デジタル-アナログコンバーター(DAC) XMC4000

産業用32ビットマイクロコントローラシリーズ

About this document

Scope and purpose

このドキュメントでは、DAC ペリフェラルの機能と、データ処理、パターン生成、ノイズ生成などの一般的なケースに合わせて DAC ペリフェラルを構成する方法について説明します。

対象製品

XMC4000 マイクロコントローラーファミリ

参考資料

[1] インフィニオン: サンプルコード: **http://www.infineon.com/XMC4000**, Documents タブ

[2] インフィニオン: XMC Lib、**http://www.infineon.com/DAVE**

[3] インフィニオン: DAVE™、**http://www.infineon.com/DAVE**

[4] インフィニオン: XMC リファレンスマニュアル、**http://www.infineon.com/XMC4000**, Documents タブ

[5] インフィニオン: XMC データシート、**http://www.infineon.com/XMC4000**, Documents タブ

Table of contents

About this document **1**
Table of contents **1**
1 デジタル-アナログコンバーター (DAC) **3**
1.1 XMC DAC 動作モード 3
1.2 XMC DAC 入力タイプ 4
1.3 XMC DAC データの調整 4
1.4 基本的な DAC 方程式 5
2 単一値モード **6**
2.1 XMC Lib の設定 6
2.2 XMC Lib の初期化 6
2.3 関数の実装 7
3 データ処理モード **8**
3.1 XMC Lib 設定 8
3.2 XMC Lib の初期化 8
3.3 関数の実装 9
4 パターンジェネレータモード **10**
4.1 XMC Lib 設定 11
4.2 XMC Lib の初期化 11
5 ノイズジェネレータモード **12**

Table of contents

5.1 XMC Lib 設定 12
5.2 XMC Lib の初期化 13
6 ランプジェネレータモード 14
6.1 XMC Lib 設定 15
6.2 XMC Lib の初期化 15
改訂履歴 16

1 デジタル-アナログコンバーター (DAC)

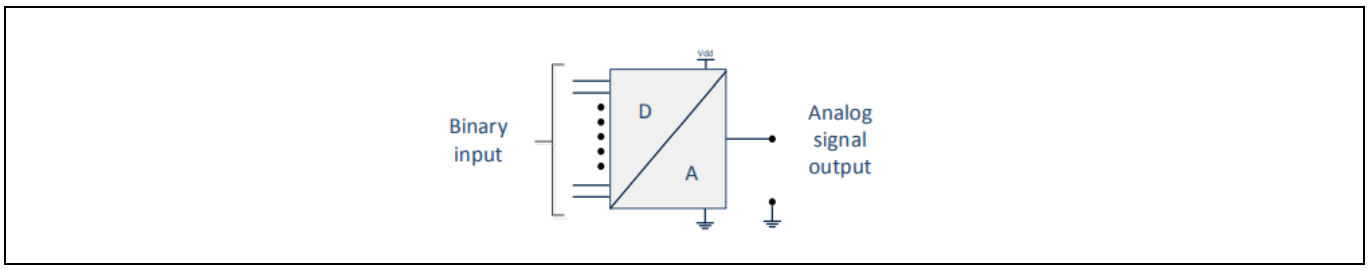

Figure 1 デジタル-アナログコンバーター

デジタル-アナログコンバーターは、1 ビットまたは複数ビットのデジタル値をアナログ値に変換するモジュールです。これは次のように定義されます。

- 解像度
- 変換速度
- オフセット
- アナログ出力の精度
- ドライブ機能

DAC のリフィル (出力値の設定) 方法は、システムの性能に影響を与えます。CPU の負荷を軽減するために、DMA (Direct Memory Access) による DAC の設定や、DAC モジュール自体に備えられた波形生成部によるリフィルが可能です。また、一部のアプリケーションでは、静的なオフセットや乗算など、データの精密化が必要です。

1.1 XMC DAC 動作モード

DAC モジュールは 5 つの動作モードを提供します。

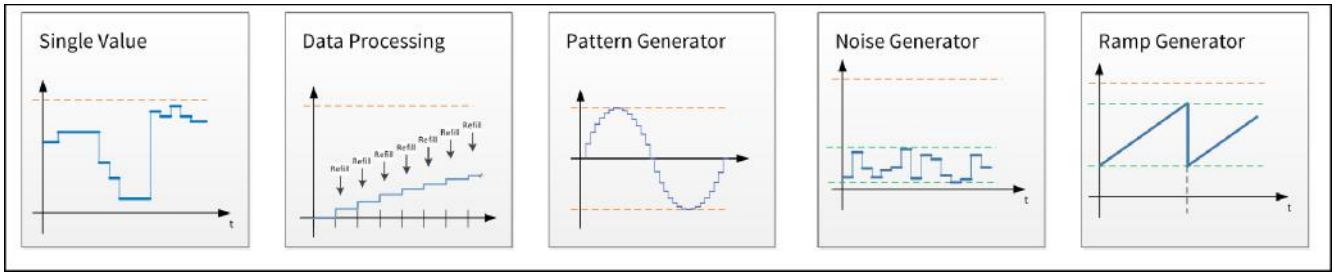

Figure 2 DAC モードの概要

CPU 負荷を軽減するために、モードはリフィル方法で区別されます。動作モードについては、次のセクションで説明します。

1.2 XMC DAC 入力タイプ

動作モードごとに、特定の入力タイプを使用できます。

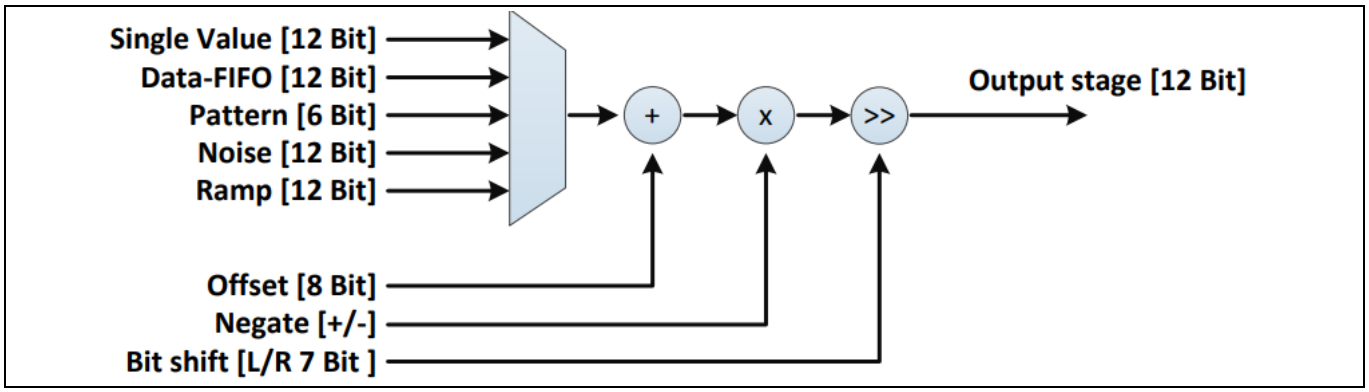

Figure 3 DAC 入力タイプとデータ操作

単一値 (Single Value) 入力は、アナログ出力への高速アクセスを提供します。この入力は、非周期的またはソフトウェア制御の出力に使用されます。

Data-FIFO 入力により、フィル (fill) ステージと DAC 出力ステージを分離できます。この入力は、定期的な更新に使用されます。リフィルは、DMA トリガを介してソフトウェア (SW) またはハードウェア (HW) で実行できます。

パターン (Pattern) ジェネレータには 6 ビットの出力があります。カスタマイズされた対称パターンを生成できます。パターンの最初の 4 分の 1 はユーザーが定義します。

ノイズ (Noise) ジェネレータは、12 ビットの分解能で疑似ランダム値を提供します。

ランプ (Ramp) ジェネレータは、12 ビットの分解能でランプを提供します。このランプでは、値はクロックサイクルごとに増加し、開始値と停止値はユーザーが定義できます。

5 つの入力は、DAC の 5 つのモードも表します

- 単一値入力: 単一値モード
- Data-FIFO 入力: データ処理モード
- パターン入力: パターンジェネレータモード
- ノイズ入力: ノイズジェネレータモード
- ランプ入力: ランプジェネレータモード

1.3 XMC DAC データの調整

出力段は、アナログ値に変換される 12 ビット値です。符号なしまたは符号付きの値として使用できます。符号なしモードでは、0 は最小のアナログ値として使用されます。符号付きモードでは、0 が中心電圧を生成します。データは、入力ステージと**出力ステージ**の間で調整できます。調整オプションは、**オフセット**、**反転**、**ビットシフト**です。

XMC は、入力に追加される 8 ビットのオフセットを提供します。オフセットの符号判定は、出力ステージに従います。

反転ビットは、入力の符号を反転します。これは、**単一値**または**データ処理モード**で HW 反転として使用したり、**パターンジェネレータモード**で負の半波で開始したり、**ランプジェネレータモード**で立ち下りランプを生成したりできます。**反転**ビットは、特定のデバイスでのみ使用できます。

ビットシフトにより、入力の乗算または除算が可能になります。これは、入力を 2^1〜2^7 [2 ... 128] で乗算または除算できることを意味します。これは、次のユースケースに使用できます。

- **単一値**または**データ処理モード**でのスケーリング
- パターンを**パターンジェネレータモード**から DAC 出力範囲にスケーリングします
- **ノイズジェネレータモード**でスケールを小さくする
- **ランプジェネレータモード**の最小・最大周波数を拡大

1.4 基本的な DAC 方程式

符号なし出力ステージからアナログ値への変換は、次の計算で実行できます。

- DEC_{target} DAC 出力レジスタのターゲット 10 進数値
- S_DEC_{target} DAC 出力レジスタのターゲット符号付き 10 進数値
- DEC_{res} DAC の 10 進数分解能
- V_{target} ターゲット DAC 出力電圧
- V_{max} 最大 DAC 出力電圧
- V_{offset} DAC オフセット電圧

DAC_{res}、V_{max}、および V_{offset} は、データシートに記載されています。

$$DEC_{target} = \frac{V_{target} - V_{offset}}{V_{max} - V_{offset}} * DEC_{res}$$

$$V_{target} = \left(DEC_{target} * \frac{(V_{max} - V_{offset})}{DEC_{res}}\right) + V_{offset}$$

出力ステージが符号付き設定の場合、次の計算を使用できます。

$$S_DEC_{target} - \left(\frac{V_{target} - V_{offset}}{V_{max} - V_{offset}} * DEC_{res}\right) - \frac{DEC_{res} + 1}{2}$$

$$V_{target} = \left(\left(S_DEC_{target} + \frac{DEC_{res} + 1}{2}\right) * \frac{(V_{max} - V_{offset})}{DEC_{res}}\right) + V_{offset}$$

次の例は、V_{max} = 2.5V、V_{offset} = 0.3V、DAC_{res} = (2^{12}-1) の式を示します。

$$DEC_{target} = \frac{V_{target} - 0{,}3V}{2{,}2V} * 4095$$

$$V_{target} = \left(DEC_{target} * \frac{2{,}2V}{4095}\right) + 0{,}3V$$

$$S_DEC_{target} = \left(\frac{V_{target} - 0{,}3V}{2{,}2V} * 4095\right) - 2048$$

$$V_{target} = \left((S_DEC_{target} + 2048) * \frac{2{,}2V}{4095}\right)$$

2 単一値モード

単一値モードは、単純なデジタルからアナログへの変換に使用できます。DATA レジスタに書き込むと、アナログ出力が変化します。DAC は符号なしデータまたは符号ありのデータを受け入れるように変更できます。データ改良 (オフセット、スケーリング、反転) が利用可能です。

チャネル 0 とチャネル 1 が使用されており、同時に更新する必要がある場合は、同期機能を使用できます。このモードでは、両方のチャネルのデータを 1 つのレジスタに書き込めます。チャネル 0 のトリガで、両方の値が更新されます。トリガは、ソフトウェアトリガ、または元のチャネル 0 データレジスタへの疑似書き込みアクセスのいずれかです。

2.1 XMC Lib の設定

この例は、シングルバリューモードの 2 つの DAC 設定を示します。DAC はソフトウェアでデータ書き込みされます。新しい値がデータレジスタに書き込まれると、次の DAC クロックで出力されます。チャネルは同期されません。最初のチャネルは符号なし用に設定され、2 番目のチャネルは符号付き値用に設定されます。どちらのチャネルも、スケーリングおよび反転機能を使用しません。両方のチャネルは、対応する設定で初期化されます。チャネル 0 は符号なし DAC、チャネル 1 は符号付き DAC です。

```
#define DAC_CH_NR_0       0U
#define DAC_CH_NR_1       1U

XMC_DAC_CH_CONFIG_t ch_config0=
{
   .output_offset  =   0U,
   .data_type      =   XMC_DAC_CH_DATA_TYPE_UNSIGNED,
   .output_scale   =   XMC_DAC_CH_OUTPUT_SCALE_NONE,
   .output_negation =  XMC_DAC_CH_OUTPUT_NEGATION_DISABLED,
};

XMC_DAC_CH_CONFIG_t ch_config1=
{
   .output_offset  =   0U,
   .data_type      =   XMC_DAC_CH_DATA_TYPE_SIGNED,
   .output_scale   =   XMC_DAC_CH_OUTPUT_SCALE_NONE,
   .output_negation =  XMC_DAC_CH_OUTPUT_NEGATION_DISABLED,
};
```

2.2 XMC Lib の初期化

DAC チャネルは一般に初期化されてから、特定のモードに設定されます。

```
XMC_DAC_CH_Init(XMC_DAC0, DAC_CH_NR_0, &ch_config0);
XMC_DAC_CH_Init(XMC_DAC0, DAC_CH_NR_1, &ch_config1);
```

初期化後、DAC モードは「単一値モード」に変更されます。

```
XMC_DAC_CH_StartSingleValueMode(XMC_DAC0, DAC_CH_NR_0);
XMC_DAC_CH_StartSingleValueMode(XMC_DAC0, DAC_CH_NR_1);
```

2.3 関数の実装

そして、DAC チャネルは値 0 で設定されます。チャネル 0 は符号なしとして設定されるため、ゼロは最小電圧として解釈されます。チャネル 1 が符号付きとして設定されている間、値ゼロは中心電圧として解釈されます。チャネル 0 には符号なし 12 ビット値を入力し、チャネル 1 には符号付き 12 ビット値を入力する必要があります。XMC_DAC_CH_Write() 関数は、DAC 変換を更新してトリガするために、定期的に呼び出せます。呼び出し頻度は、最大 DAC 更新頻度を超えてはいけません。

```
XMC_DAC_CH_Write(XMC_DAC0, DAC_CH_NR_0, 0x0000U);
XMC_DAC_CH_Write(XMC_DAC0, DAC_CH_NR_1, 0x0000U);
```

3 データ処理モード

データ処理モードは、DAC の割込みベースのリロードをサポートすることを期待する単一値モードに似ています。リロードは変換をトリガせず、DMA または SW を介して実行できます。

データ処理モードでは、データがフラッシュまたは RAM に保存される、波形生成用の大きくて正確なテーブルが可能です。

3.1 XMC Lib 設定

この例は、データ処理モードでの DAC 設定を示します。トリガが発生すると、DAC は割り込みサービスルーチン (ISR) のソフトウェアによってフィルされます。DAC モジュールはそれ自体で 10kHz のトリガを生成します。

```
#define DAC_CH_NR_0        0U
#define Trigger_ISR DAC0_0_IRQHandler


XMC_DAC_CH_CONFIG_t ch_config0=
{
   .output_offset    =     0U,
   .data_type        =     XMC_DAC_CH_DATA_TYPE_UNSIGNED,
   .output_scale     =     XMC_DAC_CH_OUTPUT_SCALE_NONE,
   .output_negation  =     XMC_DAC_CH_OUTPUT_NEGATION_DISABLED,
};
```

3.2 XMC Lib の初期化

DAC チャネルは一般に初期化されてから、特定のモードに設定されます。

```
XMC_DAC_CH_Init(XMC_DAC0, DAC_CH_NR_0, &ch_config0);
```

一般的な初期化の後、DAC をデータ処理モード (DataMode) に設定する必要があります。DAC トリガは DAC モジュールによって生成されます。したがって、周波数は 10kHz に設定されます。外部トリガを使用する場合、周波数の選択は不要であり、ゼロに設定できます。DAC が初期化されると、DAC イベントとサービス要求が有効になります。

```
XMC_DAC_CH_StartDataMode(XMC_DAC0,DAC_CH_NR_0,
                         XMC_DAC_CH_TRIGGER_INTERNAL,
                         10000U);


XMC_DAC_CH_EnableEvent(XMC_DAC0,DAC_CH_NR_0);
NVIC_EnableIRQ(DAC0_0_IRQn);
```

3.3 関数の実装

DAC0 ISR では、データ表からの新しい値が DATA レジスタに書き込まれます。

```
void DAC0_0_IRQHandler(void)
{
        XMC_DAC_CH_Write(XMC_DAC0,DAC_CH_NR_0,table_u[table_i]);
        table_i++;
        if(table_i>=LENGTH_OF_TABLE)
        {table_i=0;}
};
```

4 パターンジェネレータモード

パターンジェネレータモードは、ソフトウェアをロードせずにパターンを生成します。波形の 4 分の 1 を定義する必要があります。残りの波形は、最初の 4 分の 1 の水平および垂直ミラーリングによって生成されます。

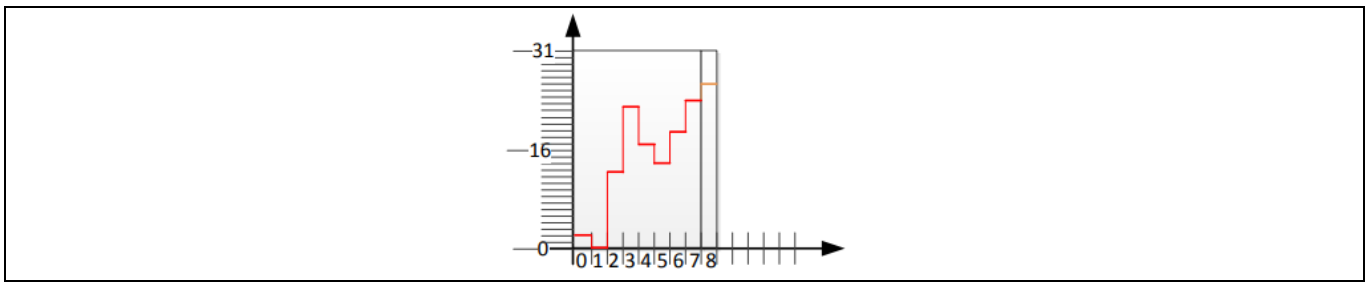

Figure 4 DAC パターン設定

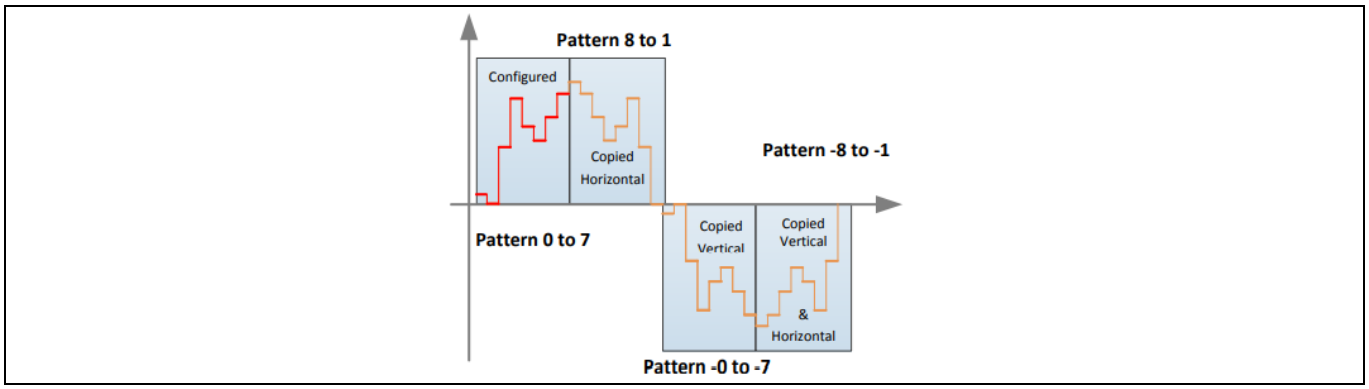

Figure 5 DAC パターン設定

波形は 32 ステップの長さ分解能と 64 ステップの振幅分解能に固定されます。6 ビットの振幅分解能は、12 ビットの DAC 出力に到達するようにスケーリングできます。

反転を有効にすると、波形は反転し始めます。

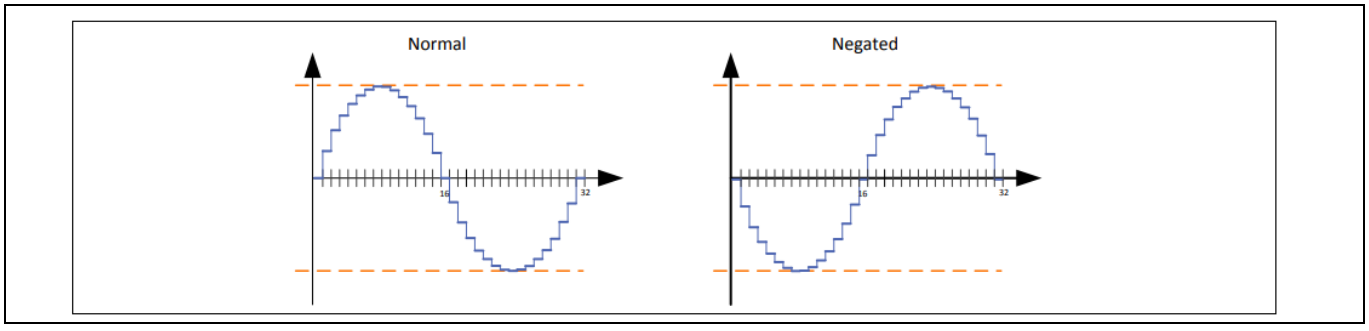

Figure 6 DAC パターンの反転

オフセットを追加することもできます。オフセットはパターンジェネレータスケール (6 ビット) です。スケーリングが追加された場合、パターンはオフセットを追加した後にスケーリングされます。

パターンジェネレータは、サイン出力も提供します。これを使用して ADC をトリガしたり、ERU を介してルーティングしてタイマなどの他の周辺機器をトリガしたりできます。信号は、ERU を介して GPIO にルーティングもできます。

4.1 XMC Lib 設定

この例は、DAC をパターンジェネレータモードで設定する方法を示します。チャネル 0 は、500Hz の正弦波を生成するように設定されます。パターンジェネレータの出力は、6 ビットの符号付きデータです。したがって、DAC は符号付きモードで設定されます。12 ビット DAC スケールに到達するには、データに 64 を掛ける (6 ビットシフトする) 必要があります。オフセットも反転も使用されません。

チャネル 0 は、対応する設定で初期化されます。

```
#define DAC_CH_NR_0        0U


XMC_DAC_CH_CONFIG_t ch_config0=
{
   .output_offset   =     0U,
   .data_type       = XMC_DAC_CH_DATA_TYPE_SIGNED,
   .output_scale    = XMC_DAC_CH_OUTPUT_SCALE_MUL_64,
   .output_negation = XMC_DAC_CH_OUTPUT_NEGATION_DISABLED,
};
```

4.2 XMC Lib の初期化

DAC チャネルは一般に初期化されてから、特定のモードに設定されます。

```
XMC_DAC_CH_Init(XMC_DAC0, DAC_CH_NR_0, &ch_config0);
```

一般的な初期化の後、DAC をパターンジェネレータモード (PatternMode) に設定する必要があります。3 つの事前定義されたパターンの 1 つである正弦波パターンが使用されます。符号出力は無効になり、DAC クロックソースはインターンに設定され、パターン周波数は 500Hz に設定されます。

```
const uint8_t pattern[] = XMC_DAC_PATTERN_SINE;


XMC_DAC_CH_StartPatternMode(XMC_DAC0, DAC_CH_NR_0,
                    pattern,
                    XMC_DAC_CH_PATTERN_SIGN_OUTPUT_DISABLED,
                    XMC_DAC_CH_TRIGGER_INTERNAL,
                    500);
```

追加の関数の実装は必要ありません。

5 ノイズジェネレータモード

ノイズジェネレータモードは、疑似乱数を生成するための 20 ビットの線形フィードバックシフトレジスタを提供します。12 ビットの乱数は、事前に選択されたトリガでサンプリングされます。実行時にスケーリング、オフセット、反転、および符号付き/符号なしの解釈を変更することにより、出力をさらにスクランブルできます。

負のスケーリング (2^X の除算) を使用することにより、振幅を制限できます。出力段が符号付きの値として解釈される場合、DAC 出力の半分のオフセットが生成されます。

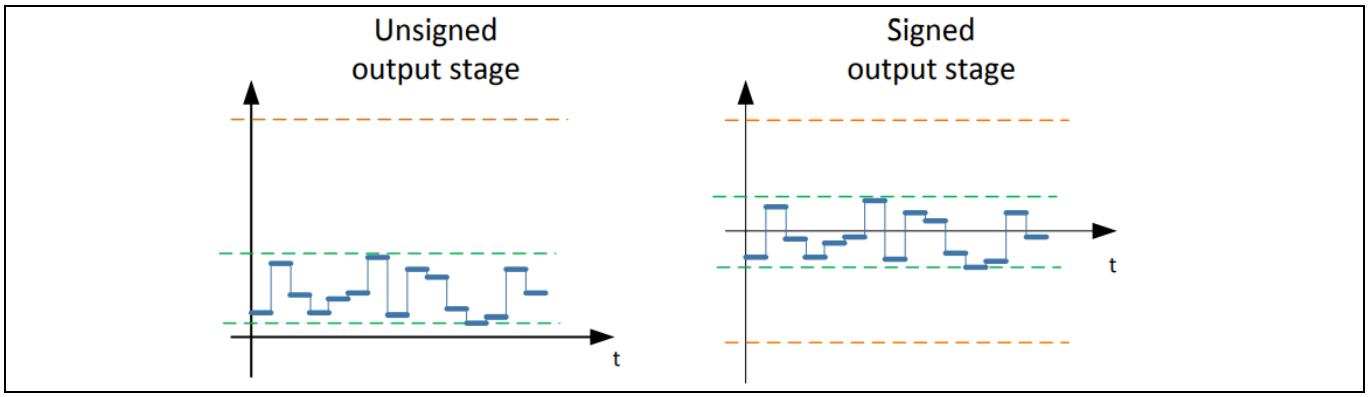

Figure 7 DAC ノイズサインと符号なし出力ステージ

5.1 XMC Lib 設定

この例では、DAC はノイズジェネレータモードで設定されます。ノイズジェネレータの出力は 12 ビットの符号なしデータです。したがって、オフセットはゼロに設定され、データ型は符号なしに設定され、スケーリングと反転は無効になります。

チャネル 0 は、対応する設定で初期化されます。

```
#define DAC_CH_NR_0       0U


XMC_DAC_CH_CONFIG_t ch_config0 =
{
   .output_offset   =     0U,
   .data_type       = XMC_DAC_CH_DATA_TYPE_UNSIGNED,
   .output_scale    = XMC_DAC_CH_OUTPUT_SCALE_NONE,
   .output_negation = XMC_DAC_CH_OUTPUT_NEGATION_DISABLED,
};
```

5.2 XMC Lib の初期化

DAC チャネルは一般に初期化されてから、特定のモードに設定されます。

```
XMC_DAC_CH_Init(XMC_DAC0, DAC_CH_NR_0, &ch_config0);
```

一般的な初期化の後、DAC をノイズジェネレータモード (NoiseMode) に設定する必要があります。DAC のクロックソースは内部に設定され、ノイズ周波数は 50kHz に設定されます。

```
XMC_DAC_CH_StartNoiseMode(XMC_DAC0, DAC_CH_NR_0,
                          XMC_DAC_CH_TRIGGER_INTERNAL,
                          50000U);
```

追加の関数の実装は必要ありません。

6 ランプジェネレータモード

ランプジェネレータモードは、12 ビットの分解能で電圧ランプを生成します。振幅は、開始値と停止値によって定義されます。スロープはクロック入力によって定義されます。

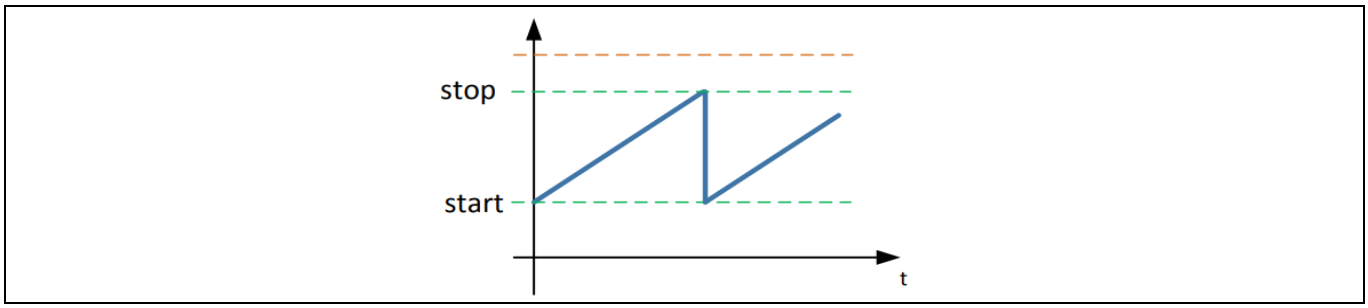

Figure 8 DAC ランプモード

反転機能により、負のランプも可能になります。反転ビットは、開始値と停止値の 2 の補数を 12 ビット形式で作成します。したがって、初期の開始値と停止値も 2 の補数形式であり、交換する必要があります。たとえば、元の開始値は 100U で、停止値は 3500U です。12 ビット形式の 100 の 2 の補数は 3996U で、3500U の補数は 569U です。次に、値を交換する必要があります。新しい開始値は 596U で、停止値は 3996U です。

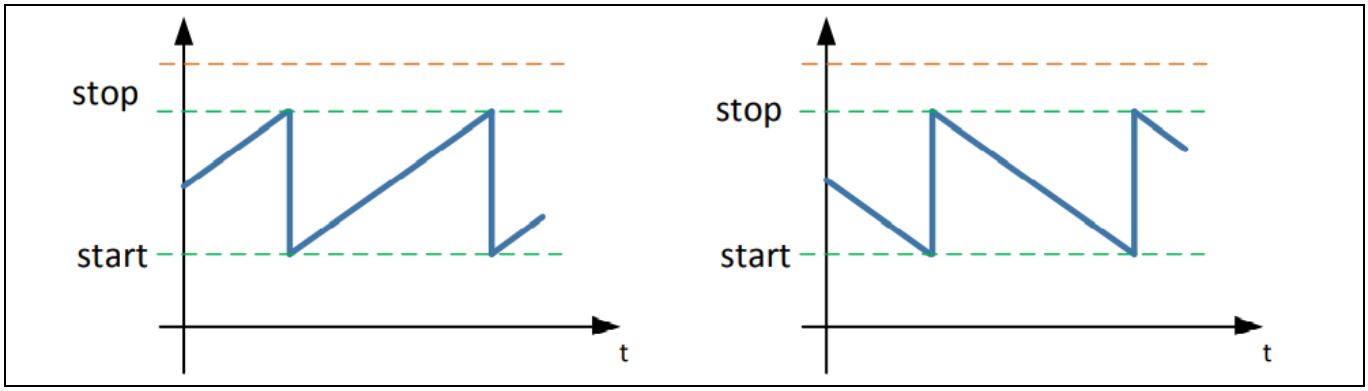

Figure 9 DAC ランプの正と負の勾配

スケーリング機能を使用すると、最小および最大ランプ周波数を拡大できます。

ランプの傾きは DAC クロックによって定義されます。クロックサイクルごとに、デジタル出力が 1 つずつ増加します。小さな振幅が使用される場合、これは最小ランプ周波数を制限する可能性があります。これは、開始値と停止値を 2 倍にし、負のスケール 2^{x-1} を追加することで補正できます。

これは逆の方法でも実行できます。フルスケールの振幅は、最大ランプ周波数を制限します。これは、開始値と停止値を半分にし、2 を乗算することで変更できます。これにより、ランプの分解能が低下します。

6.1 XMC Lib 設定

この例では、DAC チャネルをランプジェネレータモードで設定します。ランプジェネレータの出力は 12 ビットの符号なしの値です。したがって、DAC チャネルは符号なしおよびスケーリングなしとして設定されます。オフセットと反転は無効です。

```
#define DAC_CH_NR_0      0U


XMC_DAC_CH_CONFIG_t ch_config0=
{
   .output_offset   =    0U,
   .data_type       = XMC_DAC_CH_DATA_TYPE_UNSIGNED,
   .output_scale    = XMC_DAC_CH_OUTPUT_SCALE_NONE,
   .output_negation = XMC_DAC_CH_OUTPUT_NEGATION_DISABLED,
};
```

6.2 XMC Lib の初期化

DAC チャネルは一般に初期化されてから、特定のモードに設定されます。

```
XMC_DAC_CH_Init(XMC_DAC0, DAC_CH_NR_0, &ch_config0);
```

一般的な初期化の後、DAC をランプジェネレータモード (RampMode) に設定する必要があります。START_VALUE は 80 として定義され、STOP_VALUE は 3500 として定義されます。DAC クロックソースは内部に設定され、ランプ周波数は 500Hz に設定されます。

```
#define START_VALUE       80
#define STOP_VALUE        3500


XMC_DAC_CH_StartRampMode(XMC_DAC0, DAC_CH_NR_0,
                         START_VALUE,
                         STOP_VALUE,
                         XMC_DAC_CH_TRIGGER_INTERNAL,
                         500U);
```

追加の関数の実装は必要ありません。

改訂履歴

Document version	Date of release	Description of changes
**	2021-12-15	本版は英語版 AP32301 XMC4000 - Digital to Analog Converter(DAC) V1.0, 2015-07 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
AP32301

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。