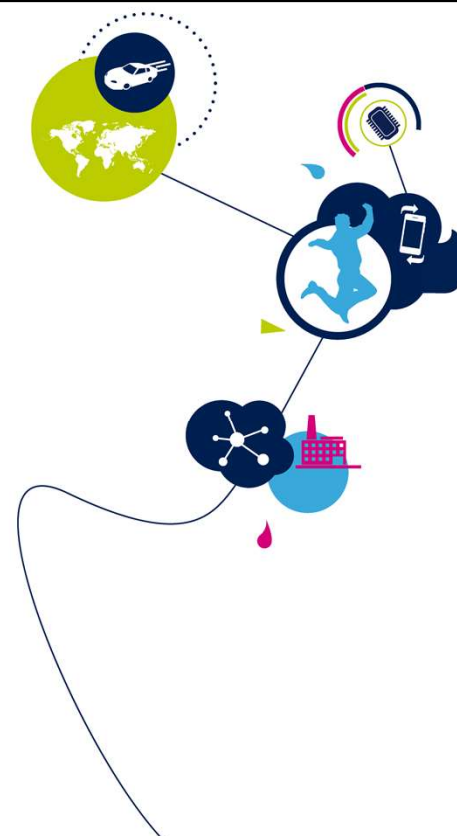


STM32G4 - FMAC

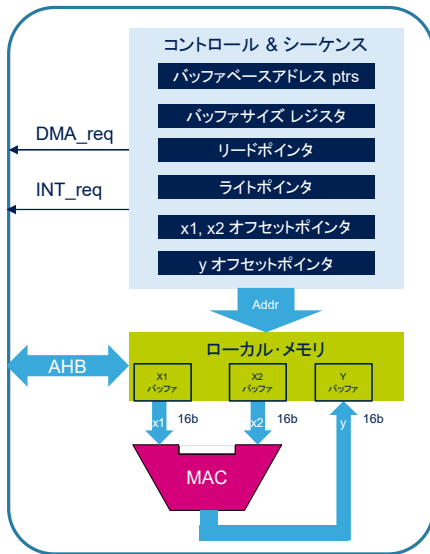
フィルタ数値演算アクセラレータ

1.0版



こんにちは、STM32G4 FMACブロックのプレゼンテーションへようこそ。

自律的にバックグラウンドで信号処理フィルタリングタスクを実行するために使用されるこのブロックの主な特徴について説明します。



- The Filter Math ACcelerator (FMAC)ユニット
 - 乗算/アキュムレータ(MAC)ユニットとサーキュラ・バッファを使ったデジタル・フィルタを搭載 (入出力)
- 乗算アキュムレータ(MAC)ユニット
 - 16 x 16bit乗算
 - 加算と減算が可能な24 + 2bitアキュムレータ

アプリケーション側の利点

- 固定小数点
- 有限および無限の衝動応答フィルタの両方を実装
- 256 x 16bitローカル・メモリ



FMACユニットは固定小数点乗算器とアキュムレータ(MAC)を中心に構築されています。

MACユニットは、内部の256x16ビットRAMから2つの固定小数点16ビットオペランドを受け取り、このメモリに結果を書き戻します。

ローカルメモリ内の入力値のアドレスは、ポインタをセットし使用します。

これらのポインタは、内部ハードウェアによってロード、インクリメント、デクリメント、またはリセットできます。

ソフトウェアはそれらに直接アクセスしません。

このユニットは、頻繁にある、または長時間のフィルタリング操作をCPUからオフロードできるようにし、プロセッサを他のタスクに解放します。

- 有限インパルス応答(FIR)と無限インパルス応答(IIR)フィルタの両方を実現可能
- ターゲット・アプリケーション
 - モータ制御、オーディオ、デジタル電源、照明、アナログ・センシング (ヘルス, フィットネス, ロボティクス,)
- CPUのオフロード
 - バックグラウンドで信号のフィルタリング・タスクを自律的に実行
 - CPUのMIPSを他のタスクへ開放するため、CPUの処理を最小限に
- 主な目的
 - フィルタの種類、順番、Coeffはプログラミング可能



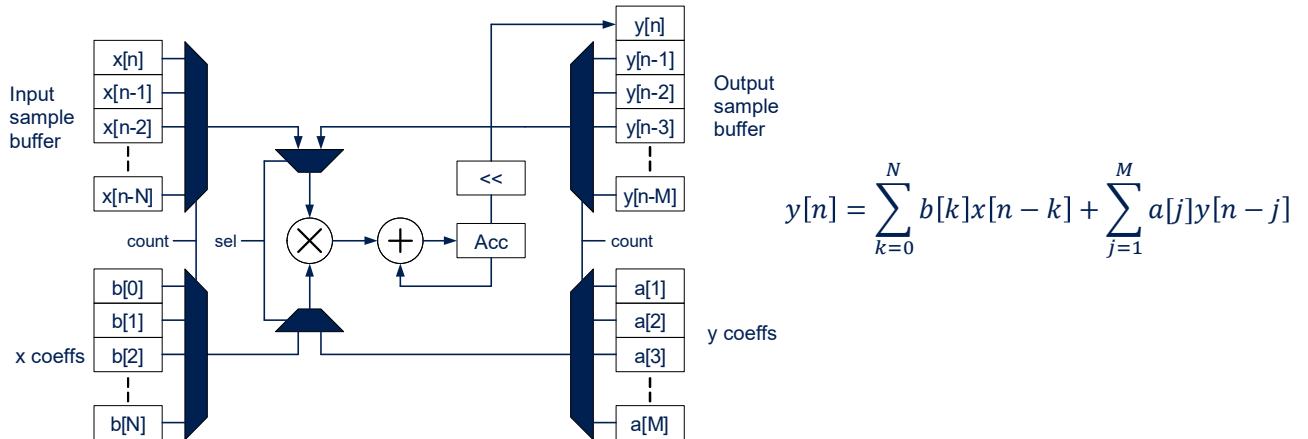
life.augmented

フィルタ機能FIRとIIRはFMACによって実現することができる。これらのフィルタを必要とする一般的なアプリケーションは、モータ制御、オーディオ、電源、照明、アナログセンシングです。FMAC は、バックグラウンドの信号のフィルタリング タスクを自律的に実行し、CPUからオフロードし、他のタスクへCPUのMIPSを解放します。FMACユニットを使用すると、ユーザーはフィルタタイプ、フィルタの順序、Coeff係数をプログラミングできます。

シングルMACアーキテクチャ

4

- 積の合計は、各出力サンプルに対する $N+M+1$ 乗算累積 (MAC) 演算となる
- コストを節約するために、1つの MAC が繰り返し使用



この図は、MACユニットのアーキテクチャの詳細を示しています。
Xは、フィルタリングするローサンプルを含む入力サンプルバッファです。

Bは、Xサンプルに適用されるフィルタのCoeffの配列です。

XとBのサイズは同じです。: $N + 1$ エントリ。

Yは、フィルタリングの結果を含む出力サンプルバッファです。

Aは、Yサンプルに適用されるフィルタの係数の配列です。

YとAのサイズは同じです。: $M + 1$ エントリ。

$y[n] = (N + 1) + M$ を取得するMAC操作の数:

- 累積ベクトルXとベクトルBを乗算する $N + 1$ MAC。
- 累積ベクトルY $[n-1:n-M]$ とベクトルAを乗算するM個のMAC

- FMACの入力と出力は、固定小数点符号付き整数q1.15フォーマットを使用。
 - q1.15フォーマットでは、数値は1つの符号ビットと15個の小数ビット (2進数) で表す
 - したがって、数値範囲は-1 (0x8000) から1 -2¹⁵ (0x7FFF)

q1.15フォーマット

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
S	c ₁₄	c ₁₃	c ₁₂	c ₁₁	c ₁₀	c ₉	c ₈	c ₇	c ₆	c ₅	c ₄	c ₃	c ₂	c ₁	c ₀

$$\text{Fractional_number} = (-1)^s \sum_{k=0}^{14} \frac{1}{2^{15-k}} c_k^k$$

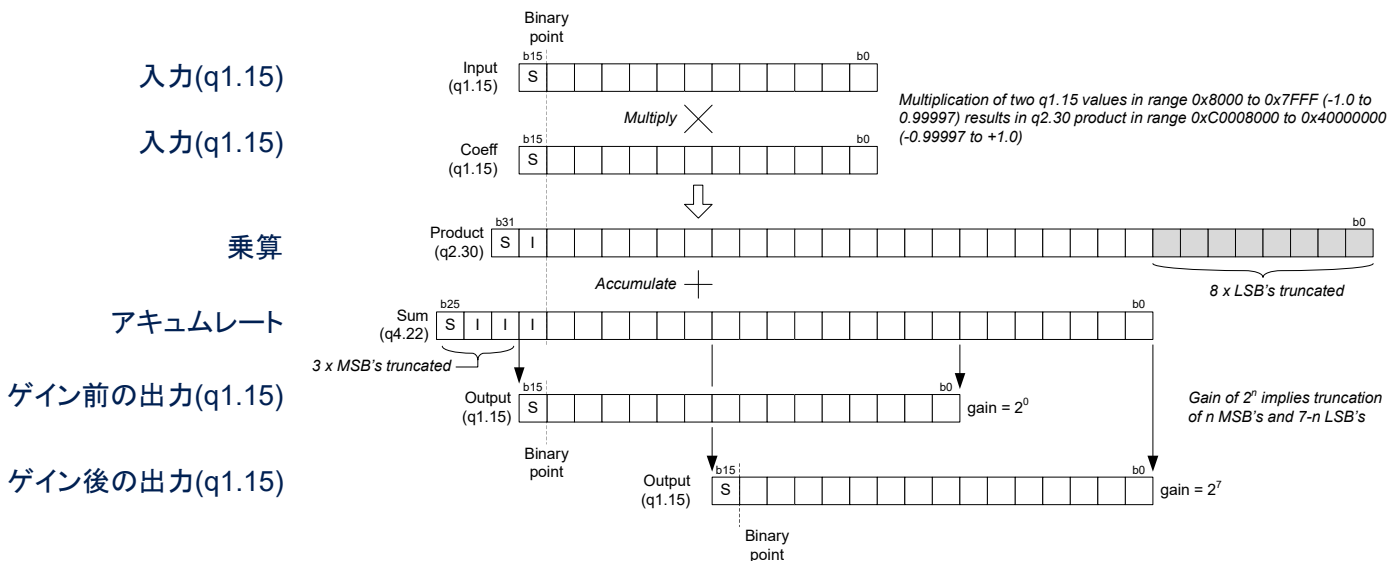
- 32bit浮動小数点数は、ソフトウェアの乗算とキャストによって固定小数点に変換可能 (Cortex[®]-M4 FPUを使用):
 - value_q15 = (int16_t)(value_f32*0x8000); /* f32 to q1.15 -8サイクル必要 */
 - value_f32 = (float)value_q15/(float)0x8000; /* q1.15 to f32 -10サイクル必要 */



FMACの入出力は、固定小数点符号付き整数q1.15形式を使用します。

q1.15形式では、数値範囲は1 (0x8000) ~ 1-2¹⁵ (0x7FFF) です。

32ビットの単精度浮動小数点数は、Cortex-M4 FPUで実行される専用の変換命令によってq1.15形式との間で変換できます



この図は、FMACによって内部的に使用されるさまざまな形式の詳細を示しています。

乗算器の出力 (q2.30フォーマット) はq2.22に切り捨てられ、アラインメントされたLSBに追加されます。

アキュムレータには26ビットがあり、そのうち22ビットは小数で、4ビットは整数/符号 (q4.22) です。

余分な整数ビットにより、アキュムレータは-8 (0x4000000) ~ +8 (0x3FFFFFF) の範囲の部分的な累積合計をサポートできます。

これは、連続する正または負の係数が多数ある場合に発生する可能性があります。

すべての周波数でフィルタゲインが1未満の場合、アキュムレータ値は常に範囲 +/- 1に戻ります。

部分的な合計がアキュムレータの数値範囲を超える場合 (ラップ)、スティッキフラグが設定されてデバッグを支援します。

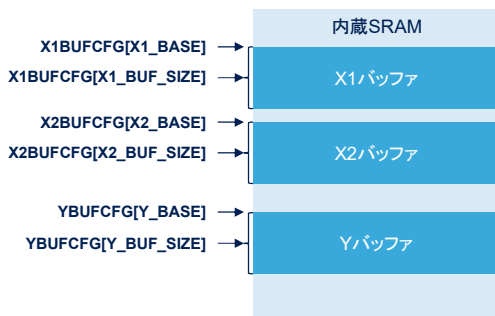
それでも、その後の追加でラッピングが取り消される場合でも、正しい結果が得られます。

アキュムレータの出力にプログラム可能なゲインを適用できます。

6dBのステップで0dBから42dBまで。

これは、IIRフィルタの実装に必要です。

- フィルタ数値演算アクセラレータ・ユニットは、ベクトルに対して算術演算を実行
- データ・バッファ (2つの入力、1つの出力) のメモリ内に最大3つの領域を定義
プログラム可能なベースアドレス・ポインタおよび関連するサイズ・レジスタによって定義される

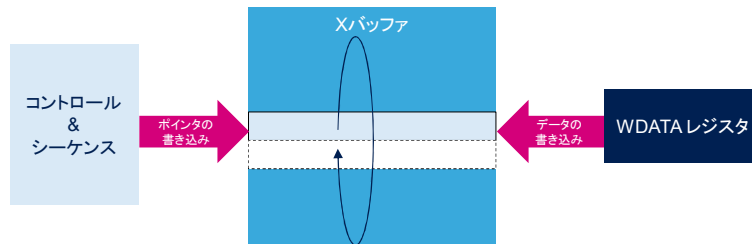


FMACユニットは、16ビットの固定小数点スカラー値の配列であるベクトルに対して算術演算を実行します。これらのベクトルはローカルSRAMに割り当てられます。ソフトウェアは、X1BUFCFG、X2BUFCFG、およびYBUFCFGレジスタを介してX1およびX2オペランドバッファとY出力バッファを構成します。すべてのバッファが内部メモリアドレスの範囲(0x00~0xFF)に収まる場合、ベースメモリは内部メモリのどこでも選択できます。バッファのベースアドレスとサイズをプログラムする必要があります。X1、X2、およびYバッファは重複する場合がありますことに注意してください。これらのバッファは、CPUマッピングでは表示されません。

入力バッファの初期化

8

- CPU (または DMA コントローラ) は、初期化関数を使用して各バッファの内容を初期化し、書き込みデータレジスタに書き込む
 - 操作の前にベクトルの要素をロード、フィルタとロード・フィルタのCoeffを初期化するために使用



フィルタ操作を開始する前に、CPUまたはDMAコントローラは、初期化関数を使用してWDATAレジスタに書き込むことにより、入力バッファの内容を初期化します。

入力バッファの内容は、フィルタリングされるデータまたはフィルタのcoeffのいずれかです。

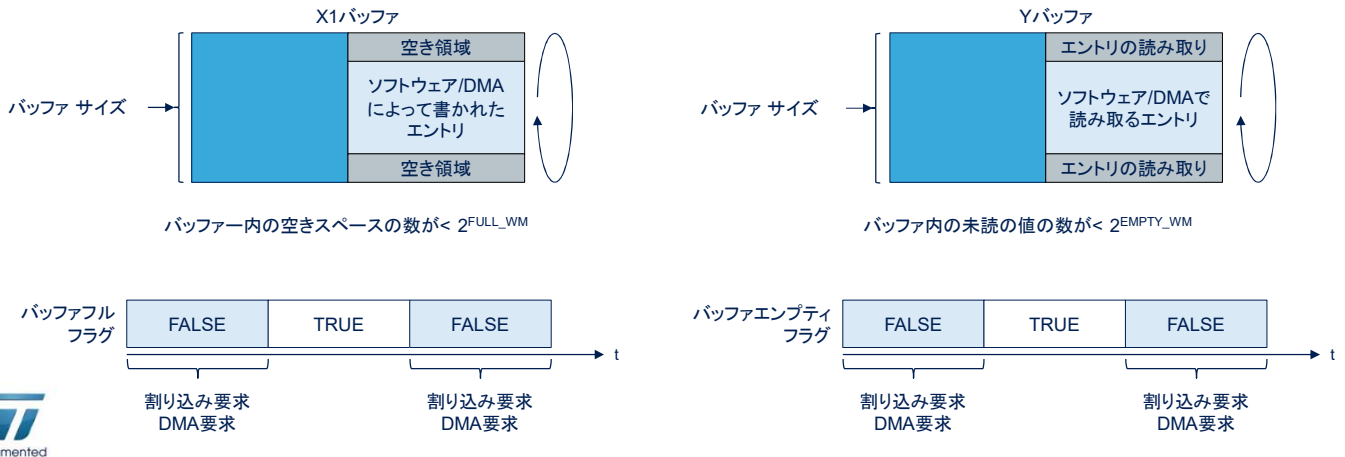
データは、書き込みポインタによって示されるターゲットバッファ内の場所に転送されます。

新しい書き込みのたびに、書き込みポインタがインクリメントされます。

書き込みポインタは、割り当てられたバッファスペースの最後に到達すると、ベースアドレスに戻ります。

	X1バッファ	X2バッファ	Y バッファ
サーキュラ・バッファ オペレーション	オプション	N/A	オプション

- CPUまたはDMAアクティビティを調整するため、ウォーターマーク・レベルの設定可能



X1バッファに関して、バッファ内の空きスペースの数がFMAC_X1BUFCFGレジスタのFULL_WMフィールドにプログラムされたウォーターマークのしきい値よりも少ない場合、バッファはフルとしてフラグが立てられます。

フルフラグが設定されていない限り、割り込みまたはDMA要求有効になっている場合、バッファの追加データを要求するために生成されます。

このウォーターマークにより、オーバーフローの危険なしに、1回の割り込みで複数のデータを転送できます。

それでも、オーバーフローが発生した場合、OVFLエラーフラグが設定され、書き込みデータは無視されます。

オーバーフローが発生しても、書き込みポインタは増加しません。

Yバッファについては、バッファ内の未読データの数がFMAC_YBUFCFGレジスタのEMPTY_WMフィールドにプログラムされたウォーターマークのしきい値よりも少ない場合、バッファは空としてフラグが付けられます。

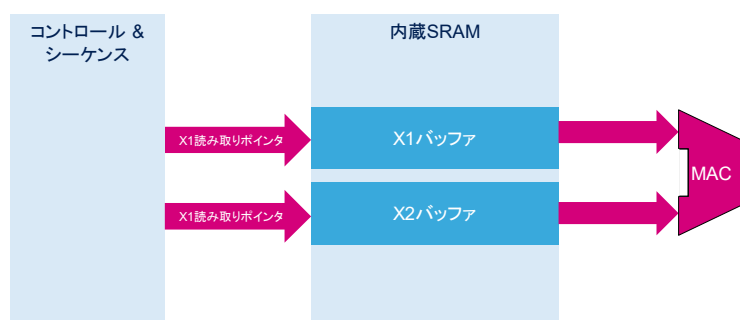
空のフラグが設定されていない限り、割り込みまたはDMA要求が生成され、有効になっている場合バッファからの読み取りを要求します。

ウォーターマークにより、アンダーフローの危険なしに、1つの割り込みで複数のデータを転送できます。

それでも、アンダーフローが発生すると、UNFLエラーフラグが設定されます。

この場合、読み取りポインタはインクリメントされず、読み取り操作は読み取りポインタアドレスのメモリの内容を返します。

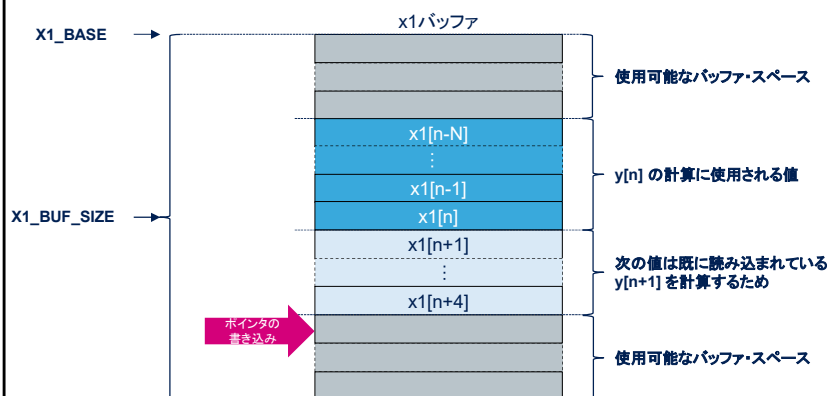
- X1およびX2バッファは、MACへの入力用のデータを格納するために使用される
 - コントロール・ユニット内のポインタは、各値の読み取りアドレス・オフセット (バッファ・ベース・アドレスに対する相対) を生成
 - ポインタは、現在の関数に従ってハードウェアによって管理される



各乗算は、X1バッファから値を受け取り、そしてX2バッファから値を受け取り、それらを乗算します。
 コントロールユニット内のポインタは、各値の読み取りアドレスオフセット(バッファ ベース アドレスに相対する)を生成します。
 ポインタは、現在の関数に従ってハードウェアによって管理されます。

入力データX1バッファ

11



- フィルタは、出力サンプル $y[n]$ を計算するために $N+1$ サンプル(入力セット)のセットを使用
- 計算が完了すると、次のサンプル ($x1[n+1]$) が入力セットに追加
最も新しいサンプル ($x1[n-N]$) が削除され、バッファ内の領域が解放される
- 入力データ・レジスタに到着した新しいサンプルは、書き込みポインタのバッファに書込まれる
 - これは常に、入力セット内の最新のサンプル $x[n]$ の後の最初の使用可能なスペースを示す



この図は、X1バッファ操作を説明しています。

書き込みポインタがバッファの最後に到達すると、先頭に戻ります。

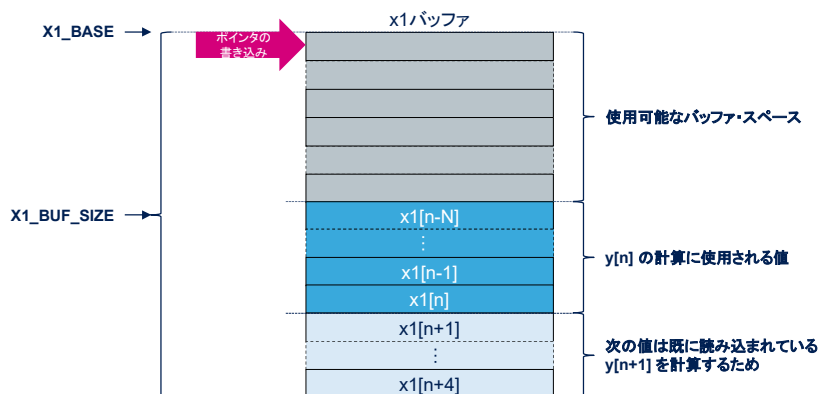
バッファ内の使用可能なスペースが転送サイズより小さい場合、入力バッファのフルフラグがアクティブになります。

入力セットの最上部である $x[n]$ が書き込みポインタに等しい場合 (つまり、利用可能な新しいサンプルがない場合)、新しいサンプルが利用可能になるまでフィルタは停止します。

プロセッサ、またはDMAコントローラは、必要に応じて新しいサンプル $x[n+1]$ がバッファスペースで利用できるようにする必要があります。

そうでない場合、バッファには空のフラグが付けられ、新しいサンプルが追加されるまでユニットの実行が停止します。

X1バッファではアンダーフロー状態は通知されません。



- X1バッファはサーキュラ・バッファとして使用可能
- このバッファの事前読み込みは、デジタル・フィルタではオプション
 - プリロードは、ベクトル演算の場合には役に立つ



X1バッファは、サーキュラ バッファとして使用できます。スペースが利用可能になると、新しいデータが入力バッファに継続的に転送されます。書き込みポインタは、図に示すように、バッファ内の最後の16ビットエントリに到達すると自動的に折り返されます。操作の開始時に入力サンプルがバッファに書き込まれていない場合は空としてフラグが付けられ、操作を開始するのに十分になるまでCPUまたはDMAをトリガして新しいサンプルをロードするため、デジタルフィルタの場合、このバッファのプリロードはオプションです。それにもかかわらず、プリロードはベクトル演算の場合に役立ちます。つまり、入力データはすでにシステムメモリで利用可能であり、サーキュラ演算は必要ありません。

- X2 バッファはベクトル・モード (サーキュラ・モードではない) でのみ使用でき、バッファの内容が1つの操作から次の操作に変更されない場合を除き、プリロードする必要がある
 - フィルタ関数の場合、X2バッファを使用してフィルタのCoeffを格納

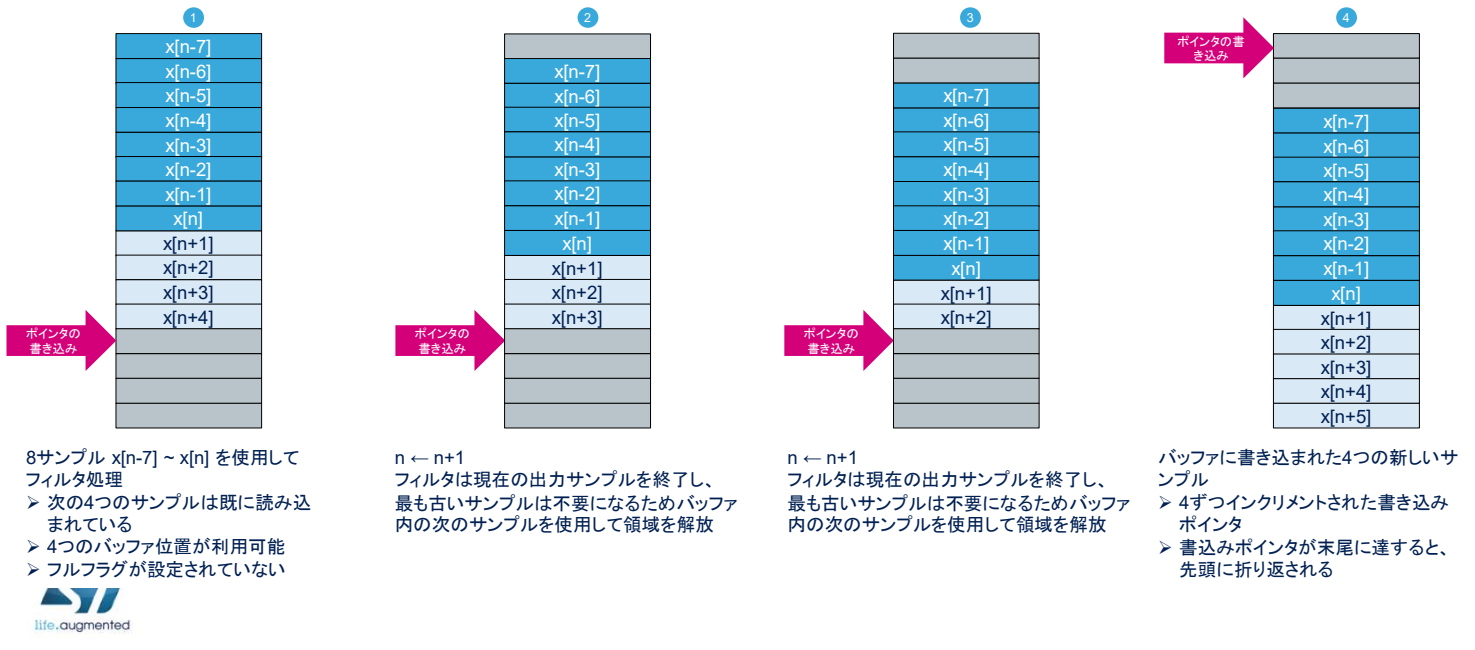


life.augmented

X2バッファは、Coeffを格納するために使用されます。
これは通常、FMACの初期化時に1回ロードされます。
したがって、それはサーキュラアドレッシングモードをサポートしていません。

入力バッファX1オペレーション

• N=8および4サンプルDMA転送を使用したフィルタ図



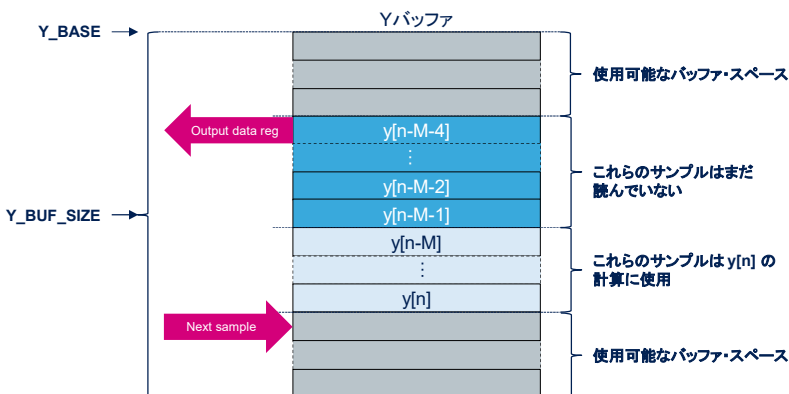
この図は、入力バッファの動作をまとめたものです。

ステップ 1 の間、フィルタは $x[n-7]$ から $x[n]$ まで $y[n]$ を計算し、次の 4 つのサンプルを読み込みます。

ステップ 2 の間に $y[n]$ が計算されるようになると、サンプル $x[n-7]$ が削除されます。次に、 n がインクリメントされます。フィルタは、 $y[n]$ を $x[n-7]$ から $x[n]$ に計算します。新しいサンプルは読み込まれません。

ステップ 3 の間に $y[n]$ が計算されるようになると、サンプル $x[n-7]$ が削除されます。次に、 n がインクリメントされます。フィルタは、 $y[n]$ を $x[n-7]$ から $x[n]$ に計算します。新しいサンプルは読み込まれません。

ステップ 4 の間に $y[n]$ が計算されるようになると、サンプル $x[n-7]$ が削除されます。次に、 n がインクリメントされます。フィルタは、 $y[n]$ を $x[n-7]$ から $x[n]$ に計算します。4 つの新しいサンプルが読み込まれます。バッファの上のアドレスに達したので、バッファの先頭に折り返しが発生します。



- Y (出力) バッファは、アキュムレータの出力を格納するために使用
 - 新しい各出力値は、プロセッサまたは DMA コントローラによって読み取られるまでバッファに格納
- フィルタは、出力サンプル $y[n]$ を計算するために M 出力サンプル (出力セット) のセットを使用
- 計算が完了すると、新しいサンプル ($y[n]$) がセットに追加され、最も新しいサンプル ($y[n-M]$) がセットから削除される
- 新しいサンプルは書き込みポインタのバッファに書き込まれる
 - これは常に、セット内の最新のサンプル $y[n-1]$ の後の最初の空き領域を示す
- Y バッファは、サーキュラ・バッファとしても動作

この図は、Y バッファの動作を説明します。

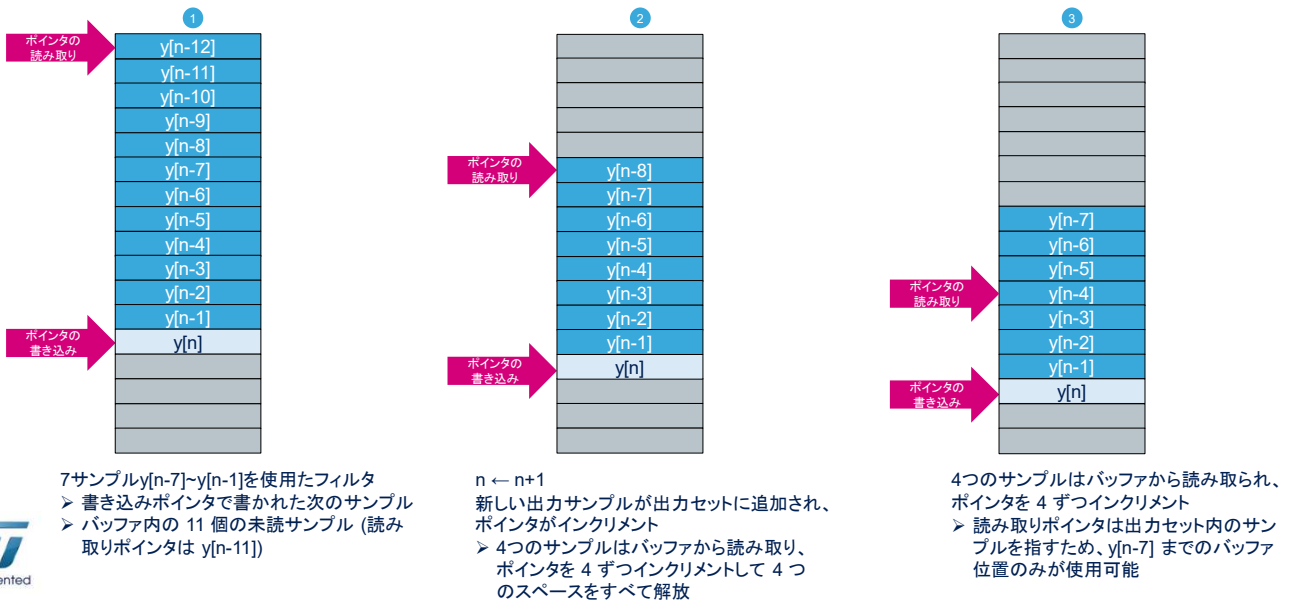
書き込みポインタがバッファの末尾に達すると、先頭に折り返されます。

リードポインタは、データが出力レジスタに対応し、最も古い未読み取りサンプルを指示します。

サンプルが読み取られ、出力セットの一部ではない場合、スペースは空になります。

書き込みポインタが読み取りポインタまたは出力セット内の最も最新でないサンプル ($y[n-M]$) と等しい場合、フィルタは停止し、出力バッファのフル フラグが設定されます。

- M=7および4サンプルDMA転送を使用したIIRフィルタ



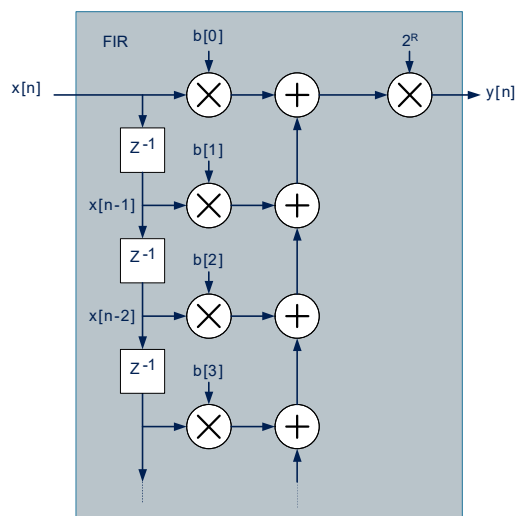
この図は、出力バッファの動作をまとめたものです。

ステップ 1 の間、フィルタは $y[n-7]$ から $y[n-1]$ まで $y[n]$ を計算します。11 個のサンプルが未読です。

ステップ 2 の間に n がインクリメントされると、フィルタは $y[n-7]$ から $y[n-1]$ まで $y[n]$ を計算します。ソフトウェアまたは DMA は 4 つのサンプルを読み取り、読み取りポインタは最も古いサンプルに移動します。

ステップ 3 の間に n がインクリメントされると、フィルタは $y[n-7]$ から $y[n-1]$ まで $y[n]$ を計算します。ソフトウェアまたは DMA は 4 つのサンプルを再度読み取り、読み取りポインタは最も古いサンプルに移動します。ただし、サンプル $y[n-7]$ ~ $y[n-5]$ は現在の計算で使用されるため、割り当て解除されません。

有限インパルス応答(FIR)フィルタ、ダイレクトフォーム1ストラクチャ



- 有限インパルス応答(FIR)デジタルフィルタは、入力サンプリング・データストリーム $x[n]$ と係数 $b[k]$ のセットとの間の畳み込み

$$y[n] = 2^R \sum_{k=0}^N b[k]x[n-k]$$

バッファ	利用	パラメータ	説明
X1	サンプリングされたデータ	P	ベクトル B の長さ
X2	フィルタ Coeff	Q	未使用
Y	出力値	R	出力に適用されるゲイン



FIR 関数は、フィルタ係数を含む長さ $N+1$ のベクトル B と、サンプリングされたデータを含む無期限の長さのベクトル X の畳み込みを実行します。

FMAC で FIR を実装するには、バッファを次のように使用します。

- X1 バッファにはベクトル X の要素が含まれます。長さ $N+1+d$ のサーキュラバッファである。
- X2 バッファにはベクトル B の要素が含まれます。長さ $N+1$ の固定バッファである。
- Y バッファには出力値 y_n が含まれます。長さ d のサーキュラバッファである。

ここでパラメータは次のとおりです。

- パラメータ P には、Coeffベクトル B の長さ $N+1$ が範囲 [2:127] で含まれます。
- パラメータ R には、アキュムレータ出力に適用されるゲインが含まれます。Y バッファに出力される値は 2^R で乗算され、R は範囲内にある [0:7]
- パラメータ Q は使用されません。

この機能は、FMAC_PARAMレジスタ内の START ビットがソフトウェアによってリセットされると完了します。

有限インパルス応答(FIR)フィルタ、ダイレクトフォーム1ストラクチャ

- メモリ・リクワイアメント
 - N Coeff と N 個の入力サンプルを計算
 - 各入力サンプルは N 回使用
 - スループットを最適化するには、入力バッファ・サイズを $N + \varepsilon$ にする必要がある
 - これにより計算に使用されていない間に1つの DMA 要求で ε サンプルを読み込むことが可能
 - 出力バッファ・サイズは、DMA 転送サイズ ε と一致するようにする必要がある
 - 合計メモリ・リクワイアメント: $2(N + \varepsilon) \times 256$



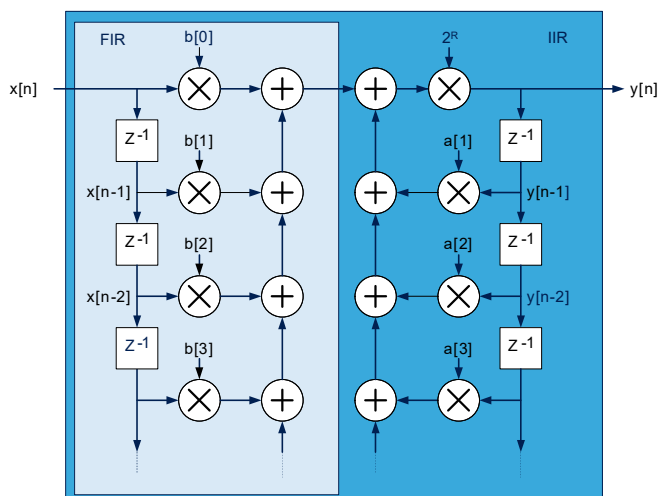
FIR では、1 つの出力サンプルを計算するために N Coeff と N 個の入力サンプルが必要です。

スループットを最適化するには、フィルタが現在のセットで作業している間、次のサンプルを読み込むために、入力バッファ サイズを N より大きくする必要があります。

たとえば、4 ビート DMA 転送を使用する場合、 ε は 4 に設定する必要があります。

また、出力バッファのサイズを ε に設定して、結果のサンプルを一意的 AHB バーストランザクションで転送する必要があります。

無限インパルス応答(IIR)フィルタ、ダイレクトフォーム1ストラクチャ



- 無限インパルス応答(IIR)フィルタは、FIR出力 $y[n]$ と2番目の係数セット $a[j]$ の間のFIRと2番目の畳み込みの組合せ

$$y[n] = 2^R \left(\sum_{k=0}^N b[k]x[n-k] + \sum_{j=1}^N a[j]y[n-j] \right)$$

バッファ	利用
X1	サンプリングされたデータ
X2	連結されたCoeffベクトルBとA ($b_0, b_1, \dots, b_N, a_1, \dots, a_M$)
Y	出力値

パラメータ	説明
P	ベクトルBの長さ
Q	ベクトルAの長さ
R	出力に適用されるゲイン



IIR フィルタ出力ベクトル Y は、長さ $N+1$ のCoeffベクトル B と無期限長のベクトル X の畳み込みに加えて、2 番目のCoeffベクトル A の遅延出力ベクトル Y' の畳み込みに加えて、長さ M の結合です。

FMAC で IIR を実装するには、バッファを次のように使用します。

- X1 バッファにはベクトル X の要素が含まれます。長さ $N+1+d$ のサーキュラバッファである。
- X2 バッファには、Coeffベクトル B と A を連結した要素が含まれます ($b_0, b_1, b_2, \dots, b_N, a_1, a_2, \dots, a_M$)。長さ $M+N+1$ の固定バッファである。
- Y バッファには出力値 y_n が含まれます。長さ $M+d$ のサーキュラバッファである。

パラメータは次のとおりです。

- パラメータ P には、Coeffベクトル B の長さ $N + 1$ の範囲 [2:64] が含まれます。
- パラメータ Q には、Coeffベクトル A の長さ M が範囲 [1:63] で含まれます。
- パラメータ R には、アキュムレータ出力に適用されるゲインが含まれます。Y バッファに出力される値は 2^R で乗算され、R は [0:7] の範囲になります。

この機能は、FMAC_PARAMレジスタ内の START ビットがソフトウェアによってリセットされると完了します。

無限インパルス応答(IIR)フィルタ、ダイレクトフォーム1ストラクチャ

- メモリ・リクアイアメント
 - NフィードフォワードCoeffとMフィードバックCoeff (M ~ N)
 - N個の入力サンプルとM出力サンプルで1つの出力サンプルを計算
 - DMA転送サイズ ε
 - 合計メモリ・リクアイアメント: $2(N + M + \varepsilon) < 256$



life.augmented

FIR には N フィードフォワードCoeffと M フィードバックCoeffが必要で、M は N より小さくなります。
入力バッファサイズは $N + \varepsilon$ 、 ε はDMAバースト内のデータ数である必要があります。
出力バッファ サイズは $M + \varepsilon$ である必要があります。

- Nタップフィルタには、出力サンプルごとにN乗算と加算が必要
 - 1サイクルあたり1回の乗算+加算を想定すると、最大スループット = サンプルあたりのNサイクル
 - 実際には、1つの乗算は、メモリからの2つのフェッチを必要とする(Coeff+ サンプル)
 - メモリは単一ポートであるため、MACオペレーションごとに2つのサイクルが必要
- したがって:
 - 最大サンプル・レート, $F_s < F_{clk} \div (2N)$
 - 最大フィルタ・サイズ, $N < F_{clk} \div (2F_s)$



FMAC モジュールのクロックリファレンスは、Cortex®-M4のFPUフリーランニングクロックFCLKです。

FIR などの N タップ フィルタでは、出力サンプルごとに N の乗算と加算が必要で、それぞれ MAC が 2 クロック サイクルを必要とし、2 つのメモリ値を読み取ります。

その結果、最大サンプルレートは $F_{clk} / (2 * N)$ になります。

N は F_{clk} 周波数よりも低く、最大サンプル レート周波数の 2 倍で割った値にする必要があります。

STM32G4の F_{clk} 周波数が170MHzであると仮定すると、以下が得られます。

- $F_s = 2\text{MSPS}$ での最大フィルタ サイズは $N < 42$ タップです
- $N = 128$ タップの最大サンプル レートは F_s の 664kHz です。

- ARM Fast mathライブラリと比較
 - STM32G474は150MHz動作
 - IAR EWARM 7.82最適化はハイスピード

フィルタ	CMSIS DSP (ソフトウェア)関数	CMSIS DSP 実行時間 (クロックサイクル / サンプル)	FMAC 実行時間 (クロックサイクル / サンプル)
単一ステージバイクワッド/2p2z (IIR with N=3, M=2)	arm_biquad_cascade_df1_fast_q15()	12	14
51タップFIR (N=51)	arm_fir_fast_q15()	92	104



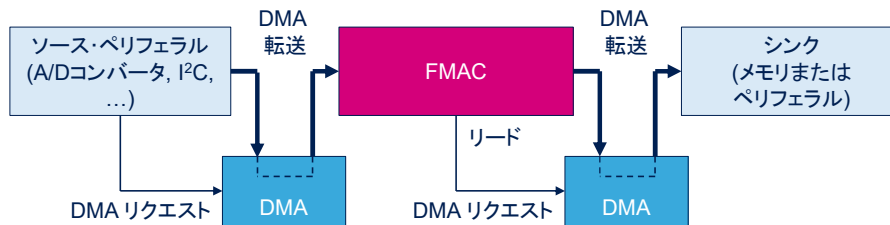
このスライドの表は、FMAC ハードウェア ブロックのパフォーマンスとフィルタ ソフトウェアの実装を比較しています。CMSIS DSP ライブラリは、最適化されたフィルタ関数が提供されます。

デュアルMAC Cortex-M4により、ソフトウェアは約15%高速です。しかし、高いサンプルレート/大きなフィルタの場合、CPUはフィルタリングタスクに多くの時間を費やしてしまいます。

FMAC + DMA を使用すると、CPU は他のタスクを自由に実行できます。

ソースドリブン・フロー・コントロール

- サンプルのソース(A/Dコンバータ、I²C、...)は、サンプル・データレートを定義



フローコントロールは、ソース、シンク、またはフィルタドリブンにすることができます。

このスライドでは、ソースドリブン フローコントロールシーケンスについて説明します。

サンプルのソース (A/Dコンバータ、I²C、...) は、サンプル データレートを定義します。

ソースは、フィルタ入力バッファにデータを転送する DMA (または CPU) にリクエストします。

フィルタは、ソースサンプルレートの2N倍よりも速いクロックレートで動作します。

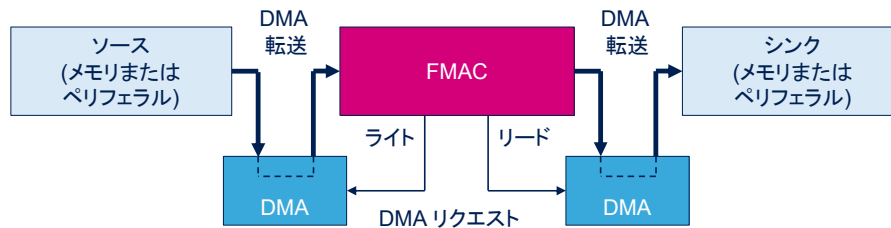
入力バッファが空の場合 (次のサンプルは使用できない)、フィルタは停止し、新しいデータが来るまで待機します。

出力バッファが空でない場合 (1 つ以上のサンプルが使用可能)、出力チャンネル DMA リクエスト (または割り込み) が生成されます。

DMA (または CPU) は、出力サンプルをメモリまたは D/Aコンバータ や PWM などの別のペリフェラルに転送します。

フィルタ・ドリブン・フロー・コントロール

- フィルタ・クロック・レートによってスループットが決定



このスライドでは、フィルタドリブン フローコントロールシーケンスについて説明します。

フィルタクロックレートによってスループットが決まります。

入力バッファがいっぱいでないと、入力チャンネル DMA リクエスト(または割り込み) が生成されます。

DMA (または CPU) は、メモリまたは別のペリフェラルから入力バッファにデータを転送します。

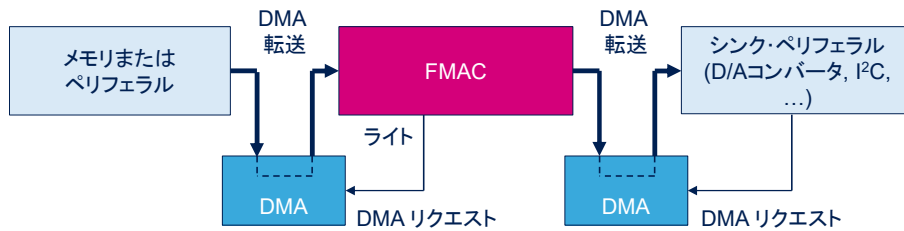
入力バッファでデータが使用可能である限り、フィルタは新しい出力サンプルを生成します。

出力バッファが空でない場合は、出力チャンネル DMA リクエスト(または割り込み) が生成されます。

DMA (または CPU) は、出力バッファからメモリまたは別のペリフェラルにデータを転送します。

シンク・ドリブン・フロー・コントロール

- サンプルの送信先(D/Aコンバータ、I²C、...)が、サンプル・データレートを定義



このスライドでは、シンクドリブン フローコントロールシーケンスについて説明します。

サンプルの送信先(D/Aコンバータ、I²C、...)は、サンプルデータレートを定義します。

送信先は、フィルタ出力からデータを転送するように DMA (または CPU) を要求します。

フィルタは、送信先のサンプルレートの2N倍よりも速いクロックレートで動作します。

出力バッファがいっぱいになると、フィルタは停止します。

入力バッファがいっぱいでない場合は、入力チャネル DMA リクエスト (または割り込み) が生成されます。

DMA (または CPU) は、メモリまたは別のペリフェラルから入力バッファにサンプルを転送します。

サマリー

- 入力バッファが空ではなく、出力バッファがいっぱいでない場合、フィルタが動作
 - 入力バッファのフル条件と出力バッファの空の条件は、それぞれの入力フラグと出力フラグを設定
 - 入力バッファのフルフラグが非アクティブの場合に入力チャンネルDMAリクエストを生成するようにフィルタを構成
 - 出力バッファの空フラグが非アクティブの場合に出力チャンネルDMAリクエストを生成するようにフィルタを構成出来る
- フィルタは、いずれかのフラグが非アクティブの場合に割り込みリクエストを生成することも可能
- フィルタクロック周波数は、選択したフローコントロールスキームに従って選択する必要がある



入力バッファが空ではなく、出力バッファがいっぱいでない場合、FMAC はフィルタ・アルゴリズムを実行します。

使用可能なスペースの数が X1 バッファの場合、X1FULL というフラグが FULL_WMLしきい値より少ない場合は、X1FULL というフラグが設定されます。

このフラグが X1 バッファを埋めるために設定されていない場合は、DMA リクエストを生成できます。

未読データの数がEMPTY_WMLしきい値より少ない場合は、YEMPTY というフラグが設定されます。

DMA リクエストは、このフラグが Y バッファを空にするために設定されていない場合に生成できます。

バッファの管理は、いずれかのフラグが非アクティブなときにアサートできる割り込みリクエストに依存するソフトウェアによっても実行できます。

フィルタクロック周波数は、選択したフローコントロールスキームに従って選択する必要があります。

モード	説明
RUN	有効
SLEEP	有効 <ul style="list-style-type: none"> ペリフェラルの割込みにより、デバイスがSLEEPモードを終了
低電力RUN	有効
低電力SLEEP	有効 <ul style="list-style-type: none"> ペリフェラルの割込みにより、デバイスが低電力SLEEPモードを終了
STOP 0/STOP 1	使用不可
STOP 2	
STANDBY	
SHUTDOWN	



FMAC ユニットは、RUN、低電力RUN、SLEEP、低電力SLEEP
 モードで有効です。
 他の低電力モードでは使用できません。

- 関連製品: STM32ファミリ