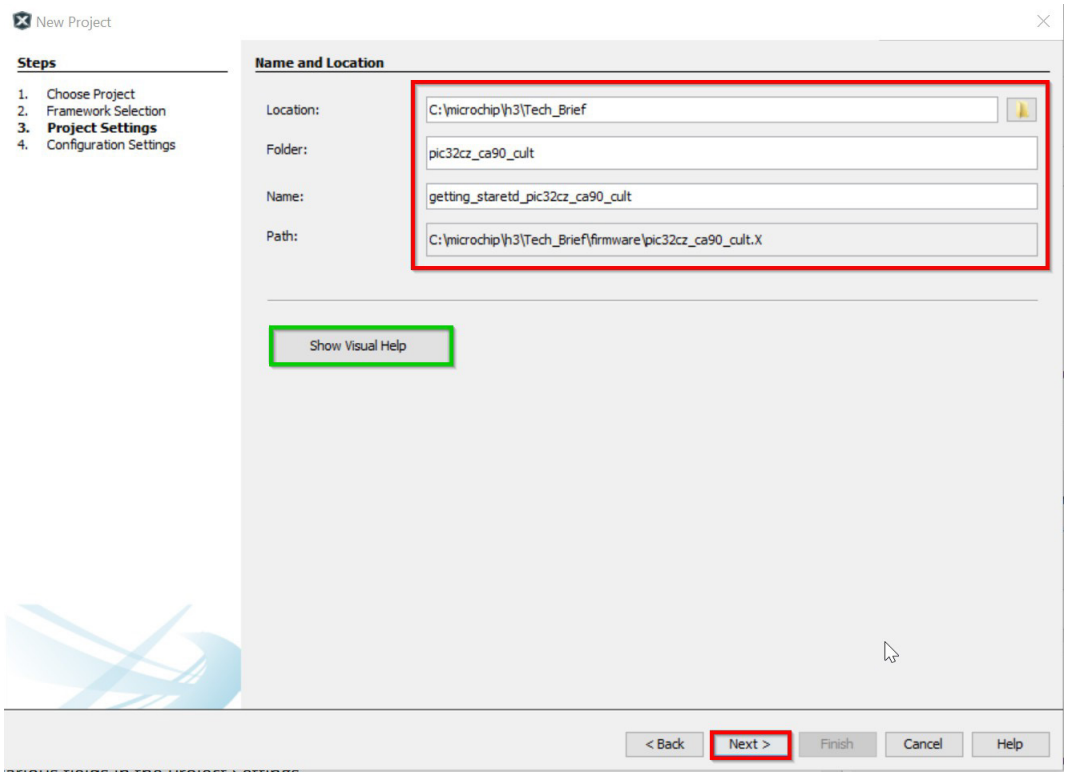


Note: このデモ アプリケーションではMPLAB Harmony v3パッケージ`csp`が必要です。MPLAB Harmony v3 Content Managerツールを使うとMPLAB Harmony v3パッケージのダウンロードを簡単に行えます。パッケージがダウンロードされていない場合、MPLAB Harmony v3 Content Managerツールを使ってダウンロードしてください。

6. **[Next]**をクリックします。
7. **[Project Settings]**を選択し、[Name and Location]プロパティ セクションに以下の詳細を入力します。
   a. Location: 新規プロジェクトのルートフォルダへのパスです。このフォルダに全てのプロジェクトファイルを保存します。プロジェクトの保存場所は、有効であればどのような場所でも可能です(例: `C:\microchip\h3\Tech_Brief`)。
   b. Folder: MPLAB X IDEフォルダの名前です。「`pic32cz_ca90_cult`」と入力し、pic32cz_ca90_cult.xフォルダを作成します。
   c. Name: MPLAB X IDEに表示されるプロジェクト名です。「`getting_started_pic32cz_ca90_cult`」と入力します。
   d. Path: この情報は、他のフィールドに変更を加えると自動で更新されます。

Note: PIC32CZ CA80 Curiosity Ultra開発ボード用のプロジェクトも同じ手順で作成して設定できます。

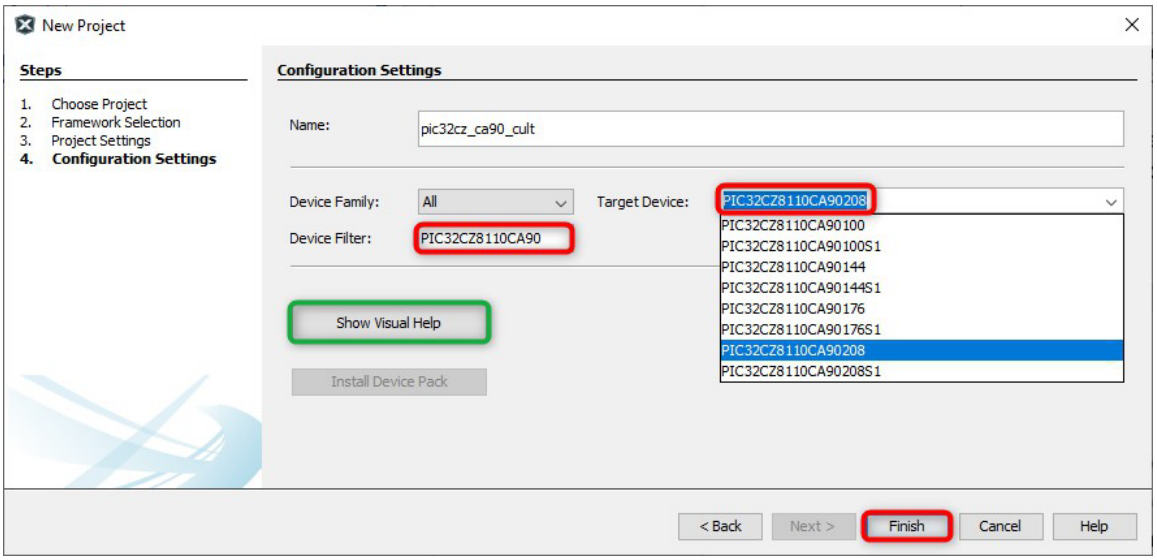
図1-4. MPLAB Harmony v3プロジェクトの作成 - Project Settings



Note: [Show Visual Help]ボタンをクリックするとコンテクストヘルプ ウィンドウが開き、[Project Settings]内の各フィールドの詳細が表示されます。

8. **[Next]**をクリックします。
9. **[Configuration Settings]**を選択し、[Configuration Settings]で以下の詳細を入力します。
    1. Name: 「`pic32cz_ca90_cult`」と入力します。
    2. Device Filter: 「PIC32CZ8110CA90」と入力します。
    3. Target Device: [Device Filter]で入力した内容がここに反映されます。PIC32CZ CA90 Curiosity Ultra開発ボード用のプロジェクトを作成するため、「PIC32CZ8110CA90208」を選択します。

図1-5. MPLAB Harmony v3プロジェクトの作成 - Configuration Settings

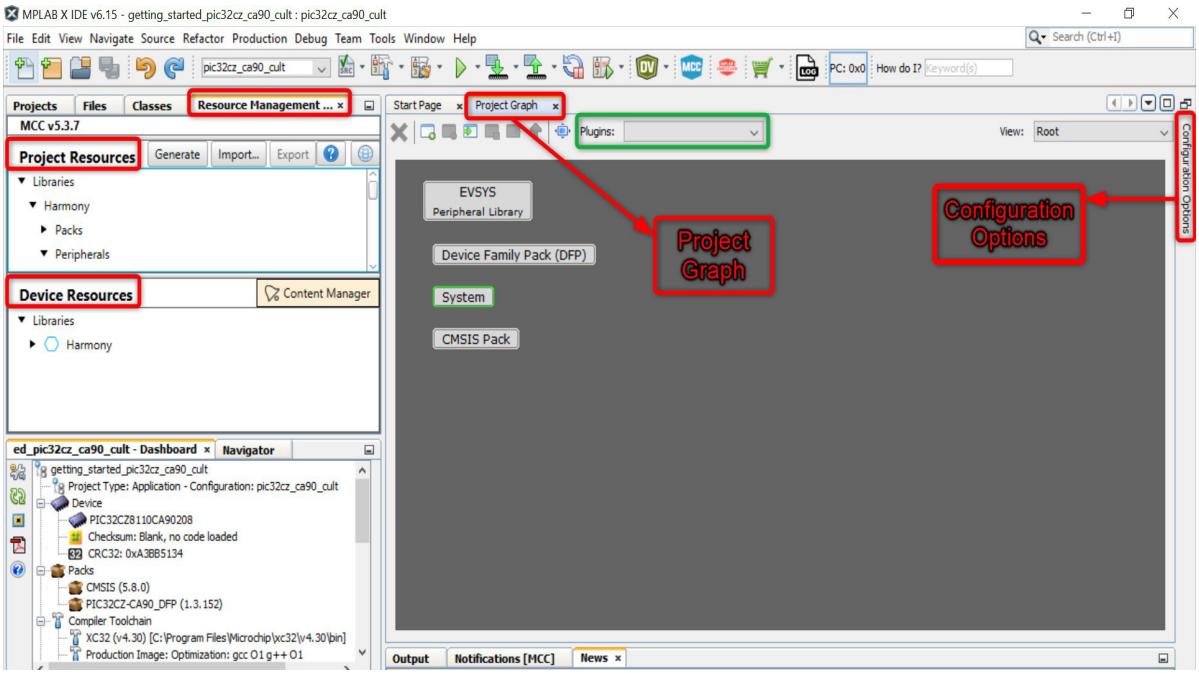


Note: [Show Visual Help]ボタンをクリックするとコンテクストヘルプ ウィンドウが開き、[Configuration Settings]の各フィールドの説明が表示されます。

Note: PIC32CZ CA90 Curiosity Ultra開発ボードのプロジェクトを作成するには、[Device Filter]ボックスに「PIC32CZ8110CA90208」と入力し、**[Finish]**をクリックします。以下の設定はPIC32CZ CA90 Curiosity Ultra開発ボードとPIC32CZ CA80 Curiosity Ultra開発ボードで有効です。PIC32CZ CA80 Curiosity Ultra開発ボードの場合、[Target Device]の名前を「PIC32CZ8110CA80208」に変更する必要があります。

10. **[Finish]**をクリックしてMCCを起動します。
    Note: 既定値ではMCC起動時にプロジェクトがメイン プロジェクトとして設定されます。
11. MCC起動前に[Configuration Database Setup]ウィンドウが表示されます。必要に応じてDevice Family Pack (DFP)とCortex Microcontroller Software Interface Standard (CMSIS)のパスを変更できます。このデモでは既定値設定を使います。
12. 新しいウィンドウでMCCプラグインが開きます。

図1-6. MPLAB Code Configuratorウィンドウ

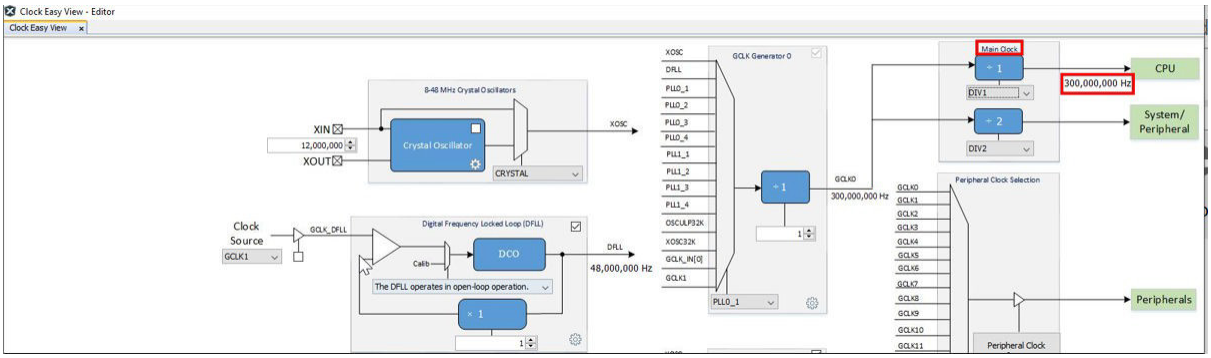


1.2 MCCを使ったMPLAB Harmony v3コンポーネントの追加と設定

MCCでMPLAB Harmony v3のコンポーネントを追加、設定するには以下のステップを実行します。

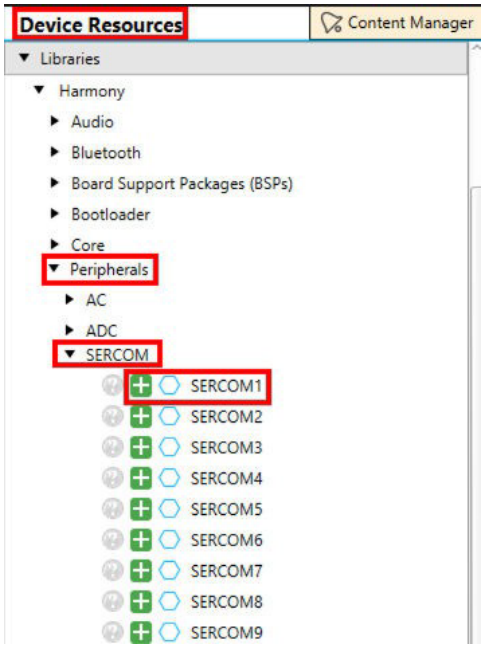
1. MPLAB X IDEでClock Easy Viewを起動するには、**[Project Graph]**をクリックし、[Plugins]ドロップダウン リストで**[Clock Configuration]**を選択します。Clock Easy ViewはMCCウィンドウ内に表示されます。
2. 右にスクロールして[Main Clock]が300 MHzに設定されている事を確認します。

図1-7. MPLAB Code Configurator - [Clock Easy View]ウィンドウ



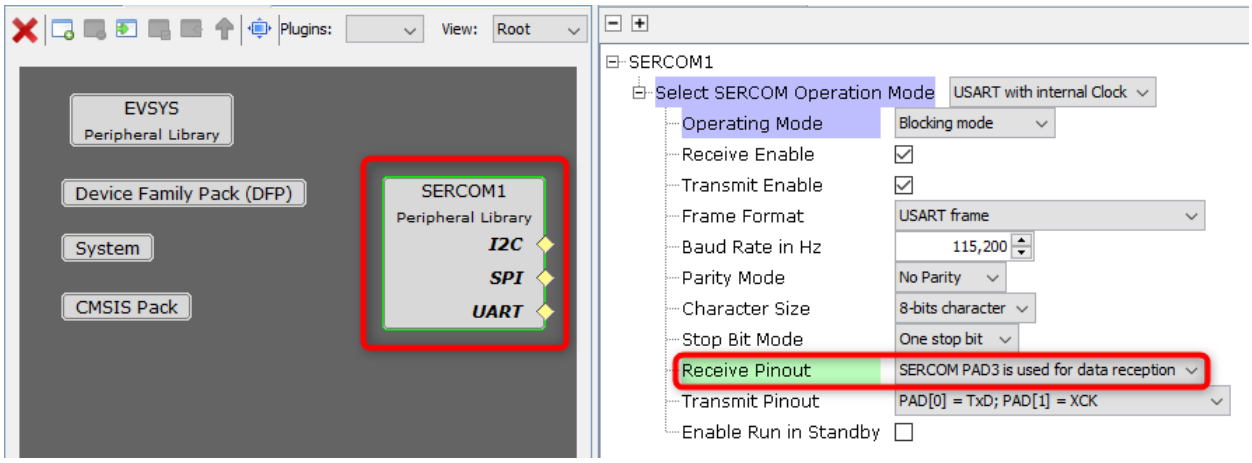
3. 左の[Device Resources]ナビゲーション ペインで[Libraries] > [Harmony] > [Peripherals] > [SERCOM]をクリックして展開し、**[SERCOM1]**をクリックします。プロジェクト グラフにSERCOM1モジュール ライブラリが追加された事を確認します。

図1-8. MPLAB Code Configurator - Device Resources



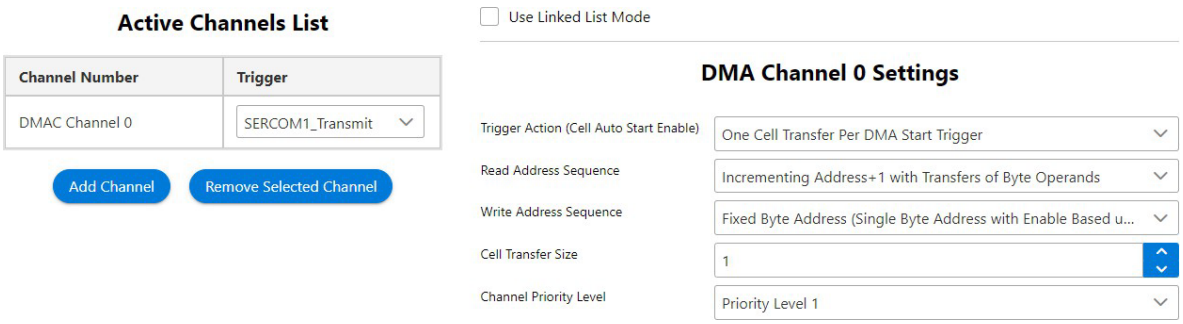
4. **[SERCOM1 Peripheral Library]**を選択し、[Configuration Options]プロパティ セクションでLED0のトグルレートをシリアル コンソールに出力する設定にします。
   a. [Receive Pinout]を表示するには、ドロップダウン リストから「**SERCOM PAD[3]**」を選択し、残りのパラメータは下図に示すように既定値のままにします。

図1-9. MPLAB Code Configurator - SERCOM1の設定



5. [Plugins]ドロップダウン リストから**[DMA Configuration]**を選択します。アプリケーション バッファをUSART TXレジスタに送信するように**DMA Channel 0**を設定します。DMAはユーザーバッファからUSART送信バッファに1回のトリガで1バイトを転送します。

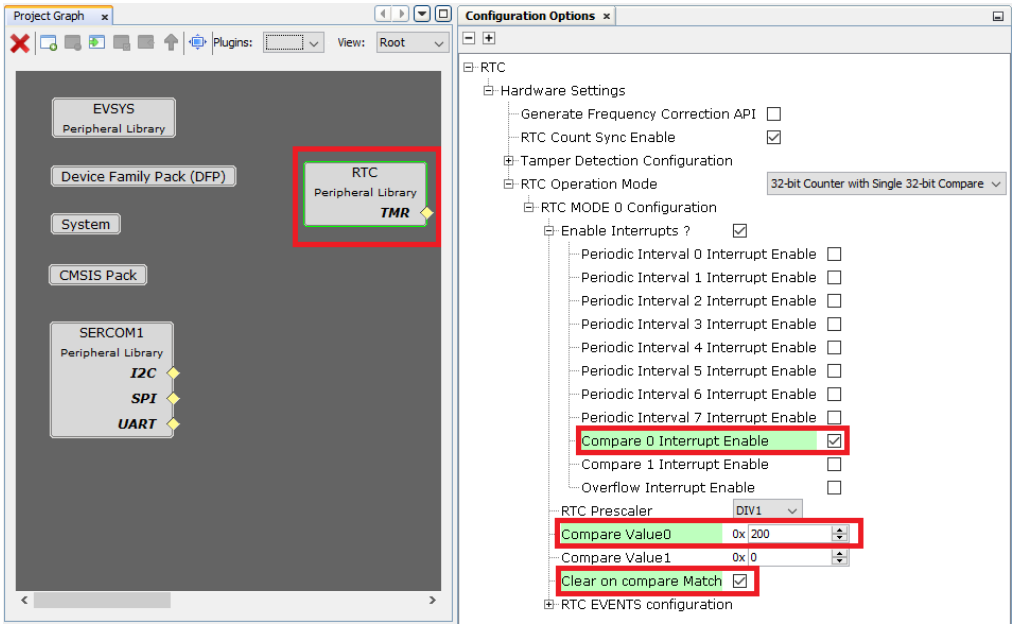
図1-10. MPLAB Code Configurator - DMAの設定



Note: SERCOM1とDMAはLED0のトグルレートをアプリケーションから取得し、そのLED0トグルレートをPC上で実行されているシリアル コンソールに出力するように設定されます。

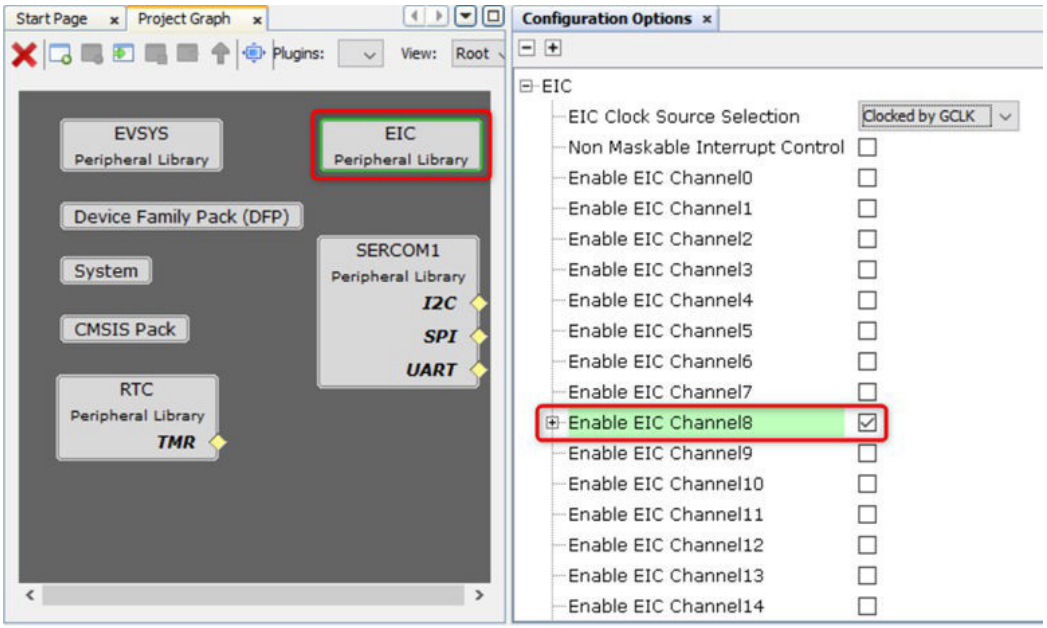
6. 左の**[Device Resources]**ナビゲーション ペインで[Libraries] > [Harmony] > [Peripherals]をクリックして展開し、**[RTC]**をクリックします。プロジェクト グラフに[RTC Peripheral Library]ブロックが追加された事を確認します。
7. **[RTC Peripheral Library]**を選択し、[Configuration Options]プロパティ セクションで下図の通り設定します(500ミリ秒ごとにコンペア割り込みを発生させる設定)。

図1-11. MPLAB Code Configurator - RTC PLIBの設定



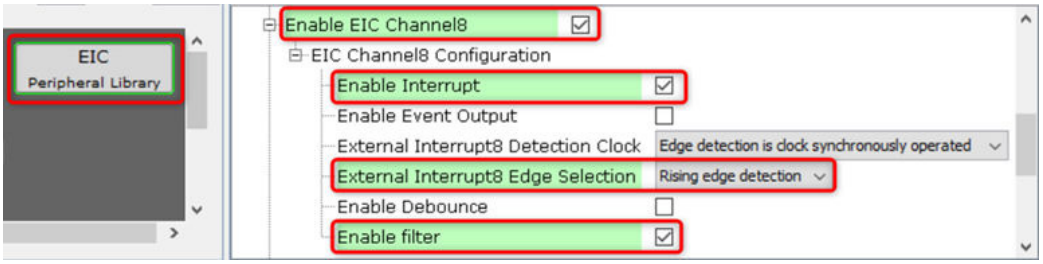
8. **[Device Resources]**で[Libraries] > [Harmony] > [Peripherals]をクリックして展開し、**[EIC]**をクリックします。
   プロジェクト グラフに[EIC Peripheral Library]ブロックが追加された事を確認します。
9. **[EIC Peripheral Library]**を選択し、[Configuration Options]プロパティ セクションで下図の通り設定します。

図1-12. MPLAB Code Configurator - EIC PLIBの設定



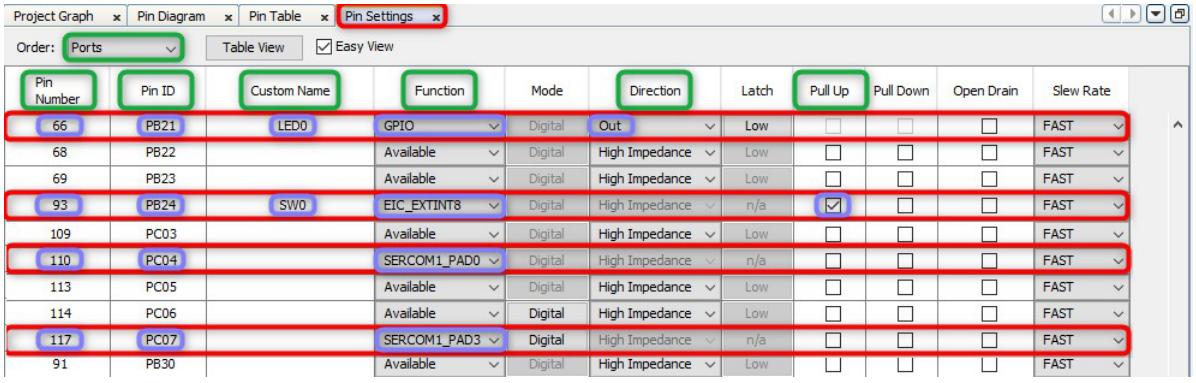
a. **[Enable EIC Channel8]**を選択して展開し、以下のように設定します。

図1-13. MPLAB Code Configurator - EIC PLIBのチャンネル8の設定



10. [Plugins]ドロップダウン リストから**[Pin Configuration]**を選択して**[Pin Settings]**をクリックします。
11. [Order]ボックスで**[Ports]**を選択し、アプリケーションの要件に従って設定します。
12. ピンIDがPB21とPB24の2本のピンについて[Custom Name]を変更します。以下のピン設定の図を参照してGPIO、SERCOM1、EIC_EXTINT8を設定します。

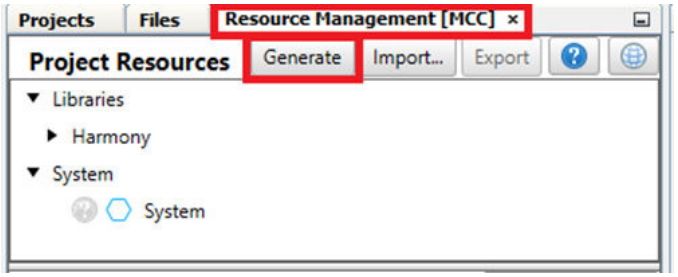
図1-14. MCCのピン設定



1.3 コードの生成

周辺モジュールの設定が完了したら、**[Resource Management [MCC]]**をクリックし、**[Generate]**タブをクリックしてコードを生成します。

図1-15. コードの生成



Note: 生成されたコードにより、32ビットMCC Harmonyプロジェクトにファイルとフォルダが追加されます。その生成コードで、RTC(リアルタイム クロック)、EIC(外部割り込みコントローラ)、PORT、SERCOM1 (USARTとして)、DMAモジュールに対して生成された周辺モジュール ライブラリ ファイルがプロジェクトに追加されている事を確認します。MCCによりプロジェクトに`main.c`ファイルも生成されます。

Note: MCCでは生成されるファイルの名前を変更できます。変更しない場合、既定値でファイル名`main.c`が生成されます。

1.4 プロジェクトにアプリケーション ロジックを追加する

アプリケーションを開発して実行するには以下のステップを実行します。

1. プロジェクトの`main.c`ファイルを開き、以下のバッファとアプリケーション ロジックの初期化を追加します。

```
uint8_t uartLocalTxBuffer[100] = {0};
    RTC_Timer32CallbackRegister(rtcEventHandler, 0);
    EIC_CallbackRegister(EIC_PIN_8, SW0_eventHandler, 0);
    DMA_ChannelCallbackRegister(DMA_CHANNEL_0, UARTDmaChannelHandler, 0);
```

図1-16. アプリケーション ロジックを追加してコールバック イベントハンドラを登録する

```
int main ( void )
{
    uint8_t uartLocalTxBuffer[100] = {0};

    /* Initialize all modules */
    SYS_Initialize ( NULL );

    RTC_Timer32CallbackRegister(rtcEventHandler, 0);
    EIC_CallbackRegister(EIC_PIN_8, SW0_eventHandler, 0);
    DMA_ChannelCallbackRegister(DMA_CHANNEL_0, UARTDmaChannelHandler, 0);
    RTC_Timer32Start();

    while ( true )
    {
```

a. `main()`関数に上記コードを追加し、コールバック イベントハンドラを登録します。コールバック イベントハンドラを登録した後、`RTC_Timer32Start()`関数を呼び出すコード行を追加します。

b. `main()`関数の前に以下のコードを追加し、周辺モジュールについて登録されたコールバック イベントハンドラを実装します。

```
static void SW0_eventHandler(uintptr_t context)
{
    changeTempSamplingRate = true;
}
```

```
static void rtcEventHandler (RTC_TIMER32_INT_MASK intCause, uintptr_t context)
{
    if (intCause & RTC_TIMER32_INT_MASK_CMP0)
    {
        isRTCTimerExpired = true;
    }
}

static void UARTDmaChannelHandler(DMA_TRANSFER_EVENT event, uintptr_t contextHandle)
{
    if (event == DMA_TRANSFER_EVENT_BLOCK_TRANSFER_COMPLETE)
    {
        isUARTTxComplete = true;
    }
}
```

2. プリプロセッサ ディレクティブ、マクロ、ローカル変数、フラグを宣言します。

```
#include <stdio.h>
#include <stddef.h>                     // Defines NULL
#include <stdbool.h>                    // Defines true
#include <stdlib.h>                     // Defines EXIT_FAILURE
#include <string.h>
#include "definitions.h"                // SYS function prototypes
#include "device_cache.h"

/* RTC Time period match values for input clock of 1 KHz */
#define PERIOD_500MS                    512
#define PERIOD_1S                       1024
#define PERIOD_2S                       2048
#define PERIOD_4S                       4096

static volatile bool isRTCTimerExpired = false;
static volatile bool changeTempSamplingRate = false;
static volatile bool isUARTTxComplete = true;
static uint8_t __attribute__ ((aligned (16))) uartTxBuffer[100] = {0};

typedef enum
{
    TEMP_SAMPLING_RATE_500MS = 0,
    TEMP_SAMPLING_RATE_1S = 1,
    TEMP_SAMPLING_RATE_2S = 2,
    TEMP_SAMPLING_RATE_4S = 3,
} TEMP_SAMPLING_RATE;

static TEMP_SAMPLING_RATE tempSampleRate = TEMP_SAMPLING_RATE_500MS;
static const char timeouts[4][20] = {"500 milliSeconds", "1 Second", "2 Seconds", "4
Seconds"};
```

3. 両方のフラグがイベントの完了を検出した時(`while()`スーパーループ内の`(isRTCExpired)`と`(isUSARTTxComplete))`にアプリケーション ロジックを追加します。

```
       while ( true )
    {
        if ((isRTCTimerExpired == true) && (true == isUARTTxComplete))
            {
                isRTCTimerExpired = false;
                isUARTTxComplete = false;
                LED0_Toggle();
                sprintf((char*)(uartTxBuffer), "Toggling LED at %s rate
\r\n",
                &timeouts[(uint8_t)tempSampleRate][0]);
                DCACHE_CLEAN_BY_ADDR(uartTxBuffer, sizeof(uartTxBuffer));
                DMA_ChannelTransfer(DMA_CHANNEL_0, uartTxBuffer, \
                    (const void *)&(SERCOM1_REGS->USART_INT.SERCOM_DATA), \
                    strlen((const char*)uartTxBuffer));
             }
```

図1-17. アプリケーション ロジックの追加

```
while ( true )
{
    if ((isRTCTimerExpired == true) && (true == isUARTTxComplete))
        {
            isRTCTimerExpired = false;
            isUARTTxComplete = false;
            LED0_Toggle();
            sprintf((char*)(uartTxBuffer), "Toggling LED at %s rate \r\n", &timeouts[(uint8_t)tempSampleRate][0]);
            DCACHE_CLEAN_BY_ADDR(uartTxBuffer, sizeof(uartTxBuffer));
            DMA_ChannelTransfer(DMA_CHANNEL_0, uartTxBuffer, \
                (const void *)&(SERCOM1_REGS->USART_INT.SERCOM_DATA), \
                strlen((const char*)uartTxBuffer));
        }
```

4. ユーザーがボード上のスイッチを押下するたびに500 ms、1s、2s、4sの各レートでLED0をトグルさせるアプリケーション ロジックを追加します。DMA転送ロジックは`main.c`ファイル内の`main()`関数にあります。

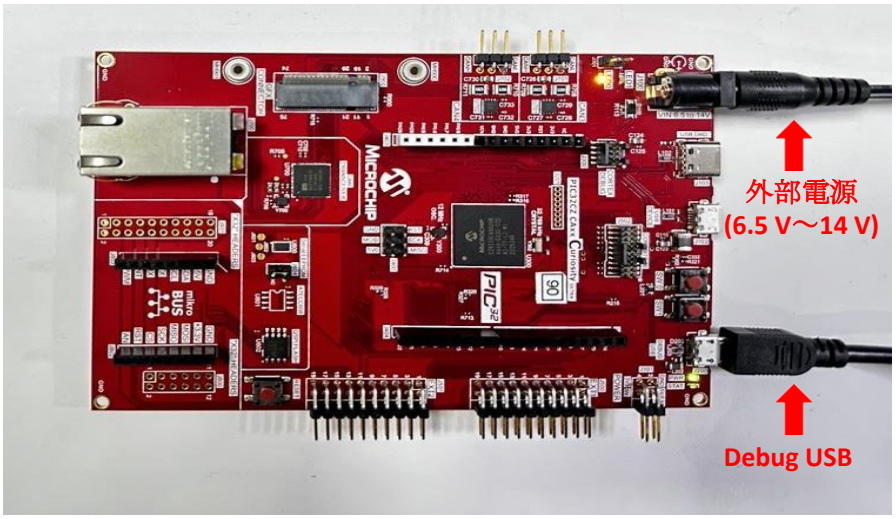
```
        /* Maintain state machines of all polled MPLAB Harmony modules. */
             if(changeTempSamplingRate == true)
            {
                changeTempSamplingRate = false;
                if(tempSampleRate == TEMP_SAMPLING_RATE_500MS)
                {
                    tempSampleRate = TEMP_SAMPLING_RATE_1S;
                    RTC_Timer32Compare0Set(PERIOD_1S);
                }
                else if(tempSampleRate == TEMP_SAMPLING_RATE_1S)
                {
                    tempSampleRate = TEMP_SAMPLING_RATE_2S;
                    RTC_Timer32Compare0Set(PERIOD_2S);
                }
                else if(tempSampleRate == TEMP_SAMPLING_RATE_2S)
                {
                    tempSampleRate = TEMP_SAMPLING_RATE_4S;
                    RTC_Timer32Compare0Set(PERIOD_4S);
                }
                else if(tempSampleRate == TEMP_SAMPLING_RATE_4S)
                {
                   tempSampleRate = TEMP_SAMPLING_RATE_500MS;
                   RTC_Timer32Compare0Set(PERIOD_500MS);
                }
                else
                {
                    ;
                }
                RTC_Timer32CounterSet(0);
                sprintf((char*)uartLocalTxBuffer, "LED Toggling rate is changed to
                %s\r\n", &timeouts[(uint8_t)tempSampleRate][0]);
                DCACHE_CLEAN_BY_ADDR(uartLocalTxBuffer, sizeof(uartLocalTxBuffer));
                DMA_ChannelTransfer(DMA_CHANNEL_0, uartLocalTxBuffer, \
                    (const void *)&(SERCOM1_REGS->USART_INT.SERCOM_DATA), \
                    strlen((const char*)uartLocalTxBuffer));
            }

    }
    /* Execution should not come here during normal operation */

    return ( EXIT_FAILURE );
}
```

1.5 アプリケーションのビルドとプログラミング

アプリケーションをビルドしてプログラミングするには以下のステップを実行します。

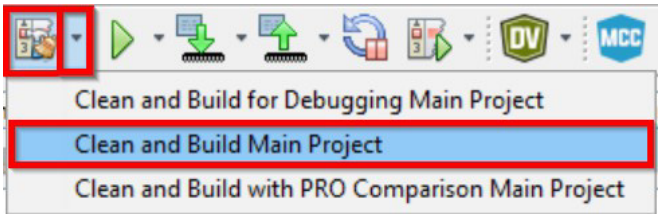
1. PIC32CZ CA90 Curiosity Ultra開発ボードはデバッグ用にオンボード デバッガ(PKoB4)をサポートしています。Type-Aオス - micro-B USBケーブルをmicro-B DEBUG USBポートに接続し、PIC32CZ CA90 Curiosity Ultra開発ボードへの電力供給とデバッグを実行します。外部電源(6.5 V～14 V)をJ100コネクタに接続します。

図1-18. ハードウェアの準備



2. プロジェクトをメイン プロジェクトに設定するには、[Project Properties]で最新のコンパイラ バージョン(v4.35)を選択します。
3. プロジェクトのクリーンとビルドを実行するには、以下に強調表示したアイコンをクリックします。または、ドロップダウン リストから**[Clean and Build Main Project]**を選択します。

図1-19. クリーンとビルド



4. 以下の強調表示されたアイコンをクリックしてアプリケーションをプログラミングします。

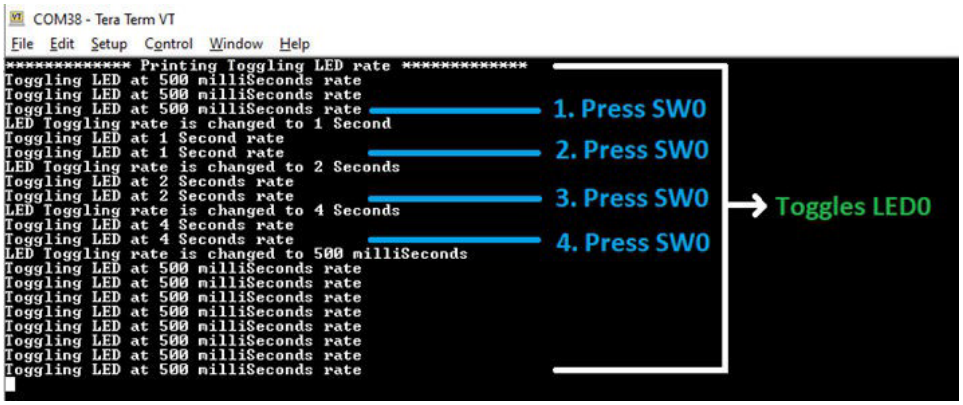
図1-20. デバイスのプログラミング

1.6 ボードとシリアル ターミナルでの出力の観察

ボード上とシリアル ターミナル上で出力を観察するには、以下の手順に従います。

1. アプリケーションのビルドとプログラミングを実行した後、PCでTera Termツールを開きます。
2. シリアルポートを選択し、baudレートを「115200」に設定します。
3. PIC32CZ CA90 Curiosity Ultra開発ボードのリセットボタンを押します。既定値ではLED0は500 msでトグルし、以後SW0スイッチが押下されるごとにLED0のトグルレートは1s、2s、4sに切り換わります。

図1-21. シリアル ターミナルに表示されたLED0トグルレート



シリアル ターミナルに表示されたLED0トグルレートはスイッチが押下されるたびに変化しますが、それに伴って開発ボードでLED0のトグルレートが変化している事を観察します。

2. 参考資料

- MPLAB Harmony v3に関する情報はMicrochip社ウェブサイトの https://www.microchip.com/mplab/mplab-harmonyと microchipdeveloper.com/xwiki/bin/view/software-tools/harmony/を参照してください。
- 各種のアプリケーションの詳細はgithub.com/Microchip-MPLAB-Harmony/reference_appsを参照してください。
- PIC32CZ CA90 Curiosity Ultra評価用ボードの詳細はwww.microchip.com/en-us/development-tool/EA58X56Aを参照してください。
- サンプル アプリケーションは[Software]にある『Getting Started Extended Application on PIC32CZ CA90 Curiosity Ultra Development Board』を参照してください。 https://www.microchip.com/en-us/development-tool/EA58X56A
- 32ビット マイクロコントローラの関連情報とソリューションは以下を参照してください。 https://www.microchip.co.jp/download/dl_download.php/ID=1492567688591dfef8e8ee3dc28f9b8765f83860/ReferenceManuals/32-bit-Microcontroller-Collateral-and-Solutions-Reference-Guide-DS70005534.pdf

3. 改訂履歴

リビジョンA - 2024年1月

本書は初版です。

Microchip社の情報

Microchip社ウェブサイト

Microchip社はウェブサイト(www.microchip.com)を通してオンライン サポートを提供しています。当ウェブサイトでは、お客様に役立つ情報やファイルを提供しています。以下を含む各種の情報をご覧になれます。

- **製品サポート** - データシートとエラッタ、アプリケーション ノートとサンプル プログラム、設計リソース、ユーザーガイドとハードウェア サポート文書、最新のソフトウェアと過去のソフトウェア
- **技術サポート** - FAQ(よく寄せられる質問)、技術サポートのご依頼、オンライン ディスカッション グループ、Microchip社のデザイン パートナー プログラムおよびメンバーリスト
- **ご注文とお問い合わせ** - 製品セレクタと注文ガイド、最新プレスリリース、セミナー/イベントの一覧、お問い合わせ先(営業所/正規代理店)の一覧

製品変更通知サービス

Microchip社の製品変更通知サービスは、お客様にMicrochip社製品の最新情報をお届けする配信サービスです。ご興味のある製品ファミリまたは開発ツールに関する変更、更新、リビジョン、エラッタ情報をいち早くメールにてお知らせします。

http://www.microchip.com/pcnにアクセスし、登録手続きをしてください。

お客様サポート

Microchip社製品をお使いのお客様は、以下のチャンネルからサポートをご利用頂けます。

- 正規代理店
- 技術サポート

サポートは正規代理店にお問い合わせください。本書の最後のページに各国の営業所の一覧を記載しています。

技術サポートは以下のウェブページからもご利用頂けます。www.microchip.com/support

Microchip社のデバイスコード保護機能

Microchip 社製品のコード保護機能について以下の点にご注意ください。

- Microchip社製品は、該当するMicrochip 社データシートに記載の仕様を満たしています。
- Microchip社では、通常の条件ならびに動作仕様書の仕様に従って使った場合、Microchip 社製品のセキュリティ レベルは、現在市場に流通している同種製品の中でも最も高度であると考えています。
- Microchip社はその知的財産権を重視し、積極的に保護しています。Microchip 社製品のコード保護機能の侵害は固く禁じられており、デジタル ミレニアム著作権法に違反します。
- Microchip社を含む全ての半導体メーカーで、自社のコードのセキュリティを完全に保証できる企業はありません。コード保護機能とは、Microchip 社が製品を「解読不能」として保証するものではありません。コード保護機能は常に進化しています。Microchip 社では、常に製品のコード保護機能の改善に取り組んでいます。

法律上の注意点

本書および本書に記載されている情報は、Microchip 社製品を設計、テスト、お客様のアプリケーションと統合する目的を含め、Microchip 社製品に対してのみ使う事ができます。それ以外の方法でこの情報を使う事はこれらの条項に違反します。デバイス アプリケーションの情報は、ユーザーの便宜のためにのみ提供されるものであり、更新によって変更となる事があります。お客様のアプリケーションが仕様を

満たす事を保証する責任は、お客様にあります。その他のサポートはMicrochip 社正規代理店にお問い合わせ頂くか、https://www.microchip.com/en-us/support/design-help/client-support-servicesをご覧ください。

Microchip 社は本書の情報を「現状のまま」で提供しています。Microchip 社は明示的、暗黙的、書面、口頭、法定のいずれであるかを問わず、本書に記載されている情報に関して、非侵害性、商品性、特定目的への適合性の暗黙的保証、または状態、品質、性能に関する保証をはじめとするいかなる類の表明も保証も行いません。

いかなる場合もMicrochip 社は、本情報またはその使用に関連する間接的、特殊的、懲罰的、偶発的または必然的損失、損害、費用、経費のいかんにかかわらず、またMicrochip 社がそのような損害が生じる可能性について報告を受けていた場合あるいは損害が予測可能であった場合でも、一切の責任を負いません。法律で認められる最大限の範囲を適用しようとも、本情報またはその使用に関連する一切の申し立てに対するMicrochip 社の責任限度額は、使用者が当該情報に関連してMicrochip 社に直接支払った額を超えません。

Microchip 社の明示的な書面による承認なしに、生命維持装置あるいは生命安全用途にMicrochip社の製品を使う事は全て購入者のリスクとし、また購入者はこれによって発生したあらゆる損害、クレーム、訴訟、費用に関して、Microchip 社は擁護され、免責され、損害をうけない事に同意するものとします。特に明記しない場合、暗黙的あるいは明示的を問わず、Microchip社が知的財産権を保有しているライセンスは一切譲渡されません。

商標

Microchip 社の名称とロゴ、Microchip ロゴ、Adaptec、AVR、AVR ロゴ、AVR Freaks、BesTime、BitCloud、CryptoMemory、CryptoRF、dsPIC、flexPWR、HELDO、IGLOO、JukeBlox、KeeLoq、Kleer、LANCheck、LinkMD、maXStylus、maXTouch、MediaLB、megaAVR、Microsemi、Microsemi ロゴ、MOST、MOST ロゴ、MPLAB、OptoLyzer、PIC、picoPower、PICSTART、PIC32 ロゴ、PolarFire、Prochip Designer、QTouch、SAM-BA、SenGenuity、SpyNIC、SST、SST ロゴ、SuperFlash、Symmetricom、SyncServer、Tachyon、TimeSource、tinyAVR、UNI/O、Vectron、XMEGA は米国とその他の国におけるMicrochip Technology Incorporated の登録商標です。

AgileSwitch、APT、ClockWorks、The Embedded Control SolutionsCompany、EtherSynch、Flashtec、Hyper Speed Control、HyperLightLoad、Libero、motorBench、mTouch、Powermite 3、Precision Edge、ProASIC、ProASIC Plus、ProASIC Plus ロゴ、Quiet-Wire、SmartFusion、SyncWorld、Temux、TimeCesium、TimeHub、TimePictra、TimeProvider、TrueTime、ZL は米国におけるMicrochip Technology Incorporated の登録商標です。

Adjacent Key Suppression、AKS、Analog-for-the-Digital Age、Any Capacitor、AnyIn、AnyOut、Augmented Switching、BlueSky、BodyCom、Clockstudio、CodeGuard、CryptoAuthentication、CryptoAutomotive、CryptoCompanion、CryptoController、dsPICDEM、dsPICDEM.net、Dynamic Average Matching、DAM、ECAN、Espresso T1S、EtherGREEN、GridTime、IdealBridge、In-Circuit Serial Programming、ICSP、INICnet、Intelligent Paralleling、IntelliMOS、Inter-Chip Connectivity、JitterBlocker、Knob-on-Display、KoD、maxCrypto、maxView、memBrain、Mindi、MiWi、MPASM、MPF、MPLAB Certified ロゴ、MPLIB、MPLINK、MultiTRAK、NetDetach、Omniscient Code Generation、PICDEM、PICDEM.net、PICkit、PICtail、PowerSmart、PureSilicon、QMatrix、REAL ICE、RippleBlocker、RTAX、RTG4、SAM-ICE、Serial Quad I/O、simpleMAP、SimpliPHY、SmartBuffer、SmartHLS、SMART-I.S.、storClad、SQI、SuperSwitcher、SuperSwitcher II、Switchtec、SynchroPHY、TotalEndurance、Trusted Time、TSHARC、USBCheck、VariSense、VectorBlox、VeriPHY、ViewSpan、WiperLock、XpressConnect、ZENAは米国とその他の国におけるMicrochip Technology Incorporated の商標です。

SQTP は米国におけるMicrochip Technology Incorporated のサービスマークです。

Adaptec ロゴ、Frequency on Demand、Silicon Storage Technology、Symmcom はその他の国における Microchip Technology Incorporatedの登録商標です。

ISBN: 978-1-6683-4858-1

品質管理システム

Microchip社の品質管理システムについてはwww.microchip.com/qualityをご覧ください。

各国の営業所とサービス

南北アメリカ	アジア/太平洋	アジア/太平洋	欧州
本社 2355 West Chandler Blvd. Chandler, AZ 85224-6199 Tel: 480-792-7200 Fax: 480-792-7277 技術サポート: http://www.microchip.com/support URL: www.microchip.com **アトランタ** Duluth, GA Tel: 678-957-9614 Fax: 678-957-1455 **オースティン、TX** Tel: 512-257-3370 **ボストン** Westborough, MA Tel: 774-760-0087 Fax: 774-760-0088 **シカゴ** Itasca, IL Tel: 630-285-0071 Fax: 630-285-0075 **ダラス** Addison, TX Tel: 972-818-7423 Fax: 972-818-2924 **デトロイト** Novi, MI Tel: 248-848-4000 **ヒューストン、TX** Tel: 281-894-5983 **インディアナポリス** Noblesville, IN Tel: 317-773-8323 Fax: 317-773-5453 Tel: 317-536-2380 **ロサンゼルス** Mission Viejo, CA Tel: 949-462-9523 Fax: 949-462-9608 Tel: 951-273-7800 **ローリー、NC** Tel: 919-844-7510 **ニューヨーク、NY** Tel: 631-435-6000 **サンノゼ、CA** Tel: 408-735-9110 Tel: 408-436-4270 **カナダ - トロント** Tel: 905-695-1980 Fax: 905-695-2078	**オーストラリア - シドニー** Tel: 61-2-9868-6733 **中国 - 北京** Tel: 86-10 -8569-7000 **中国 - 成都** Tel: 86-28-8665-5511 **中国 - 重慶** Tel: 86-23-8980-9588 **中国 - 東莞** Tel: 86-769-8702-9880 **中国 - 広州** Tel: 86-20-8755-8029 **中国 - 杭州** Tel: 86-571-8792-8115 **中国 - 香港SAR** Tel: 852-2943-5100 **中国 - 南京** Tel: 86-25-8473-2460 **中国 - 青島** Tel: 86-532-8502-7355 **中国 - 上海** Tel: 86-21-3326-8000 **中国 - 瀋陽** Tel: 86-24-2334-2829 **中国 - 深圳** Tel: 86-755-8864-2200 **中国 - 蘇州** Tel: 86-186-6233-1526 **中国 - 武漢** Tel: 86-27-5980-5300 **中国 - 西安** Tel: 86-29-8833-7252 **中国 - 厦門** Tel: 86-592-2388138 **中国 - 珠海** Tel: 86-756-3210040	**インド - バンガロール** Tel: 91-80-3090-4444 **インド - ニューデリー** Tel: 91-11-4160-8631 **インド - プネ** Tel: 91-20-4121-0141 **日本 - 大阪** Tel: 81-6-6152-7160 **日本 - 東京** Tel: 81-3-6880-3770 **韓国 - 大邱** Tel: 82-53-744-4301 **韓国 - ソウル** Tel: 82-2-554-7200 **マレーシア - クアラルンプール** Tel: 60-3-7651-7906 **マレーシア - ペナン** Tel: 60-4-227-8870 **フィリピン - マニラ** Tel: 63-2-634-9065 **シンガポール** Tel: 65-6334-8870 **台湾 - 新竹** Tel: 886-3-577-8366 **台湾 - 高雄** Tel: 886-7-213-7830 **台湾 - 台北** Tel: 886-2-2508-8600 **タイ - バンコク** Tel: 66-2-694-1351 **ベトナム - ホーチミン** Tel: 84-28-5448-2100	**オーストリア - ヴェルス** Tel: 43-7242-2244-39 Fax: 43-7242-2244-393 **デンマーク - コペンハーゲン** Tel: 45-4485-5910 Fax: 45-4485-2829 **フィンランド - エスポー** Tel: 358-9-4520-820 **フランス - パリ** Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79 **ドイツ - ガーヒンク** Tel: 49-8931-9700 **ドイツ - ハーン** Tel: 49-2129-3766400 **ドイツ - ハイルブロン** Tel: 49-7131-72400 **ドイツ - カールスルーエ** Tel: 49-721-625370 **ドイツ - ミュンヘン** Tel: 49-89-627-144-0 Fax: 49-89-627-144-44 **ドイツ - ローゼンハイム** Tel: 49-8031-354-560 **イスラエル - ラーナナ** Tel: 972-9-744-7705 **イタリア - ミラノ** Tel: 39-0331-742611 Fax: 39-0331-466781 **イタリア - パドヴァ** Tel: 39-049-7625286 **オランダ - ドリューネン** Tel: 31-416-690399 Fax: 31-416-690340 **ノルウェー - トロンハイム** Tel: 47-7288-4388 **ポーランド - ワルシャワ** Tel: 48-22-3325737 **ルーマニア - ブカレスト** Tel: 40-21-407-87-50 **スペイン - マドリッド** Tel: 34-91-708-08-90 Fax: 34-91-708-08-91 **スウェーデン - ヨーテボリ** Tel: 46-31-704-60-40 **スウェーデン - ストックホルム** Tel: 46-8-5090-4654 **イギリス - ウォーキンガム** Tel: 44-118-921-5800 Fax: 44-118-921-5820