XMC™でのファームウェアの整合性の管理

XMC1000およびXMC4000ファミリデバイス

About this document

Scope and purpose

このドキュメントでは、フラッシュの整合性を管理する方法を紹介し、XMC1000 および XMC4000 ファミリデバイスで CRC-32 方式を使用した例を示します。

Intended audience

このドキュメントは、XMC1000 および XMC4000 ファミリデバイスのユーザーを対象とします。

Table of contents

About this document **1**
Table of contents **1**
1 フラッシュの整合性と管理の概要 **2**
1.1 巡回冗長検査 (CRC) 方式 2
2 アプリケーションコードの調整 **5**
2.1 方法論 5
2.1.1 DAVE™、SRecord、および XMC™ flasher の使用 5
2.1.2 SRecord の生成 6
2.1.3 SRecord のフォーマット 8
2.1.4 XMC™ flasher を使用した CRC の適用 9
3 アプリケーション例 **11**
3.1 XMC1000 ファミリデバイス 11
3.2 XMC4000 ファミリデバイス 16
改訂履歴 **19**

1 フラッシュの整合性と管理の概要

マイクロコントローラフラッシュの整合性を管理することは、製造時、ファームウェアのアップロードごと、または起動ごとに、フラッシュの安全なプログラミングを確認するために重要です。冗長性チェック、チェックサム、フォールバックブートモードなど、フラッシュの整合性を管理するための一般的な方法があります。このドキュメントは、すべての方法の説明を網羅していません。代わりに、一般的な多項式を使用した 32 ビット CRC 計算を使用した例に焦点を当てます。これによって、デバイスが目的のデータでフラッシュされたことを確認する信頼できる方法を提供します。さらに、DAVE™ v4、XMC™ flasher、およびオープンソースツール SRecord を使用して、ネイティブ DAVE™ v4 環境でのアプリケーションビルドから、ターゲットにダウンロードする準備ができている信頼性の高いファームウェアイメージ (整合性チェックを含む) の作成までのプロセスを作成できます。最後に、これらの整合性チェックを管理するために使用できるターゲットコードの例を示します。

1.1 巡回冗長検査 (CRC) 方式

巡回冗長検査 (CRC) は、データのアクシデントによる破損を検出するためにデジタルアプリケーションで一般的に使用される方法です。CRC は、データ転送のノイズやフラッシュの破損によって引き起こされるエラーを軽減するように特別に設計されます。このチェック手順の前提は、データのブロックがその内容の多項式除算を受け、一意の剰余が生成されることです。これらの多項式は、多くの場合、さまざまなアプリケーションに共通する事前に決定された値であり、それを行うためにテストされています。CRC は、通信システム (CAN、イーサネット、RS485、モバイル通信など) では一般的です。XMC4000 デバイスで使用可能な柔軟な CRC エンジンには、CRC 計算のためのいくつかのオプションがあります。XMC1000 を使用して、専用の CRC エンジンがない場合に一般的な CRC アルゴリズムを適用します。以下の **Figure 1** に、XMC4000 FCE システムと FCE 内で使用可能なコンポーネントのブロック図を示します。

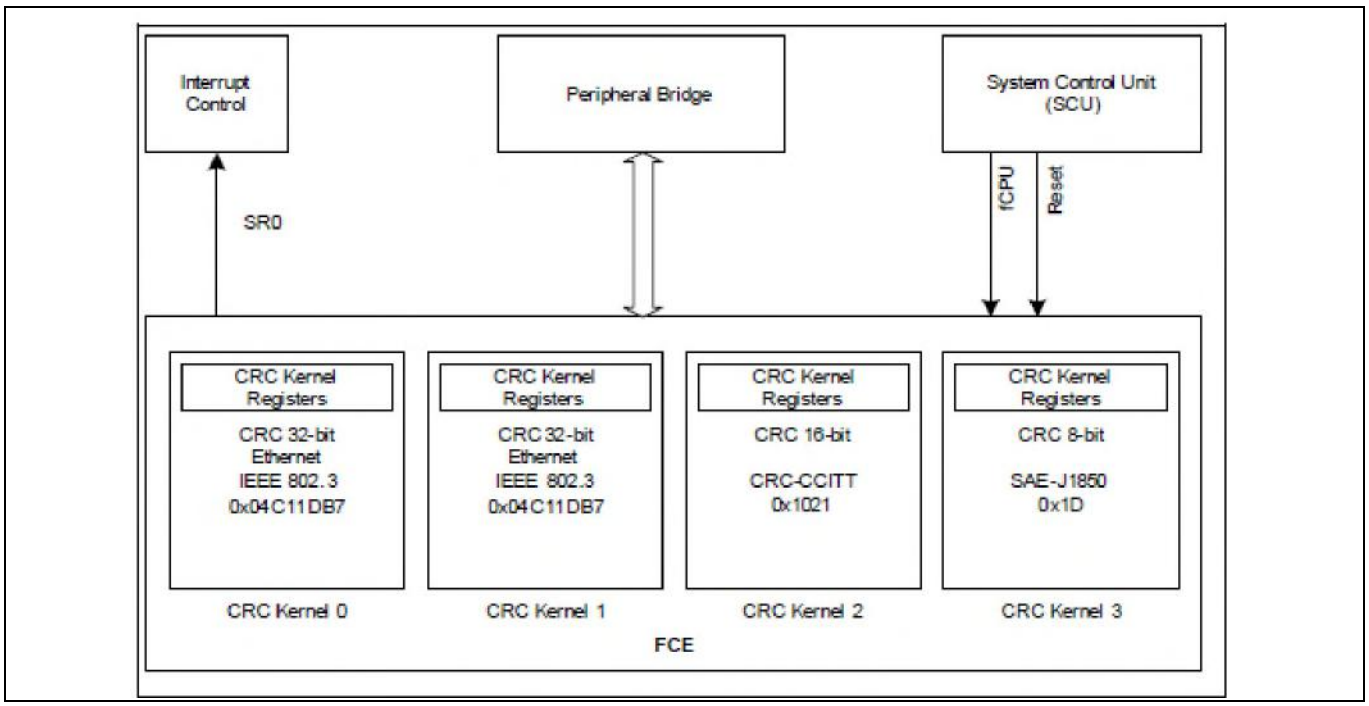

Figure 1 FCE ブロック図

このドキュメントの焦点が大量のデータの整合性のテストにあることを考えると、ビット CRC が高いほど、より一意のコードが計算されます。FCE で使用可能なイーサネット多項式 0x04C11DB7 などの多項式を使用できます。これは次のように解釈されます。

Eq. 1　$x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^{8}+x^{7}+x^{5}+x^{4}+x^{2}+x^{1}+1$ (32 番目のビットで先頭に '1'が付きます):

またはバイナリでは以下です。

100000100110000010001110110110111

剰余の多項式除算の例として、32 ビットワード 0x01234567 を使用して、以下の CRC を計算します。

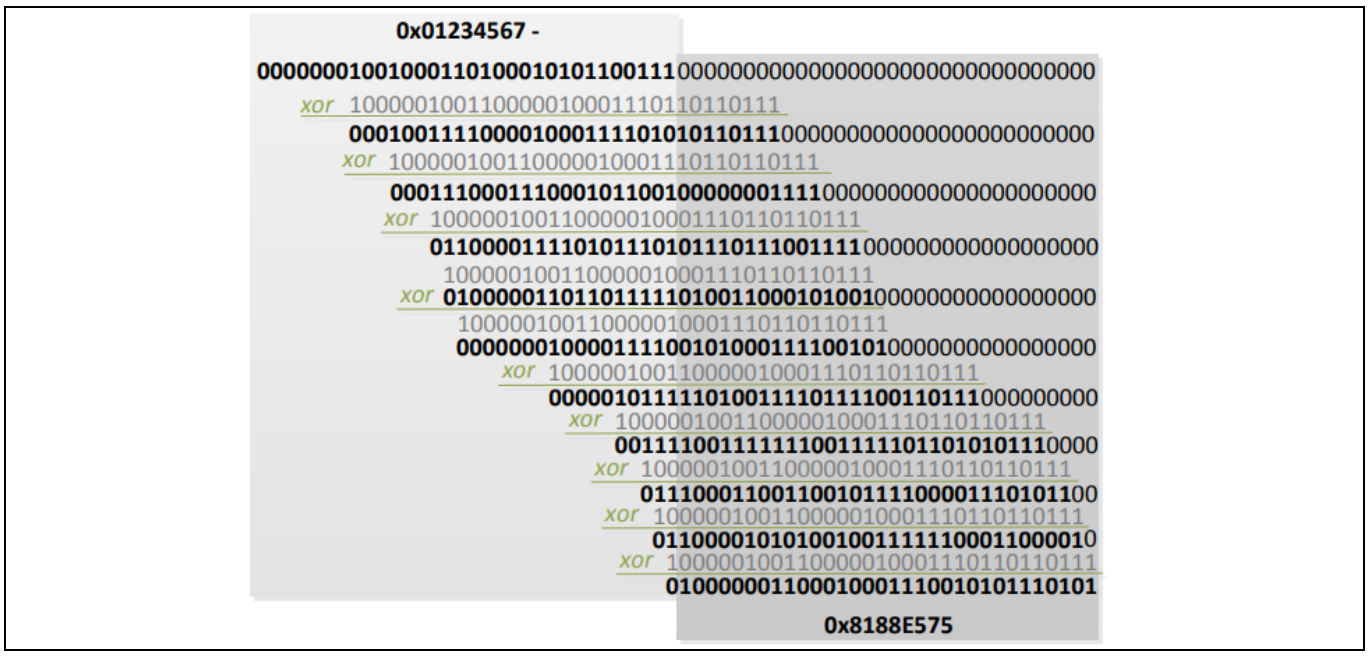

Figure 2　FCE カーネル 0/1 多項式を使用した 32 ビット CRC の計算例

上図では、左側に示されている元のデータがあります。データの最後に、32 ビットのシード値を追加します。N-1 は多項式のサイズ (この場合は 32 ビット) です。ここでは、追加されたシード値として 0x00000000 を使用します。シード値として 0xFFFFFFFF を使用することも一般的です。CRC の演算は、先頭の'1'を提供する各ビット位置で順次排他的論理和 (XOR) 演算を実行することです。データ全体で XOR 演算が完了した後、残りは計算された CRC 値です。XMC4000 デバイスでこの動作を確認できます。以下の **Figure 3** は、ハードウェアエンジンで実行されている上記の例を示します。

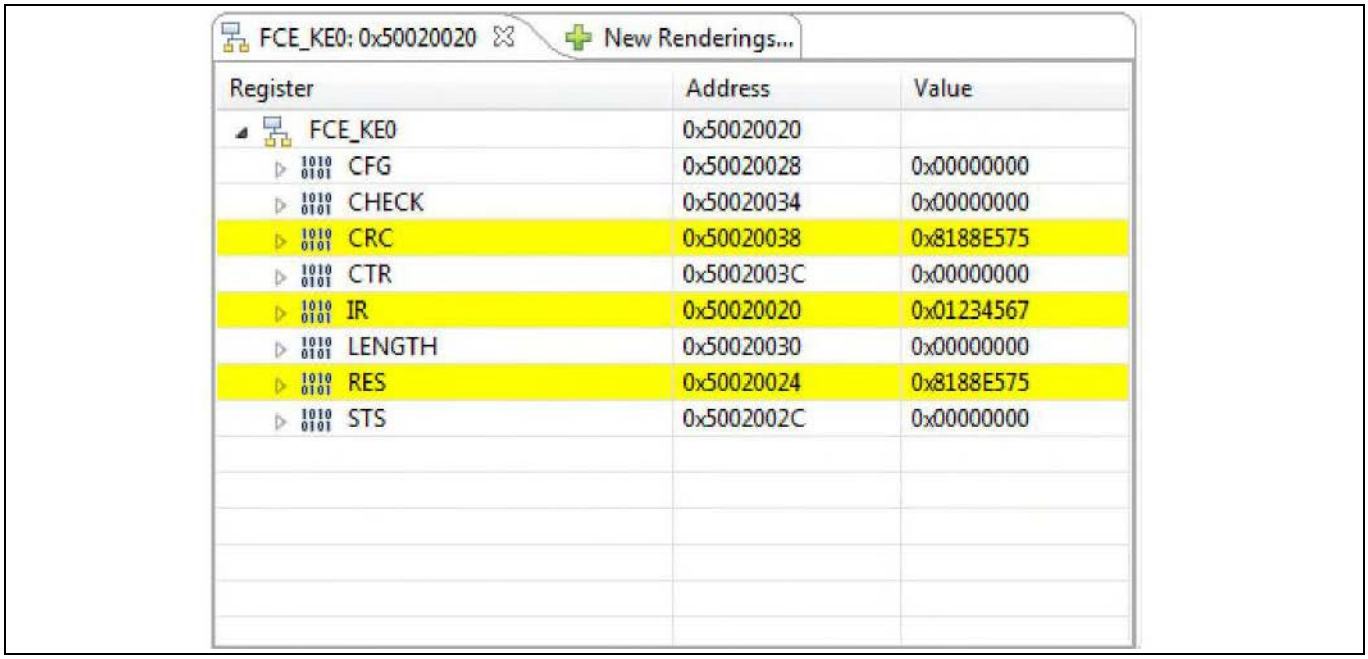

Figure 3　FCE ハードウェアで実行される 32 ビット CRC の計算例

FCE 入力レジスタ (IR) は、32 ビット幅の形式でデータを受け取ります。次に、ハードウェアは上記の操作を実行し、結果 (RES) を生成します。上記のように、データは **Figure 2** の計算例と一致します。

2 アプリケーションコードの調整

整合性チェックを実行してその正確性を確認するためには、使用可能な情報とともにアプリケーションコードを調整する必要があります。。主に、データのブロックのサイズを決定し、既知の CRC 値をアプリケーションコードに保存する必要があります。これにより、起動時または再フラッシュ時のチェックにおいて、良好なチェックが得られることが分かっています。

2.1 方法論

概略を説明したように、データ整合性 CRC を正確に計算するためには、2 つの項目を制御する必要があります。まず、決定的なアプリケーションサイズを知る必要があります。1 つの方法は、アプリケーションの最後にキーを配置することです。この一意の識別子キーを使用すると、ルーチンはキーを見つけてサイズを計算し、データの終わりを判断できます。もう 1 つの重要なステップは、フラッシュの特定のセクターまたは複数のセクターを埋めることです。セクターの終端点は通常、さまざまな機能 (つまり、アプリケーションコードとエミュレートされた EEPROM または他の形式のキャリブレーション) のフラッシュパーティションと一致します。特定のセクターに制限を設定することは、CRC を配置するためのクリーンな方法です。次に、比較を実行するために、アプリケーションに正しい CRC をインストールすることを検討する必要があります。計算を確認するために、これを既知の適切な値である hex ファイルの一部 (PC 側のツールによってインストールされた値) と比較する必要があります。これを行うには、既知のアドレスに正しい CRC を適用する必要があります。コンパイル時に正しい CRC 値がわからないため、後で CRC を計算し、アドレスの場所を正しい CRC 値に変更できる場所を割り当てる必要があります。アプリケーションは次のように表示されます。

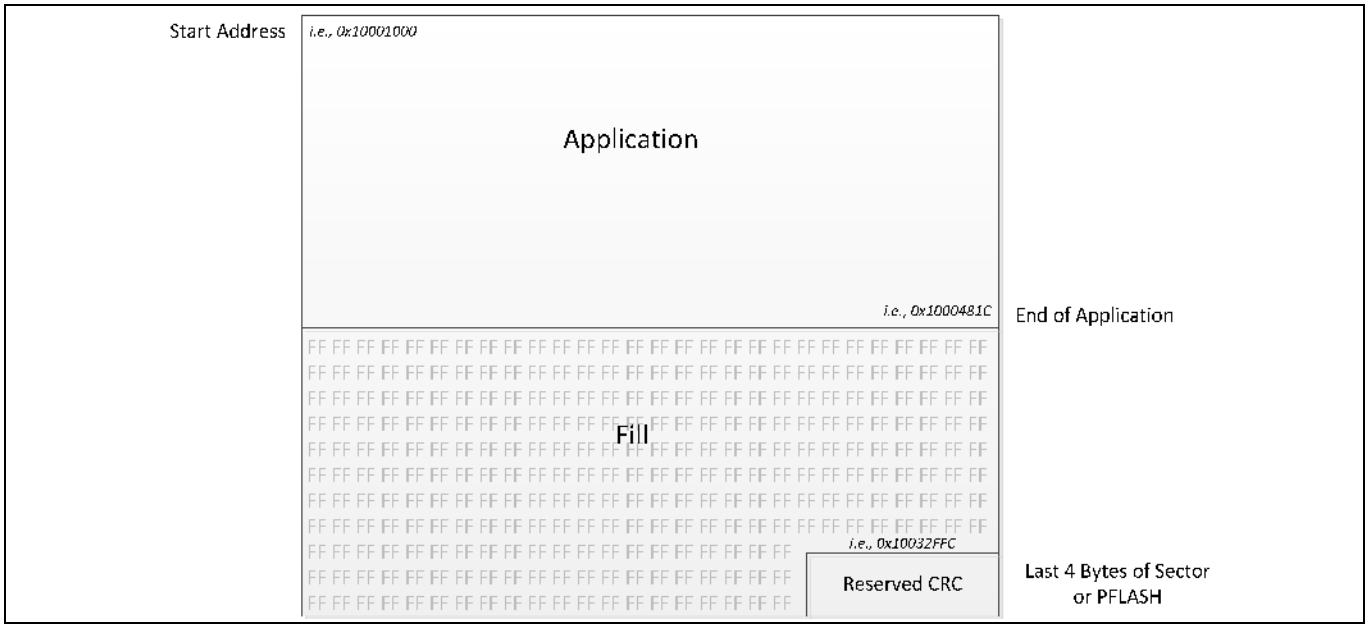

Figure 4 アプリケーションコードの高レベルの編成

2.1.1 DAVE™、SRecord、および XMC™ flasher の使用

前述のように、CRC を入力して生成するには、未加工のアプリケーションコードを調整する必要があります。コードを事前調整するには、CRC 値の場所を埋めて予約するユーティリティが必要です。これを行う 1 つの方法は、広く利用可能な SRecord ユーティリティ (http://srecord.sourceforge.net/) を使用することです。SRecord では、SRecord 形式の出力を追加、トリミング、塗りつぶすなどの方法で、いくつかの方法で操作できます。修正された形式の SRecord を取得したら、ファイルを操作して正しい CRC 値を含められます。これは、XMC™ Flasher を使用して実行できます。

2.1.2 SRecord の生成

ARM-GCC (DAVE™ v4 で使用されているコンパイラ) は、SRecord を出力する機能があります。そのためには、プロジェクトプロパティの基本的な操作を行う必要があります。次の設定は、DAVE™プロジェクト (www.infineon.com/dave) の設定方法を知っていることが前提です。SRecord を出力したいプロジェクトで、プロジェクトを右クリックして”Properties”を選択できます。

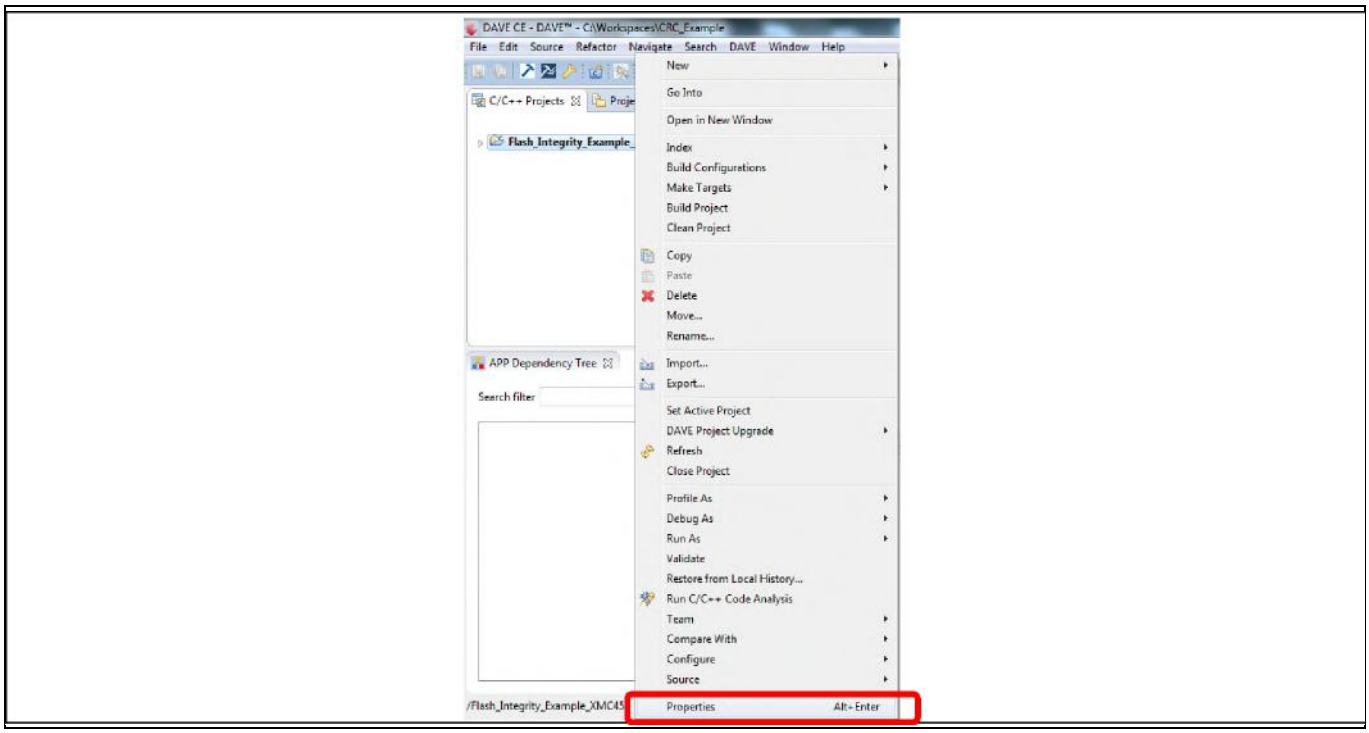

Figure 5 プロパティタブ

これにより、プロジェクトのプロパティを含むウィンドウが開きます。”Settings”の下に、GCC コンパイラ、リンカ、およびアセンブラに関連するプロパティの別の選択セットがあります。”ARM-GCC Create Flash Image”と小見出し”Output”の下に、フラッシュイメージの出力タイプを選択するオプションがあります。デフォルトでは、Intel Hex または**’ihex’**です。以下で説明するように、この選択を**’srec’**に変更できます。

まず、”Properties”に移動してください。これにより、プロジェクトを右クリックして、プロジェクトのプロパティを開けます。次に、”C / C ++ Build”と”Settings”に移動してください。”ARM-GCC Create Flash Image”の下で、”Output”を選択して**’srec’**に変更できます。

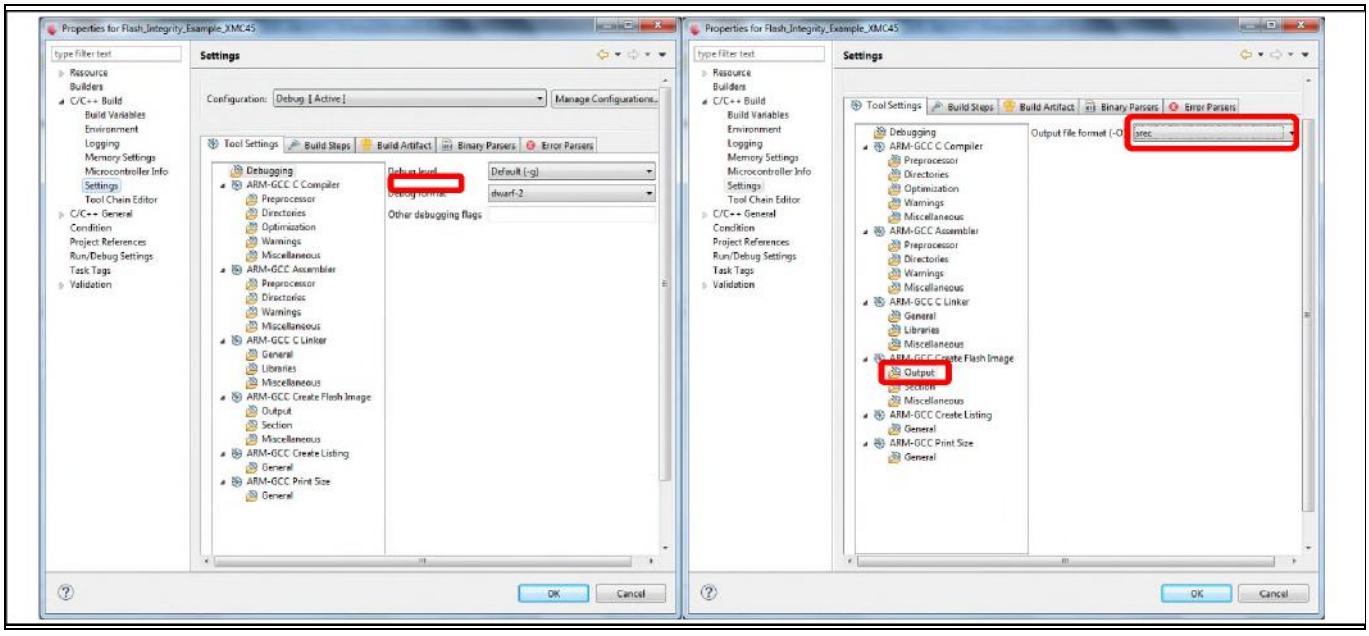

Figure 6　SRecord 出力のプロパティ設定

“ARM-GCC Create Flash Image”オプションページでは、コンパイラに優先出力ファイル名を通知する必要もあります。これには、”Command line pattern”の操作が必要です。

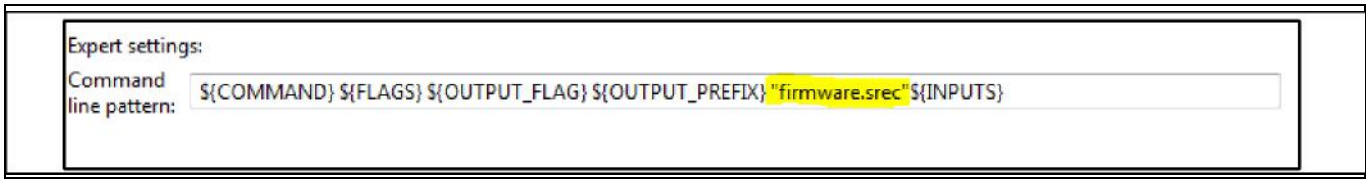

Figure 7　SRecord の出力設定

プロジェクトをビルドする時、SRecord ファイルが”Debug”フォルダに表示されます。SRecord ファイルは未加工のアプリケーションコードであり、フィルや CRC 値は追加されていません。

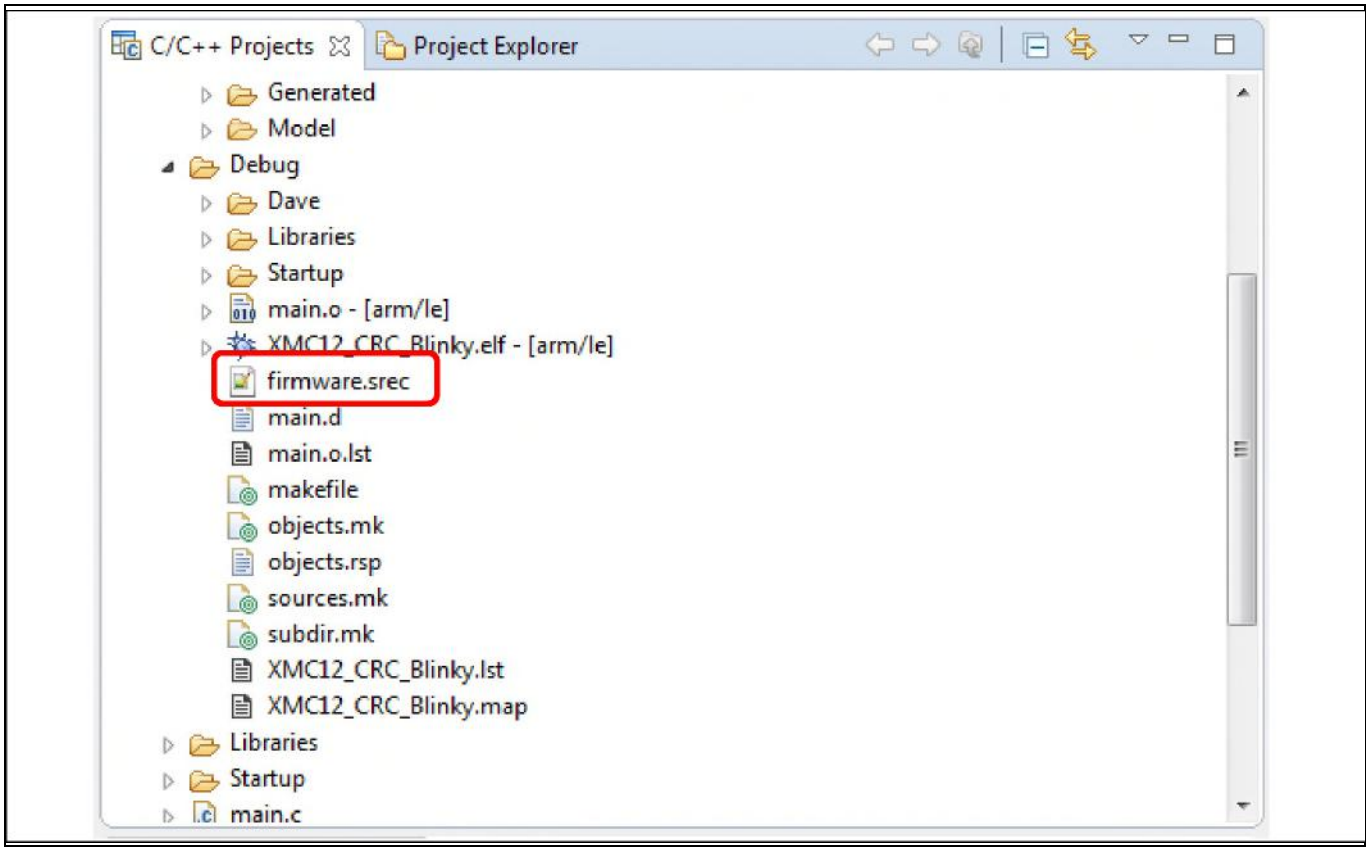

Figure 8　SRecord のデバッグ出力

2.1.3　SRecord のフォーマット

SRecord の生の出力が得られたので、目的の基準を満たすようにファイルを変更できます。まず、SRecord を、フラッシュのページやセクターなどの特定のフラッシュパーティションに入力します。これは、フラッシュの特定の境界のフラッシュの整合性をチェックできるので有利です。最良の塗りつぶしコンテンツに関してはいくつかの意見がありますが、一般にゼロ以外で塗りつぶすことを推奨します。これらの例は、オープンフラッシュスペースを'0xFF'で埋めます。これにより、セクター全体の整合性がチェックされることが保証されます (CRC によってゼロ位置が渡されます) が、アプリケーションの範囲外の条件で特別な操作は提供されません。場合によっては、操作を使用しないか、ステートメントをジャンプしてプログラムカウンタを特定のアドレスにプッシュすることが合理的です。最後に、CRC を保存するフラッシュワードを事前調整する必要があります。記録された CRC を作成するために使用するツールは、'マジック'ナンバーを使用します。これは、実際のアプリケーションコードに存在する可能性が非常に低い 32 ビット値です。これを行うために使用するツールである XMC™ flasher は、'マジック'ナンバー 'FECABAFA'を認識します。このツールは、PC 側の SRecord で CRC チェックを実行し、この'マジック'ナンバーを実際の CRC 計算に置き換えます。これは、アプリケーション内の CRC チェックとの静的な比較になります。

これを行うには、以下のコマンドを使用して、SRecord を使用してマジックキーを入力および生成できます。

srec_cat.exe application_firmware.srec -crop 0x10001000 0x10032FFC
-fill 0xFF 0x10001000 0x10032FFC
-generate 0x10032FFC 0x10033000
-constant-b-e 0xFECABAFA 4
-o new_application_firmware.srec

SRecord ツールが管理するコマンドをいくつか呼び出します。

- ‘application_firmware.srec’は入力ファイルです。
- “-crop”コマンドは、操作したい関連するアドレス領域のみにファイルをトリミングします。
- “-fill”コマンドは、未使用のアドレススペースをすべて取得し、値’0xFF’で埋めます。
- “-generate”コマンドは、アドレス範囲で特定の値を生成するようにツールに指示します。
- “-constant-be”コマンドは、”-generate”に次の定数 (oxFECABAFA) を 4 バイトにわたって適用するように指示します。
- “-o new_application_firmware.srec”は、new_application_firmware.srec への出力ファイルを定義します。

2.1.4 XMC™ flasher を使用した CRC の適用

これで、計算された CRC32 値のない条件付きアプリケーションファイルができました。CRC を計算するために、”-addchecksum”機能を介して XMC Flasher を使用できます。

XMCFlasher.jar -addchecksum firmware.srec



Figure 9 SRecord 調整のコマンドラインの例

ここで、塗りつぶしとマジックキー (srecord ファイルからの抜粋) を含む画像が見れます。

```
6399 S32510032FA0FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF18
6400 S32510032FC0FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF8
6401 S32510032FE0FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFECABAFA58
6402 S5031900E3
6403 S70510001019C1
```

そして CRC 後の計算 (srecord ファイルからの抜粋) です。

```
12796 S31510032FA0FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF18
12797 S31510032FB0FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF08
12798 S31510032FC0FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF8
12799 S31510032FD0FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFE8
12800 S31510032FE0FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFD8
12801 S31510032FF0FFFFFFFFFFFFFFFFFFFFFFFF365CCC2DE
12802 S70510001019C1
```

3 アプリケーション例

3.1 XMC1000 ファミリデバイス

サンプルアプリケーション”XMC12_CRC_Blinky”は、説明された画像ファイルの調整がターゲットデバイスでどのように機能するかの例を示します。この例では、最初にいくつかのインジケーターを設定します。P0.0 (XMC1200 ブートキットの LED 回路に接続される) は 5Hz のパルスであり、アプリケーションが正しく実行されていることをユーザーに知らせます。P0.2 は、CRC 計算が成功したか失敗したかを示すために使用される LED インジケーターでもあります。LED が OFF の場合は計算に失敗し、LED が ON の場合は計算に合格しています。合格条件は、計算が特定のフラッシュアドレスに PC に配置された CRC と一致することです。

このアプリケーションには、ブートキットのデバッガーを介した UART から USB への接続に構成された UART もあります。これを実現するには、UART DAVE™アプリを適切なピン (P1.3 Rx、P1.2 Tx) に構成し、それらをデバッガーの仮想 COM セットアップに関連付けます。そのため、LED と照合できるだけでなく、ターミナルウィンドウから値も確認できます。

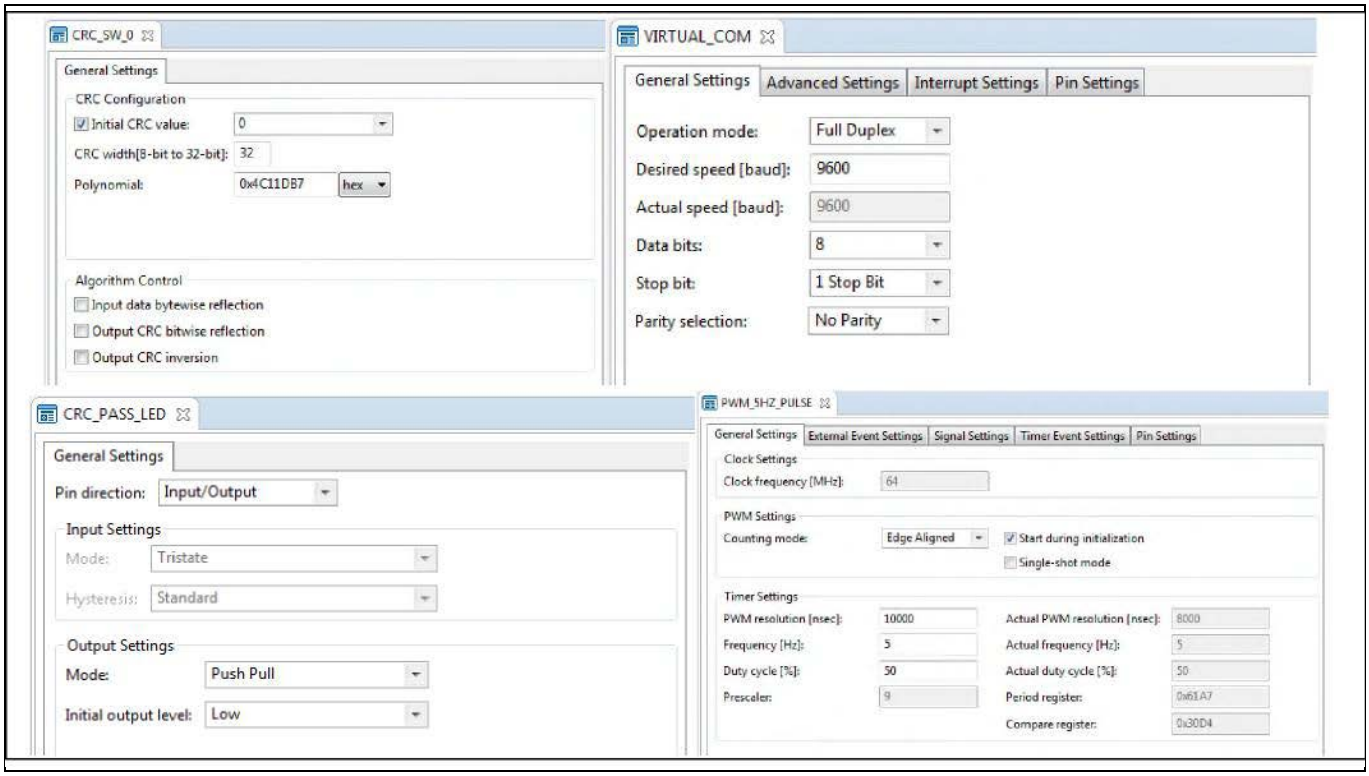

Figure 10 関連アプリの構成

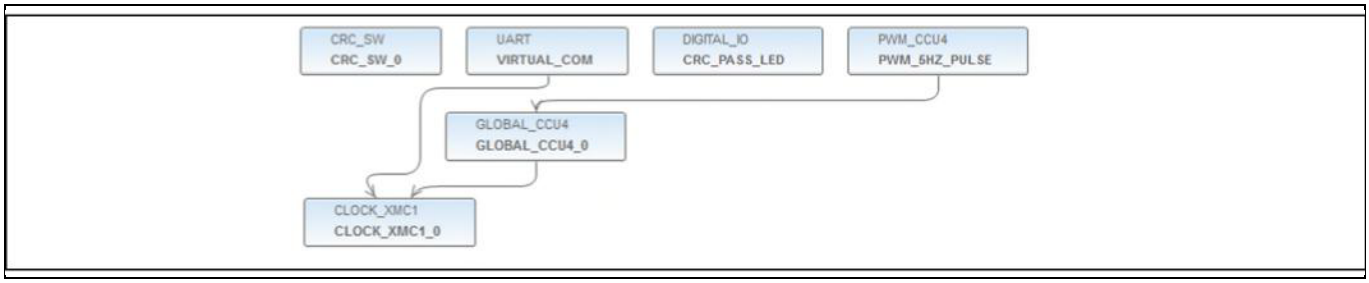

Figure 11 XMC12_CRC_Blinky の DAVE™アプリのセットアップ

プロジェクトにはドキュメントが付属します。プロジェクトを DAVE™にインポートすることでこれをテストできます (File->Import->Infineon->DAVE Project)。

ファイルをコンパイルして調整するために、プロジェクトは前述の方法で SRecord を出力するように調整されます。ファイルの調整を実行するバッチプロセスファイルも含まれます。このバッチファイルは、“srec_cat.exe”ユーティリティと“XMCFlasher.jar”ユーティリティの両方を利用します。これらは両方ともディレクトリに存在する必要があります。DAVE™ v4 から SRecord をコピーし、ファイル名を調整してバッチファイルを実行すると、フォーマットされた SRecord 出力が得られます。

```
srec_cat.exe XMC12_CRC_Blinky.srec          -crop 0x10001000 0x10033000
                                            -fill 0xFF 0x10001000 0x10032FFC
                                            -generate 0x10032FFC 0x10033000
                                            -constant-b-e 0xFECABAFA 4
                                            -o new_XMC12_CRC_Blinky.srec

XMCFlasher.jar -addchecksum new_XMC12_CRC_Blinky.srec
```

Figure 12　XMC12_CRC_Blinky バッチファイルの内容

アプリケーションを実行するために、XMC™ flasher を介してアプリケーションファームウェア (srec ファイル) をロードします。これを行うためには、XMC™ flasher ツールを開いてください。

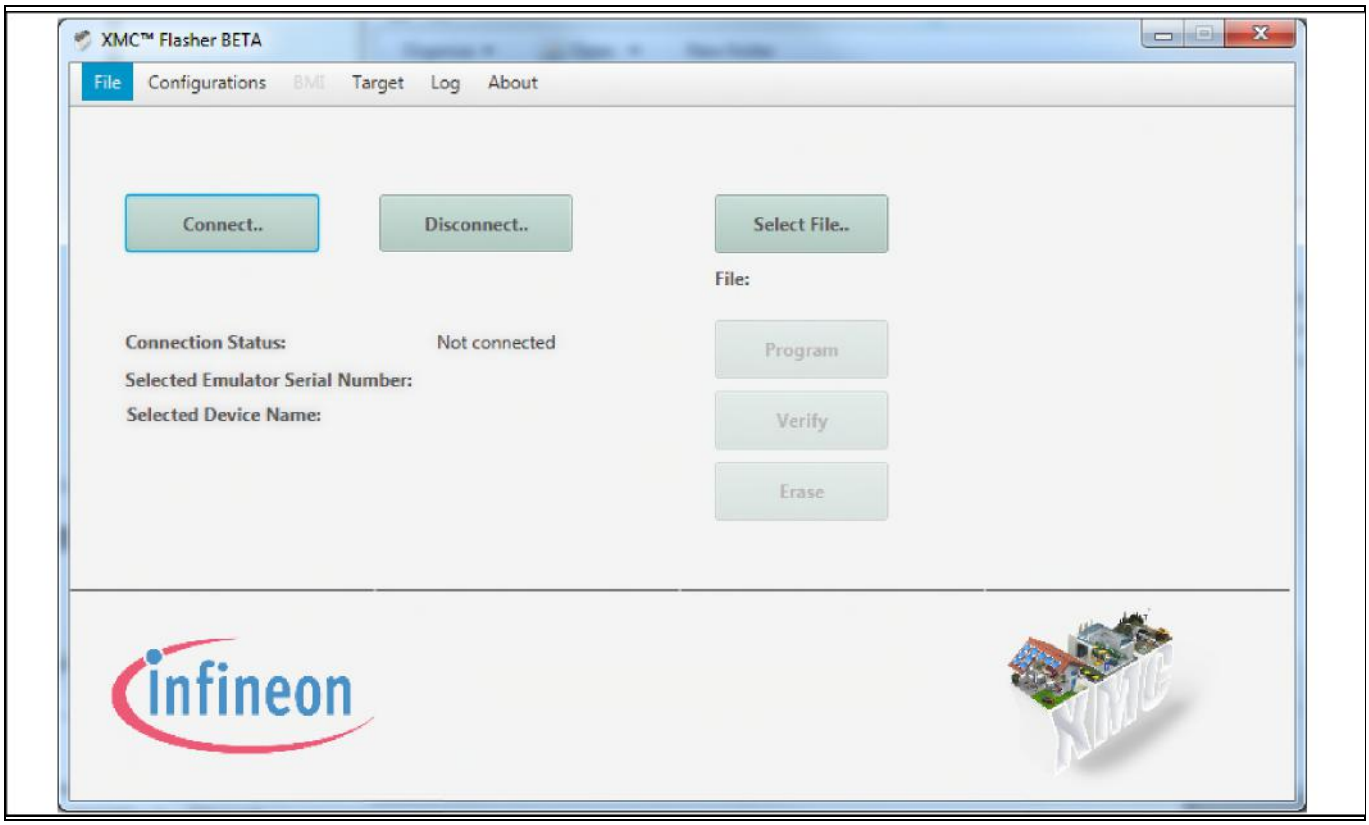

Figure 13　XMC™ flasher 接続プロセス

“Connect”を選択してください。デバイス選択ウィンドウが表示されます。”XMC1200-200“を選択してください。

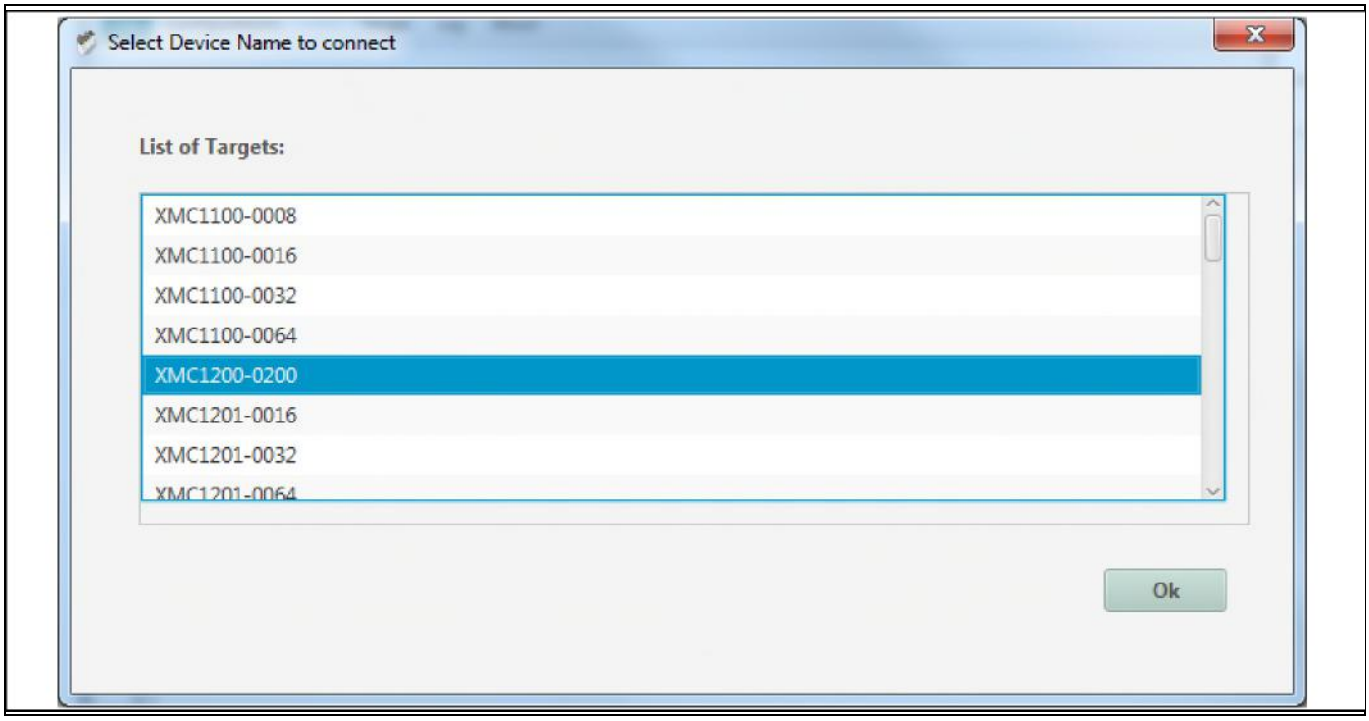

Figure 14　XMC™ flasher でのデバイス選択

これでターゲットに接続されました。“Select File”を選択し、*.srec ファイルを表示するには、”Motorola SREC-Files (*.srec、*.s19、*.mot、*.s) ”形式を選択する必要があります。ファイル ”new_XMC12_CRC_Blinky.srec” は、フォーマットされたフラッシュイメージです。

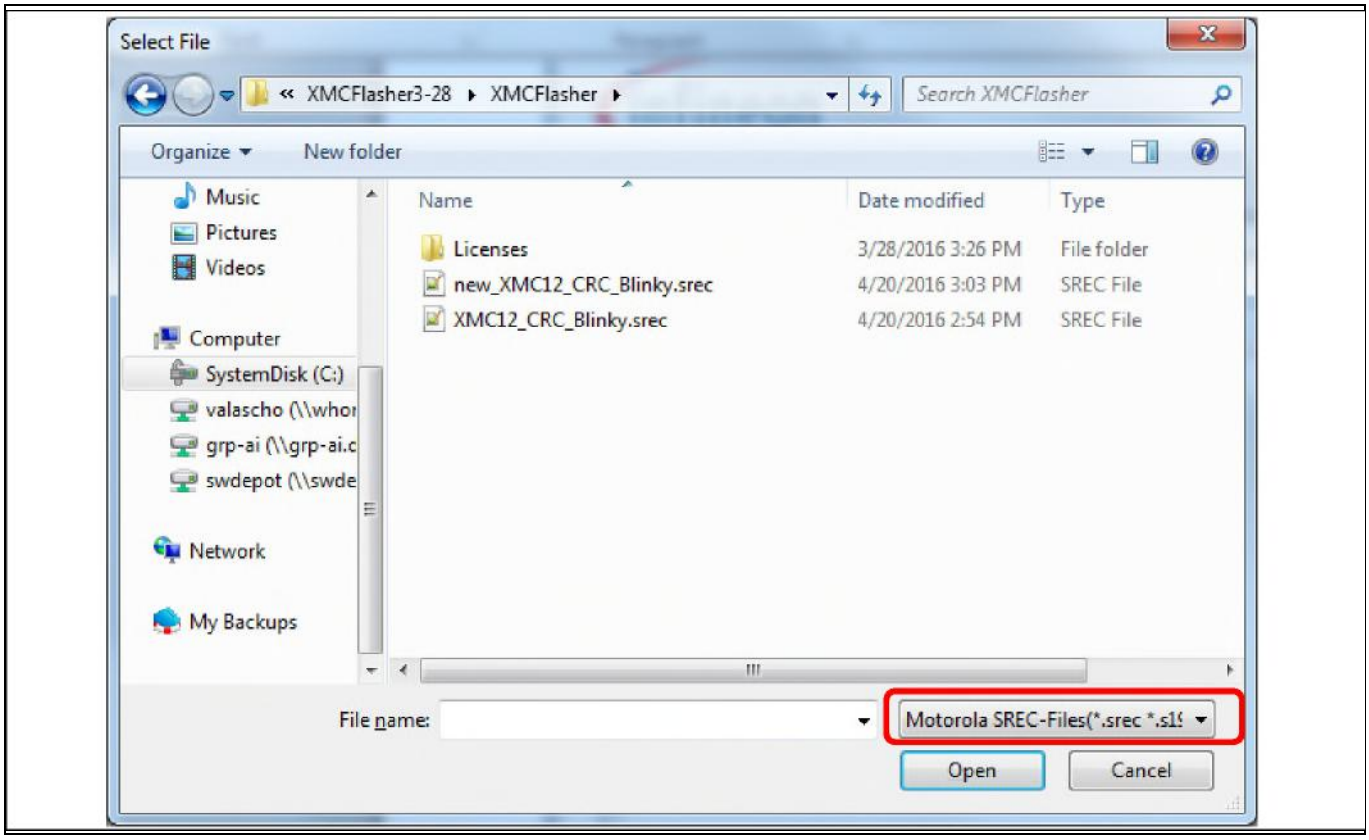

Figure 15 XMC™ flasher ファイルの選択

ファイルを開き、”Program”を選択してください。

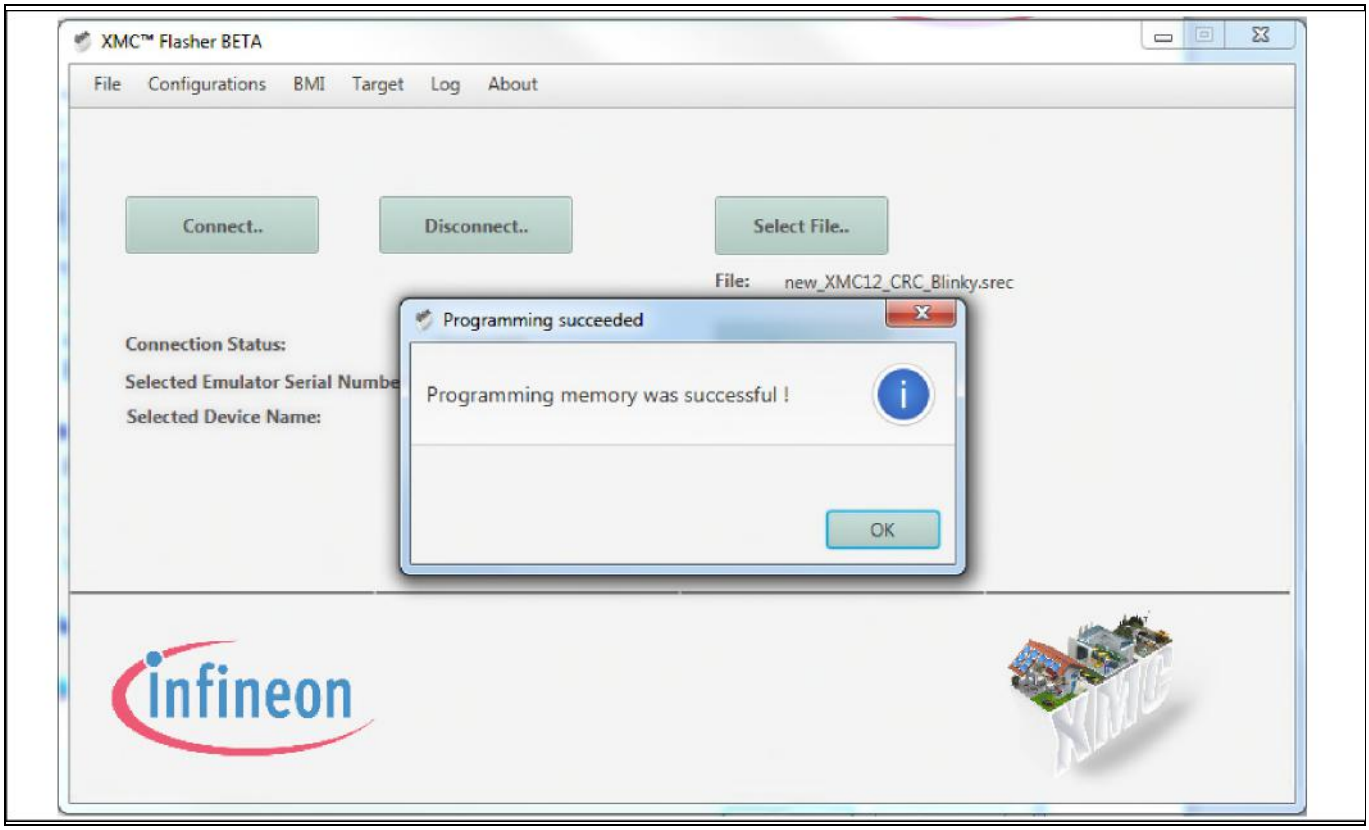

Figure 16 XMC™ flasher 成功プログラム

この時点で、アプリケーションは実行されており、LED インジケーターまたはターミナルウィンドウを介してプロセスを確認できるはずです。

ターミナルウィンドウを介して参照ください。

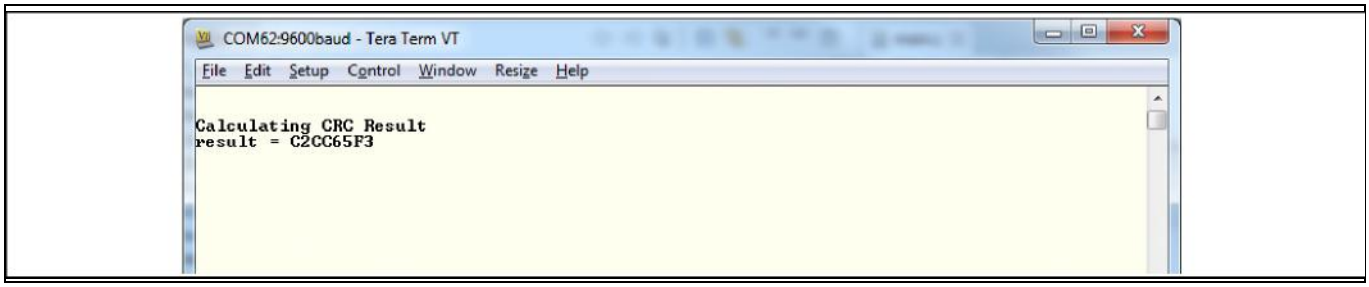

Figure 17 ターミナルウィンドウ出力

CRC プログラム例の実行時間は、32MHz クロックで実行される XMC1200 で 1.52 秒です。

3.2 XMC4000 ファミリデバイス

例として、”XMC45_CRC_Blinky”が提供されます。この例は、XMC4500 Relax または Relax Lite キットで実行されます。このボードにはデバッガーを介した仮想通信ポートがないため、ターゲットプロセッサから直接 USB VCOM セットアップを使用します。USB VCOM は、情報を 1 秒間隔でターミナルウィンドウに出力します。この例は、柔軟な CRC エンジンが含まれることを除けば、XMC1200 ベースの設計と非常によく似ています。

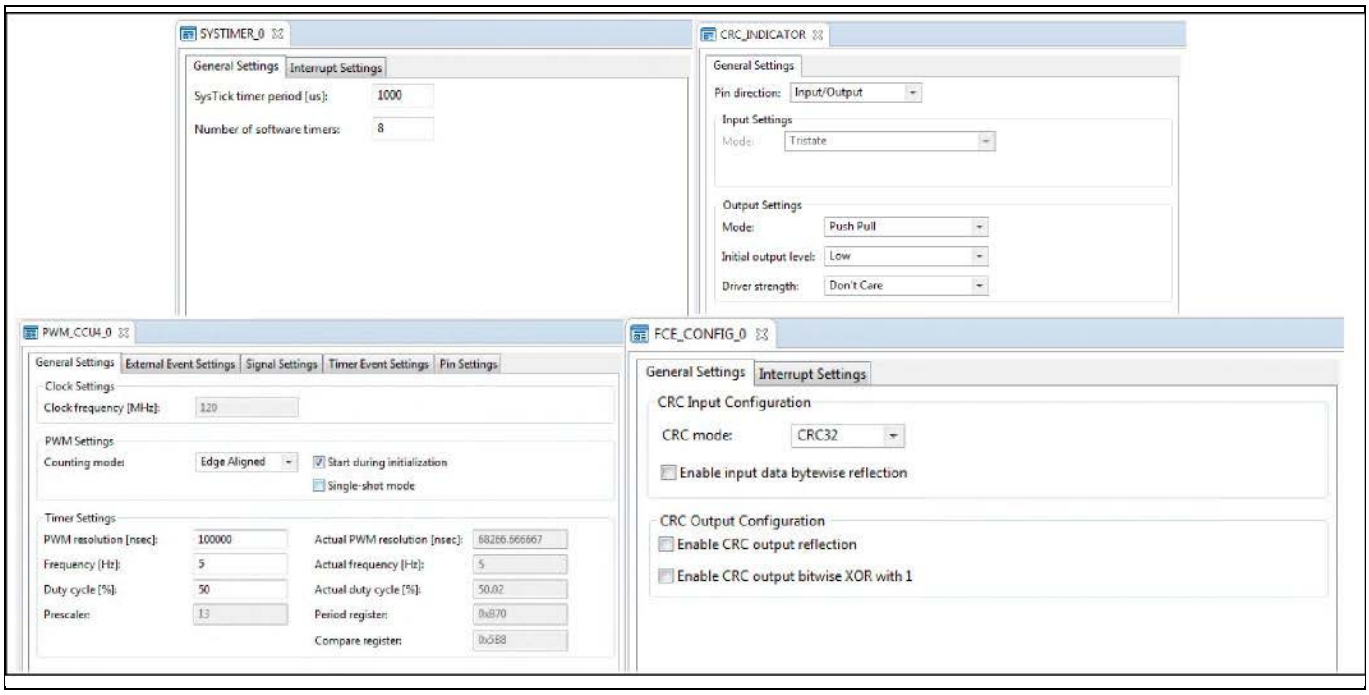

Figure 18 関連アプリの構成

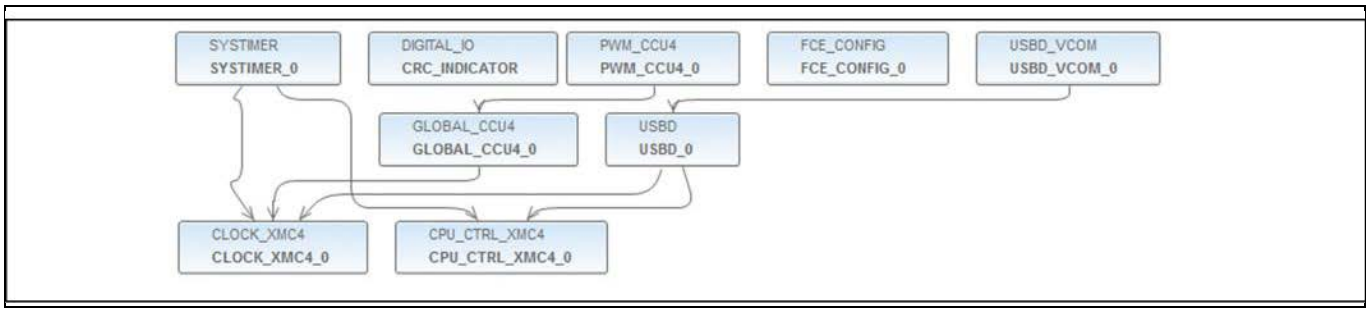

Figure 19 XMC45_CRC_Blinky の DAVE アプリの設定

XMC4000 ファミリは、いくつかの例外を除いて、基本的に XMC1000 ファミリと同じプロセスに従います。現在、ハードウェアアクセラレーションによるフレキシブル CRC エンジンを使用していて、結果としてこれを機能させるには、コードの調整を少し行う必要があります。XMC™プロセッサはリトルエンディアンです。CRC32 エンジンの他の主な用途は通信、特にイーサネットの管理であるため、FCE はビッグエンディアンです。したがって、違いを管理するために、いくつかの変換を行う必要があります。そのためのアルゴリズムがコードに配置されます。

```
//Change the endianess before write to FCE.
for (uint32_t index = 0; index < MEMORY_SIZE; index=index + 4)
{
            value_for_FCE = 0;
     for (int k = 0; k < 4; k++)
     {
     value_for_FCE |= ((uint32_t) (*(data_for_FCE + (index + k)))) << (8 * (3 - k));
     }
     //Push 4 bytes to CRC engine
     FCE_KE0->IR = (value_for_FCE);
}
for(uint8_t i=0; i<20; i++); //Small Delay to Insure Correct CRC Result
crc_result = FCE_KE0->RES;      //Read Result
```

Figure 20　FCE のエンディアン調整アルゴリズム

同じプロセスに従って、SRecord 出力を生成します。未加工のアプリケーションイメージをフォーマットされたバージョンに変換するためのバッチプロセスファイルが作成されます。XMC™ flasher を介してアプリケーションをロードすると、P1.0 LED が 5Hz の速度で点滅し、ターゲットが実行中であることを示します。P1.1 は、CRC が確認に合格したことを示す LED 表示です。再度、ターミナルウィンドウを介して参照できます (ターゲットに接続されている USB ポートを介して接続し、PC 側に”driver.inf”をインストールしていることを確認してください。USB_VCOM DAVE 生成ソフトウェアの一部 'Dave\Generated\USBD_VCOM\inf ')。

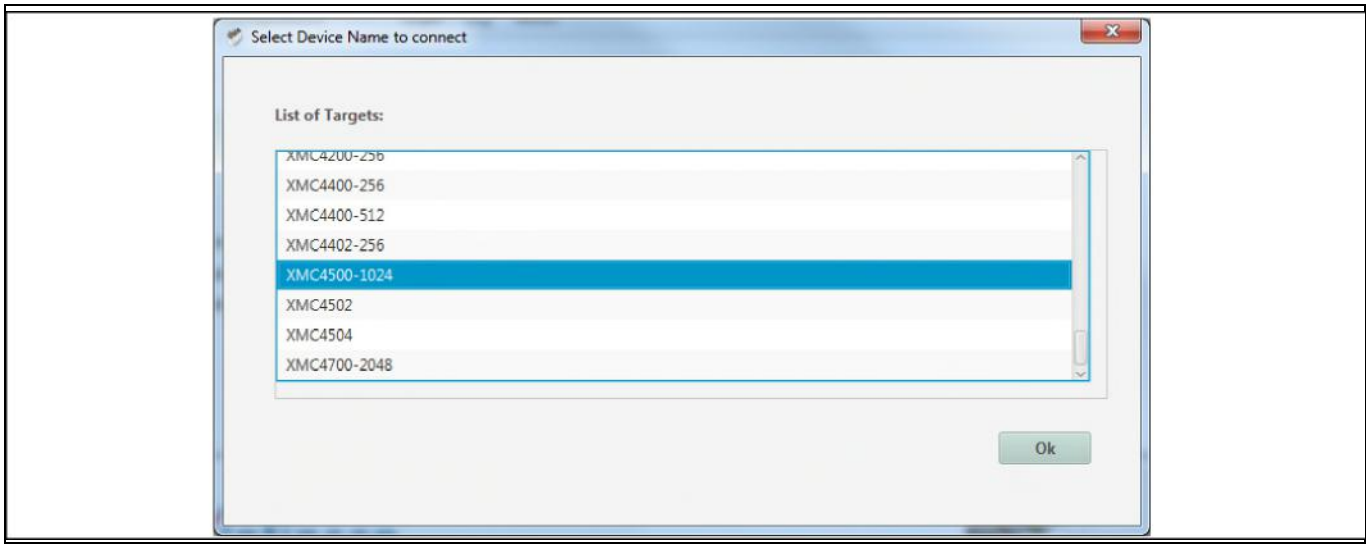

Figure 21　XMC™ flasher デバイスの選択

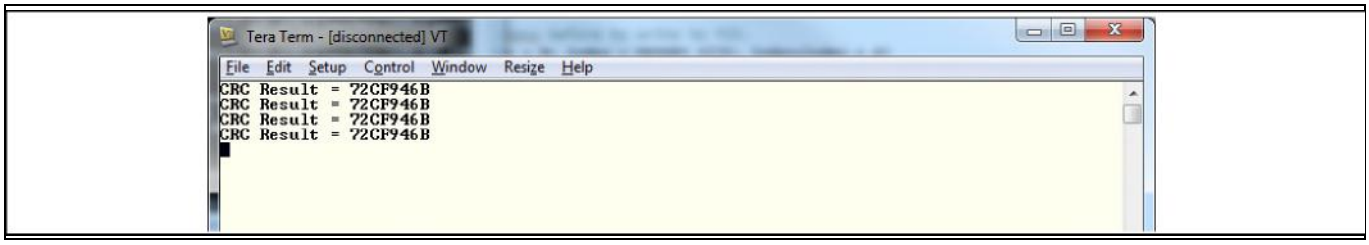

Figure 22　ターミナルウィンドウの結果

CRC プログラム例の実行時間は、120MHz クロックの XMC4500 で 1.8 ミリ秒です。

改訂履歴

Document version	Date of release	Description of changes
1.0	2021-12-15	本版は英語版 AP32339 Managing firmware integrity in XMC™, XMC1000 and XMC4000 family devices V1.0 について、CYPRESS DEVELOPER COMMUNITY の参画者によって日本語に翻訳されたドキュメントです。

Trademarks
All referenced product or service names and trademarks are the property of their respective owners.

Edition 2021-12-15
Published by
Infineon Technologies AG
81726 Munich, Germany

© 2021 Infineon Technologies AG.
All Rights Reserved.

Do you have a question about this document?
Email: erratum@infineon.com

Document reference
AP32339

重要事項

本文書に記載された情報は、いかなる場合も、条件または特性の保証とみなされるものではありません（「品質の保証」）。本文に記された一切の事例、手引き、もしくは一般的価値、および／または本製品の用途に関する一切の情報に関し、インフィニオンテクノロジーズ（以下、「インフィニオン」）はここに、第三者の知的所有権の不侵害の保証を含むがこれに限らず、あらゆる種類の一切の保証および責任を否定いたします。

さらに、本文書に記載された一切の情報は、お客様の用途におけるお客様の製品およびインフィニオン製品の一切の使用に関し、本文書に記載された義務ならびに一切の関連する法的要件、規範、および基準をお客様が遵守することを条件としています。

本文書に含まれるデータは、技術的訓練を受けた従業員のみを対象としています。本製品の対象用途への適合性、およびこれら用途に関連して本文書に記載された製品情報の完全性についての評価は、お客様の技術部門の責任にて実施してください。

本製品、技術、納品条件、および価格についての詳しい情報は、インフィニオンの最寄りの営業所までお問い合わせください（**www.infineon.com**）。

警告事項

技術的要件に伴い、製品には危険物質が含まれる可能性があります。当該種別の詳細については、インフィニオンの最寄りの営業所までお問い合わせください。

インフィニオンの正式代表者が署名した書面を通じ、インフィニオンによる明示の承認が存在する場合を除き、インフィニオンの製品は、当該製品の障害またはその使用に関する一切の結果が、合理的に人的傷害を招く恐れのある一切の用途に使用することはできないこと予めご了承ください。