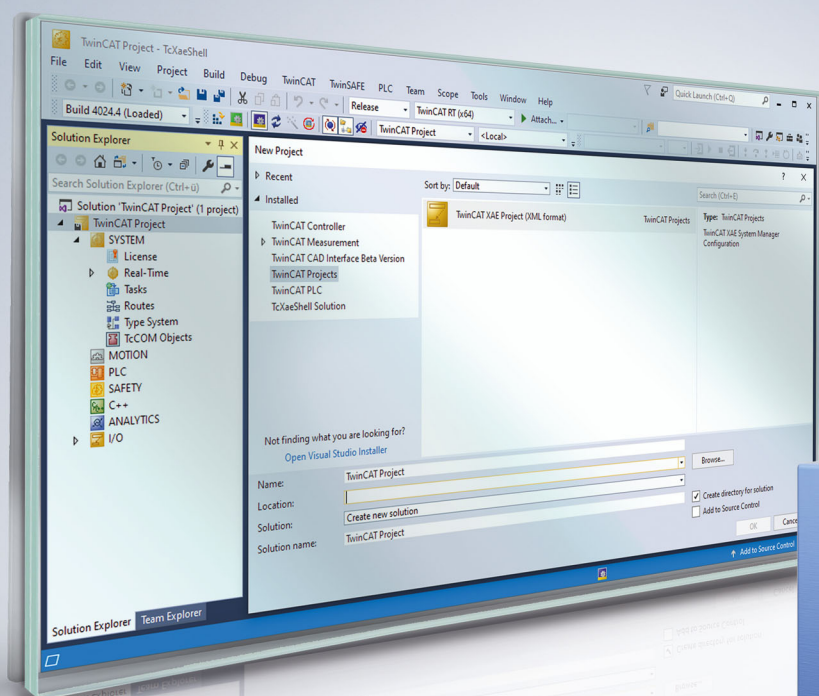


取扱説明書 | JA

TF3600 | Condition Monitoring

TwinCAT 3



目次

1 序文	7
1.1 取扱説明書に関する注記	7
1.2 安全にご使用いただくために	8
1.3 情報セキュリティに関する注記	8
2 概要	10
2.1 導入	10
2.1.1 フーリエ解析	11
2.1.2 データストリームの解析	16
2.1.3 一定期間のトリガー解析	23
2.1.4 スペクトルのスケーリング	25
2.1.5 統計解析	27
2.2 アプリケーションの概念	32
2.2.1 振動評価	33
2.2.2 周波数解析	34
2.2.3 ベアリングの監視	38
2.2.4 ギアボックスの監視	45
2.2.5 疲労寿命解析と損傷計算	49
2.3 文献注記	59
3 インストール	61
3.1 システム要件	61
3.2 互換性	61
3.3 インストール方法	62
3.4 インストール方法（TwinCAT 3.1. ビルド 4026）	65
3.5 ライセンス認証	65
4 技術概要	68
4.1 メモリ管理	68
4.2 タスク設定	69
4.3 NaN値	71
4.4 転送トレイを使った並列処理	72
4.5 マルチアレイの取り扱い方法	74
5 PLC API	79
5.1 ファンクションブロック	82
5.1.1 FB_CMA_AnalyticSignal	83
5.1.2 FB_CMA_ArgSort	87
5.1.3 FB_CMA_BufferConverting	90
5.1.4 FB_CMA_CrestFactor	93
5.1.5 FB_CMA_CrestFactorPlus	97
5.1.6 FB_CMA_ComplexDataHandling	101
5.1.7 FB_CMA_ComplexFFT	104
5.1.8 FB_CMA_Correlation	108
5.1.9 FB_CMA_DiscreteClassification	113
5.1.10 FB_CMA_Downsampling	117

5.1.11	FB_CMA_EmpiricalExcess	118
5.1.12	FB_CMA_EmpiricalMean	123
5.1.13	FB_CMA_EmpiricalSkew	128
5.1.14	FB_CMA_EmpiricalStandardDeviation.....	133
5.1.15	FB_CMA_Envelope	137
5.1.16	FB_CMA_EnvelopeSpectrum	141
5.1.17	FB_CMA_HistArray	145
5.1.18	FB_CMA_InstantaneousFrequency.....	150
5.1.19	FB_CMA_InstantaneousPhase.....	153
5.1.20	FB_CMA_IntegratedRMS	157
5.1.21	FB_CMA_MagnitudeSpectrum	161
5.1.22	FB_CMA_MeanStressCorrection.....	165
5.1.23	FB_CMA_MinersRule.....	168
5.1.24	FB_CMA_MomentCoefficients.....	172
5.1.25	FB_CMA_MultiBandRMS	177
5.1.26	FB_CMA_OrderPowerSpectrum	181
5.1.27	FB_CMA_PowerCepstrum	185
5.1.28	FB_CMA_PowerSpectrum	189
5.1.29	FB_CMA_Quantiles	193
5.1.30	FB_CMA_RainflowCounting.....	198
5.1.31	FB_CMA_ReadCsvFile	201
5.1.32	FB_CMA_RealFFT	203
5.1.33	FB_CMA_RMS.....	207
5.1.34	FB_CMA_SlidingDFT	210
5.1.35	FB_CMA_SparseSpectrum	214
5.1.36	FB_CMA_SpikeEnergySpectrum	219
5.1.37	FB_CMA_Sink	223
5.1.38	FB_CMA_Source	230
5.1.39	FB_CMA_SourcePaired.....	236
5.1.40	FB_CMA_VibrationAssessment	242
5.1.41	FB_CMA_WatchUpperThresholds.....	246
5.1.42	FB_CMA_ZoomFFT	250
5.1.43	FB_CMA_MovingMean.....	254
5.1.44	FB_CMA_MovingStandardDeviation	258
5.1.45	FB_CMA_MovingSkew.....	262
5.1.46	FB_CMA_MovingExcess.....	266
5.2	ファンクション	270
5.2.1	F_MA_IsNaN	270
5.2.2	F_CM_CalculateRecommendedOverlap.....	270
5.2.3	F_CM_CalculateWoehlerCurve	270
5.2.4	F_CM_CalculateWindow	271
5.2.5	F_CM_ApplySpectralScaling	271
5.3	データ型	272
5.3.1	E_CM_ComplexDataHandling.....	272
5.3.2	E_CM_CorrelationMode	272

5.3.3	E_CM_MCoefOrder	272
5.3.4	E_CM_MeanStressCorrection	273
5.3.5	E_CM_ScalingType	273
5.3.6	E_CM_SlidingDFTType	273
5.3.7	E_CM_SpectrumType	273
5.3.8	E_CM_UnwrapMethod.....	274
5.3.9	E_CM_WindowMode	274
5.3.10	E_CM_WindowType.....	274
5.3.11	E_MA_ElementTypeCode	274
5.3.12	T_CM_WindowParameters.....	275
5.3.13	エラーコード	275
5.3.14	InitPars構造体	280
5.4	グローバル定数	304
5.4.1	GVL_CM.....	304
5.4.2	GVL_CM_Base	304
5.4.3	Global_Version.....	305
5.4.4	Param	305
6	サンプル.....	307
6.1	実数値入力信号によるFFT.....	307
6.2	複素数値入力信号によるFFT	309
6.3	マグニチュードスペクトル:	311
6.4	マルチチャンネルのマグニチュードスペクトル	314
6.5	個々のスペクトル値の計算	315
6.6	ウィンドウ関数	318
6.7	スペクトルのスケーリング	320
6.8	時間ベースのRMS値	321
6.9	マルチバンドRMS	323
6.10	ヒストグラム	325
6.11	統計的手法	327
6.12	ISO 10816-3に準拠した振動評価.....	328
6.13	ISO 10816-3に準拠した振動評価（コンパクト版）	331
6.14	ISO 10816-3に準拠した振動評価（拡張版）	333
6.15	周波数解析による状態監視	335
6.16	平均化したマグニチュードスペクトルに対する閾値の考慮	340
6.17	クレストファクタ	341
6.18	包絡線スペクトル	344
6.19	パワーケプストラム.....	346
6.20	イベントベースの周波数解析	347
6.21	次数分析	349
6.22	疲労分析	351
6.23	スパイクエネルギースペクトル	354
6.24	ズームFFT.....	357
6.25	スライディングDFT.....	359
6.26	相関ファンクション.....	362

6.27 移動平均 364

7 付録..... 367

7.1 エラーコードの概要..... 367

7.2 ADSリターンコード 368

7.3 スペクトルスケーリングのオプション..... 371

7.4 サポートとサービス..... 373

用語集..... 375

1 序文

1.1 取扱説明書に関する注記

この説明書は対応する国内規格を熟知した、トレーニングを受けた制御、オートメーションエンジニアリングの有資格者のみの使用を対象としています。

本製品の設置およびコミッショニングの際は、必ず以下の注意事項と説明に従ってください。

有資格者は、常に最新版のドキュメントを参照する管理義務があります。

本製品を使用する上での責任者は、本製品の用途および使用方法が、関連するすべての法律、法規、ガイドラインおよび規格を含む、安全に関するすべての要件を満たしていることを確認してください。

免責事項

この取扱説明書の記載内容は、一般的な製品説明および性能を記載したものであり、場合により記載通りに動作しないことがあります。

製品の情報・仕様は予告なく変更されます。

この説明書に記載されているデータ、図および説明に基づいて、既に納品されている製品の変更を要求することはできません。掲載されている写真やイラストと、実際の製品は異なる場合があります。この説明書は最新でない可能性があります。必ず最新バージョンの説明書を参照してください。

商標

Beckhoff®, TwinCAT®, TwinCAT/BSD®, TC/BSD®, EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, Safety over EtherCAT®, TwinSAFE®, XFC®, XTS®, XPlanar® は、Beckhoff Automation GmbH の登録商標です。

この取扱説明書で使用されているその他の名称は商標である可能性があり、第三者が独自の目的のために使用すると所有者の権利を侵害する可能性があります。

特許

EtherCAT Technologyについては、欧州特許

EP1590927、EP1789857、EP1456722およびEP2137893、ドイツ特許DE102015105702に記載されていますが、これらに限定されるものではありません。

EtherCAT®

EtherCAT®は、Beckhoff Automation GmbHの登録商標および特許技術です。

著作権

© Beckhoff Automation GmbH & Co.KG, Germany.

明示的な許可なく、本書の複製、配布、使用、および他への内容の転載は禁止されています。

これに違反した者は損害賠償の責任を負います。ベッコフは、特許、実用新案、意匠の付与に関するすべての権利を留保しています。

1.2 安全にご使用いただくために

安全に関する注意事項

本書の各項目に記載されている製品の安全に関する指示をよくお読みになり、それに従ってください。

免責事項

すべての製品は、用途に適した特定のハードウェア構成およびソフトウェア構成を有する状態で供給されます。ハードウェアまたはソフトウェアに取扱説明書に記載されている以外の変更を加えることは許可されていません。許可されていない変更を加えると、Beckhoff Automation GmbH & Co. KGの保証の対象外となります。

使用者の資格

この説明は対応する国内規格を熟知した、トレーニングを受けた制御、自動化、およびドライブ技術の有資格者のみを対象としています。

安全記号の説明

この取扱説明書では、以下の安全記号を使用します。人的傷害および物的損害を避けるため、安全に関する注意事項はよくお読みになり、必ず指示に従ってください。人的傷害および物的損害を避けるため、安全に関する注意事項はよくお読みになり、必ず指示に従ってください。

人的傷害に関する警告:

⚠ 危険

この記号がついた注意事項に従わない場合、死亡または重傷を負います。

⚠ 警告

この記号がついた注意事項に従わない場合、死亡や重傷を負う可能性があります。

⚠ 注意

この記号がついた注意事項に従わない場合、軽傷を負う可能性があります。

物的損害に関する警告

注記

環境汚染、物的損害またはデータ消失の可能性があります。

製品の取り扱いに関する情報



ここには
製品に関する推奨措置、援助、または追加情報などが記載されます。

1.3 情報セキュリティに関する注記

Beckhoff Automation GmbH & Co. KG（ベッコフ）の製品は、オンラインアクセスが可能であれば、プラント、システム、機械、ネットワークの安全な運用をサポートするセキュリティ機能を備えています。セキュリティ機能にもかかわらず、プラント、システム、機械、ネットワークをサイバー脅威から守るためには、運用のための全体的なセキュリティ コンセプトの作成、実施、継続的な更新が必要です。ベッコフが販売する製品は、全体的なセキュリティ コンセプトの一部に過ぎません。お客様は、プラント、システム、機械、ネットワークへの第三者による不正アクセスを防止する責任を負います。ネットワークは、適切な保護措置が講じられている場合にのみ、社内ネットワークまたはインターネットに接続すべきです。

また、ベッコフが推奨する適切な保護対策も遵守してください。情報セキュリティと産業セキュリティに関する詳細は、<https://www.beckhoff.com/secguide> を参照してください。

ベッコフの製品とソリューションは常に進化し続けています。これはセキュリティ機能にも当てはまります。継続的な開発により、ベッコフでは、製品を常に最新の状態に保ち、アップデートが提供され次第、製品にインストールすることを明示的に推奨しています。古いバージョンやサポートが終了した製品の使用は、サイバー脅威のリスクを高めるおそれがあります。

ベッコフ製品の情報セキュリティ情報については、RSSフィードをご購読ください <https://www.beckhoff.com/secinfo>。

2 概要

ベッコフは、制御システムに状態監視システムを実装するためのハードウェアおよびソフトウェアコンポーネントによるツールボックスを提供しています。ベッコフのソリューションのメリットは、標準的な機械制御システムに統合されていることです。これにより、複雑な相互通信を伴う追加のサブシステムを省略できます。機械制御と状態監視は同じプラットフォーム上で動作し、同じエンジニアリングツールでプログラムすることができ、EtherCATによる共通のフィールドバスシステムを保有します。

TwinCAT Condition Monitoringのライブラリは、ソフトウェアツールボックスの重要な部分を形成しています。PLCファンクションブロックとして様々な数学的アルゴリズムが利用できます。

製品情報

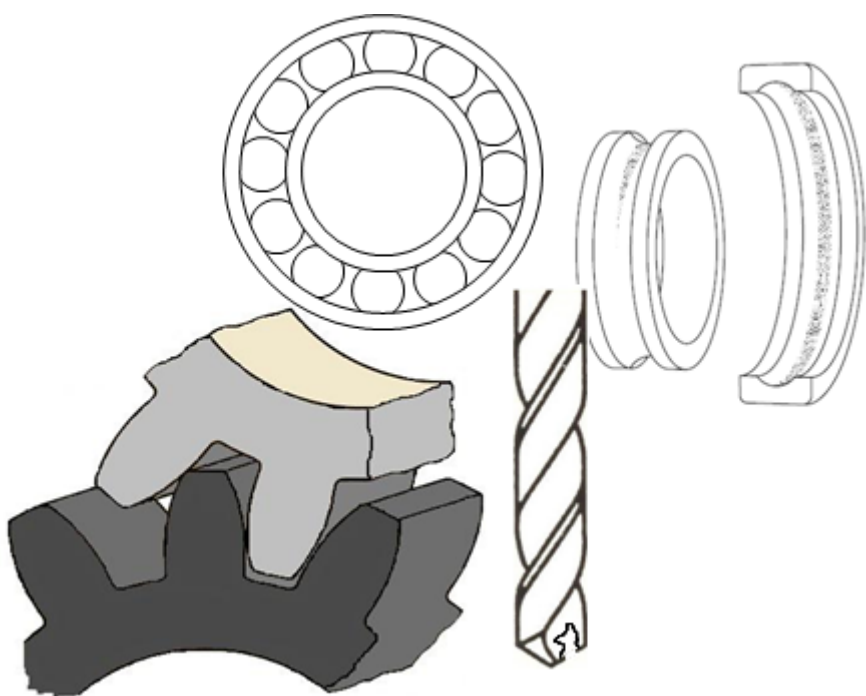
TwinCAT3 Condition Monitoringライブラリの最新バージョンは、<http://www.beckhoff.com/software>からダウンロードできます。PLCライブラリには、データ解析のための様々なアルゴリズムが用意されています。データの記録と処理を分離するために、マルチタスクのアプリケーションを推奨します。異なるタスクとCPUコア間の通信は、Condition Monitoringライブラリのメカニズムによって自動的に実施されます。

製品構成

TF360x Condition Monitoring製品は、以下のコンポーネントで構成されています。

- **PLC ライブラリ**： Tc3_CM.compiled-library, Tc3_CM_Base.compiled-library, および Tc3_MultiArray.compiled-library
- **ドライバ**： TcCM.tmx and TcMultiArray.tmx

TF3680 TC3 Filterのコンポーネントも含まれています。TF3600 Condition Monitoring レベル1ライセンスにはTF3680ライセンスが含まれているため、TF3680 TwinCAT 3 Filterの全てのコンポーネントを使用できます。



2.1 導入

状態監視や信号処理の経験がないユーザは、本書を補完するために追加の参考資料を参照することを強く推奨します。本章末の参考文献リストを参照してください。

基本的な信号処理の概念、特にフーリエ解析と統計について以下に紹介します。これにはプログラミングの詳細は含まれず、Condition Monitoringライブラリで使用するアルゴリズムのインターフェースと機能の説明に限定されます。

学習内容：

- 周波数解析とはどのようなものか
- 連続的なデータストリームのシームレスな解析はどのように機能するのか
- 時間区分を解析する方法および解析のトリガー方法
- スペクトルのスケーリング方法およびその重要性
- 信号がノイズや干渉の影響を受ける場合に信頼性の高い計測結果を得る方法

2.1.1 フーリエ解析

導入

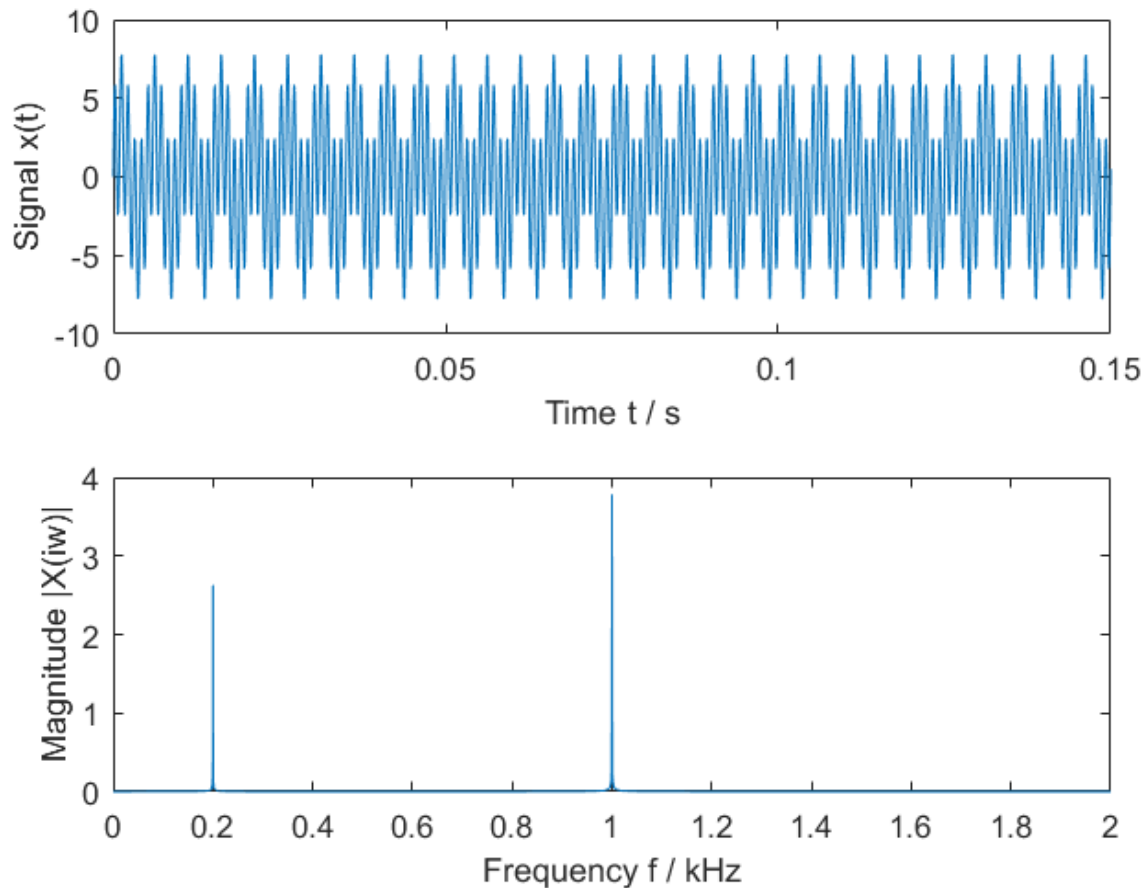
最も重要な周波数解析法はフーリエ解析です。フーリエ解析の基本的な意義は、信号 $x(t)$ を重畳された正弦波と余弦波の振動に分解するという事実から派生します。この変換の結果は、信号スペクトルまたは単にスペクトルと呼びます。フーリエ変換の定義：

$$X(w) = \int_{-\infty}^{\infty} x(t)e^{-i\omega t} dt = \int_{-\infty}^{\infty} x(t) [\cos(\omega t) - i \sin(\omega t)] dt$$

情報量という点では、信号スペクトルは元の信号と同等です。加えて、振動の起源に関する情報なども提供します。2つの機械部品が異なる周期（周波数）を持つ振動を発生させ、それが加法的に重畳している場合、フーリエ変換はこれらの2つの成分を可視化します。各周波数の正弦波と余弦波の組み合わせにより、位相角もマッピングできます。

例えば、周波数と振幅の異なる2つの正弦波を重ね合わせると、下図のようになります。経時的な変化から、結果として生じる信号の組成を読み取ることは困難であるか不可能です。逆に、振幅スペクトラム $|X(w)|$ 、すなわちフーリエ変換された絶対値から、適切なスケーリング（[スペクトルのスケーリング \[▶ 25\]](#)を参照）を行えば、周波数0.2kHz、振幅2.6の振動と、周波数1kHz、振幅3.8の振動があることが簡単に分かります。絶対値計算のため、位相情報は隠れています。

マグニチュードスペクトル：[\[▶ 311\]](#)のサンプルでは、この形式の信号についてマグニチュードスペクトルが計算され、表示されています。



音の伝達中に機械に発生する振動信号に影響を与える2つのプロセスがあります。1つ目は、周波数によって振動を減衰させる機械部品を介した伝達、2つ目は他の機械部品からの振動との重ね合わせで、相互作用なしに振幅が加算される場合です。どちらの要素もフーリエ変換の特性により分離されます：

- 遅延はフーリエ変換の位相にのみ影響する
- 振動振幅の周波数による選択的な減衰と構成による重ね合わせは、フーリエ変換のマグニチュードに現れる

時間離散信号の処理と離散フーリエ変換

フーリエ解析の応用において非常に重要な点は、信号の時間的サンプリングです。フーリエ変換は、連続的で時間的制限のない信号に対して数学的に定義されます。

しかし、実際には離散フーリエ変換（DFT）が使用されます。有限個の離散周波数成分を持つ離散周期信号に対して定義されます。「離散（Discrete）」とは、信号が等間隔でスキャンされることを意味し、通常はEL3xxxやELM3xxxなどのアナログ/デジタルコンバータで直接スキャンされます。

周期 T の時間連続信号がサンプリングされた場合、結果の値列は次のようになります。

$$x[n] = x(t = nT), \quad n = 0..N-1$$

DFTを使えば、 N 個の値からなるこの系列を離散スペクトルに変換できます。

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-\frac{2\pi i n k}{N}} = \sum_{n=0}^{N-1} x[n] \left[\cos\left(\frac{2\pi n k}{N}\right) - i \sin\left(\frac{2\pi n k}{N}\right) \right]$$

変数 k は、周波数チャンネルを表し、周波数ビンとも呼ばれます。変数 n と同様、0から $N-1$ まであります。

時間の離散化と値の量子化（デジタル化）

アナログ信号のデジタル処理には2つの操作が必要です：アナログ値からデジタル値への量子化と、量子化された値の離散シーケンスを形成するために、時間的に連続した物理信号をサンプリングすることです。

アナログ・デジタルコンバータは、I/Oターミナルの計測値をデジタル化します。値の量子化は一般に、符号付き16ビット表現または24ビット表現の整数信号の形をとります。TwinCAT 3 Condition Monitoring ライブラリの処理は、常に64ビットのIEEE倍精度浮動小数点形式で行われます。時間的サンプリングもI/Oターミナルで行われ、定義されたサンプリング周波数で入力信号をサンプリングします。サンプリング周波数は、タスクサイクル時間 T_c とオーバーサンプリング係数 c_s から計算できます：

$$f_s = c_s \cdot \frac{1}{T_c}$$

例：タスクサイクルタイムを1ms、オーバーサンプリング係数を10とすると、サンプリングレートは $f_s = 10 * 1 / (10^{-3} * 1s) = 10 \text{ kHz}$ となる

● サンプリング周波数に特に注意してください

i TwinCATでは、サンプリング周波数はタスクサイクルタイムと使用するターミナルのオーバーサンプリング係数に基づきます： $f_s := \text{Oversamples} * 1000.0 / \text{TaskCycleTime\_ms}$ 注意：通常のTwinCATと同様に、タスクサイクルタイムにはミリ秒の単位が使用されます。タスクサイクルタイムは、グローバルデータ型PlcTaskSystemInfoを使用してPLCで取得できます。参照：INFOSYSグローバルデータ型

サンプリングの定理

DFT適用における実用上の主な制限は、固有に表現できる周波数の制限です。ナイキストの定理またはサンプリングの定理によれば、最高周波数 $f_{\max}$ がサンプリング周波数 f_s の半分以下の信号だけが明確に表現できます（若干、簡略化した説明です）。したがって、サンプリング周波数は、アナログ信号で発生する最も高い周波数よりも大きくしなければなりません。

$$f_s > 2f_{\max}$$

アナログ信号に高い周波数が存在すると、スペクトルにエイリアシングと呼ばれる望ましくない影響が生じます。その結果、アナログ信号は離散信号の中で正しく表現されません。そのため、アナログ・デジタル変換の前に、アナログフィルタを設定することにより、アナログ信号から高い周波数を除去する必要があります。

● エイリアシング対策フィルタ

i EtherCATターミナルのEL3xxxおよびELM3xxxは、ターミナルの種類に応じて様々なフィルタを提供します。例えば、EtherCATターミナルのEL3632は、パラメータ設定可能なアナログ5次のローパスフィルタを備えており、エイリアシングを回避するために使用します。EL3751とELM3xxxモジュールには、エイリアシング対策フィルタおよび必要な信号のフィルタ用に設定できる複数のフィルタレベルがあります。TC3 Filter Designerも参照してください。

周波数分解能

周波数分解能（信号の周波数成分に基づく離散分解能）により、異なる信号成分を特定の機械要素や欠陥へ結びつけることが出来ます。そのため、多くの場合、離散周波数軸の分解能をできるだけ高くすることが有利に働きます。

一般的に、フーリエ変換の長さ N は、離散周波数軸 $k \Delta f$ のステップサイズ Δf を決定します。基本的な考察が理解を容易にします：ある正弦波の周波数を周波数領域で正しく表現するためには、その正弦波の1周期分以上の計測時間が必要です。この結果、分解能 Δf と計測時間 T_m の間には次のような関係があります：

$$\Delta f = \frac{1}{T_m} = \frac{f_s}{N}$$

典型的なPLCコードの構文（例：MAINルーチンのサンプルマグニチュードスペクトル：[▶ 311]）：

```
fSampleRate := cOversamples * (1000.0 / fTaskCycleTime);
```

$f_{\text{Resolution}} := f_{\text{SampleRate}} / c_{\text{FFTLengh}};$

したがって、高い周波数分解能を得るには、長い計測時間が必要となります。入力信号の前後にゼロを対称的に追加すること（ゼロパディング）で、DFTの入力データを拡張できます。これにより、一定のサンプリングレート f_s での信号シーケンスの長さ N が長くなり、離散分解能 Δf が向上します。ゼロパディングは、信号に付加的な情報を加えるものではありません。ゼロパディングを使用すると、2つの異なる種類の解像度が区別されます：一方は、離散周波数軸上のある周波数ビンから次の周波数ビンへのステップサイズ、すなわち $k\Delta f$ から $(k+1)\Delta f$ への遷移であり、もう一方は、入力信号の隣接する2つの周波数を区別するための分解能です。

ゼロパディングは離散分解能 Δf を低下させますが、計測分解能は変わりません。計測分解能の向上は、それに対応できるだけの長い計測時間（FFTベースのアルゴリズムのCondition Monitoringライブラリでは、よく $WindowLength$ と呼ばれる）によってのみ実現できます。実用的なアプリケーションの場合、重要な係数は通常、計測の周波数分解能であり、これは隣接する2つの信号周波数間の微分可能性に影響します。

● ゼロパディング

1 ゼロパディングは、解析する信号にいかなる情報も追加しません。したがって、隣接する2つの信号周波数を区別するために精緻化されるのは周波数分解能ではなく、周波数軸の数値分解能だけです。

実例に基づく図解：

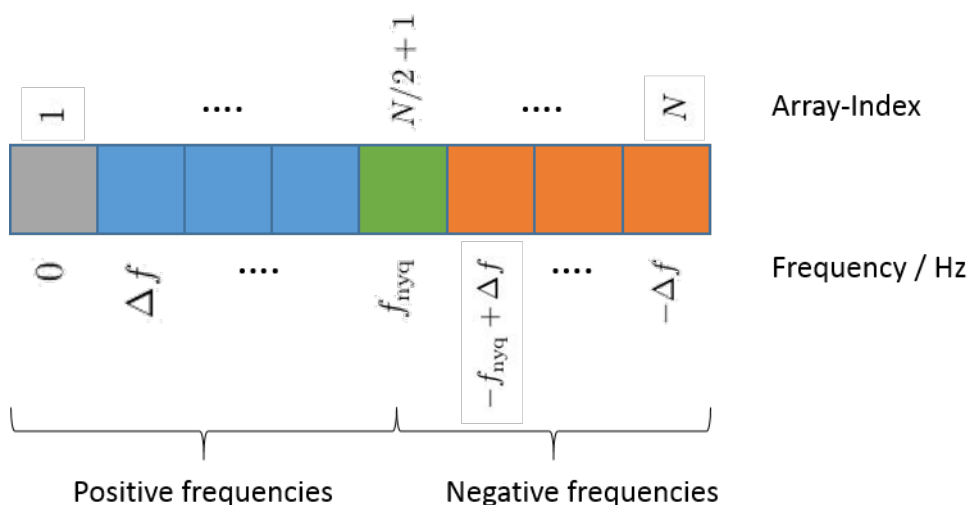
タスクサイクル時間 $T_c = 1\text{ms}$ 、オーバーサンプリング係数10（すなわち $f_s = 10\text{kHz}$ ）で、長さ3200のバッファが満たされています。その結果、計測時間は $T_m = T_c * 3200 / 10 = 320\text{ms}$ となり、計測分解能 $\Delta f = 1 / 320\text{ms} = 3.125\text{Hz}$ となります。さらなる解析/計算のためにFFTを使用し、バッファは $2 * 448$ のゼロで対称的に拡張され、長さ N は $2^{12} = 4096 > 3200$ に達します（ N は2の"累乗"である必要があります。次の項を参照）。ゼロパディングによって、数値分解能は $\Delta f = 10\text{kHz} / 4096 = 2,44140625\text{Hz}$ に精度が増します。

離散周波数軸は、ゼロ周波数（オフセット）と、サンプリング周波数の半分に相当するナイキスト周波数 f_{nyq} によって制限されます。ナイキストの定理によれば、これは記録された信号の表現可能な最高周波数に相当します。離散スペクトル $X[k]$ が、 $1 \dots N$ から始まるインデックス m のPLC配列に保存されている場合、 $X[k]$ の周波数軸は次のようになります。

$f_{\text{Frequency}} := (m-1) * f_{\text{Resolution}}; // m = 1..N/2+1$

$m = 1$ はオフセットを表し、 $m = N/2+1$ はナイキスト周波数を表します。 $N/2+2$ から N までの m に対する添字は、いわゆる負の周波数を形成します。これは、FFTの入力信号 $x[n]$ が複素数値を持つ場合にのみ実際に意味を持つ。[フーリエ解析 \[▶ 15\]](#) の項を参照してください。

次の図は、長さ N （ N は偶数）のDFT周波数軸の構成を示しています。



FFTアルゴリズムによる効率的な計算

厳密に言えば、高速フーリエ変換（FFT）は離散フーリエ変換（DFT）のアルゴリズムの一群です。FFTは、特に効率的に実装でき、同じ数値結果を導きます。単純に実装された N 個の時間値を持つDFTの複雑度が $O(N^2)$ であるのに対し、FFTの複雑度は $O(N \cdot \log N)$ で済みます。 N の値が大きいほど、その差は大きくなります。例えば $N=1024$ の場合、すでに約100倍になります。一般的にFFTアルゴリズムは、2の累乗を表す N （FFTの長さ）の値、すなわち256、512、1024などで定義されています。

複素数値の結果

FFT（およびDFT）は、入力信号 $x[n]$ を多数の正弦波と余弦波の振動に分割します。各周波数は、正弦成分と余弦成分の係数と関連付けられています。両方の係数は複素数として表されます。この分解はオイラーの公式で表されます：

$$e^{i\omega t} = \cos \omega t + i \sin \omega t$$

$$\operatorname{Re}\{e^{i\omega t}\} = \cos \omega t, \quad \operatorname{Im}\{e^{i\omega t}\} = \sin \omega t$$

各フーリエ係数の実数部 $\operatorname{Re}\{\dots\}$ は余弦成分に該当し、虚数部 $\operatorname{Im}\{\dots\}$ は正弦成分に該当します。この2つの成分の比率は、周波数成分の位相角を反映しています。

$$\Phi(k) = -\operatorname{atan} \frac{\operatorname{Im}\{X(k)\}}{\operatorname{Re}\{X(k)\}}$$

多くの場合、関心の対象は信号の正確な時間特性ではなく、マグニチュードスペクトルです。これは複素数の絶対値を計算することによってフーリエ変換から求めることができます：

$$|X(k)| = \sqrt{\operatorname{Re}\{X(k)\}^2 + \operatorname{Im}\{X(k)\}^2}$$

● 複素数データ型

i 実数値または複素数値の入力信号のFFT結果は、複素数値です。LREALおよびLCOMPLEXのデータ型が信号表現に使用されます。FB_CMA_MagnitudeSpectrum [▶ 161]またはFB_CMA_PowerSpectrum [▶ 189]の計算にファンクションブロックが使用された場合、結果は実数値になります。

画像周波数

実数信号のフーリエ変換では、負の周波数の係数は正の周波数の複素共役係数に等しくなります。 $X[k]$ を $x[n]$ のフーリエ変換、 $X^*[k]$ を複素共役とすると、 N 点を持つフーリエ変換は次のようになります：

$$X[k] = X^*[N - k]$$

実数値信号の場合、入力信号の時間反転はフーリエ変換の複素共役に相当します。ナイキスト周波数以下の周波数のスペクトル値は、ナイキスト周波数以上の値に反映されます。したがって、 $k > N/2$ の値は、実入力シーケンスでは冗長であるため、実シーケンスに対するフーリエ変換は、通常、最初の $n/2$ の値に限定されます（ $k = 0 \dots N-1$ に適用）。

スペクトル値計算の概要

Condition Monitoringライブラリには、スペクトル値を計算する様々な方法が用意されています。以下の表に概要を示します。FFTと（オーバーラップする）ウィンドウの関係については、次の章で詳しく説明します。

ファンクションブロック	入力			出力			注記
	次元	rea l	cplx	次元	Real	cplx	
FB_CMA_ComplexFFT [▶ 104]	N	-	✓	N	-	✓	- 複素数値の入力信号に対するFFT - ウィンドウ処理、オーバーラップ、スケーリングなし - 全スペクトルの計算

ファンクションブロック	入力			出力			注記
	次元	real	cplx	次元	Real	cplx	
FB_CMA_MagnitudeSpectrum [▶ 161]	L	✓	-	N/2+1	✓	-	- スペクトル値の絶対値 - ウィンドウ処理とスケーリングを調整可能 - 全スペクトルの計算
FB_CMA_OrderPowerSpectrum [▶ 181]	V,P	✓	-	N/2+1	✓	-	- オーダースペクトル - ウィンドウ処理とスケーリングを調整可能 - 全スペクトルの計算
FB_CMA_PowerSpectrum [▶ 189]	L	✓	-	N/2+1	✓	-	- パフォーマンス関連のスペクトル値 - ウィンドウ処理とスケーリングを調整可能 - 全スペクトルの計算
FB_CMA_PowerCepstrum [▶ 185]	L	✓	-	N/2+1	✓	-	- パワーケプストラム - ウィンドウ処理とスケーリングを調整可能 - ケプストラム全体の計算
FB_CMA_RealFFT [▶ 203]	N	✓	-	N/2+1, N	-	✓	- 実数値入力信号のFFT - ウィンドウ処理、オーバーラップ、スケーリングなし - 全スペクトルの計算
FB_CMA_SlidingDFT [▶ 210]	*	✓	-	nBins	-	✓	- オンラインDFT、つまりサンプルごとのアップデート - 最小の待ち時間 - 計算する周波数領域は設定可能
FB_CMA_SparseSpectrum [▶ 214]	M	✓	-	nBins	✓	✓	- 個別に設定可能なスペクトル線の計算 - ウィンドウ処理とスケーリングを調整可能
FB_CMA_SpikeEnergySpectrum [▶ 219]	L	✓	-	N/2+1	✓	-	- 周期的に発生する高周波パルスの検出 - ウィンドウ処理とスケーリングを調整可能
FB_CMA_ZoomFFT [▶ 250]	N	✓	✓	N/(2D)+1	-	✓	- スペクトルの周波数抜粋の計算 - ウィンドウ処理、オーバーラップ、スケーリングなし

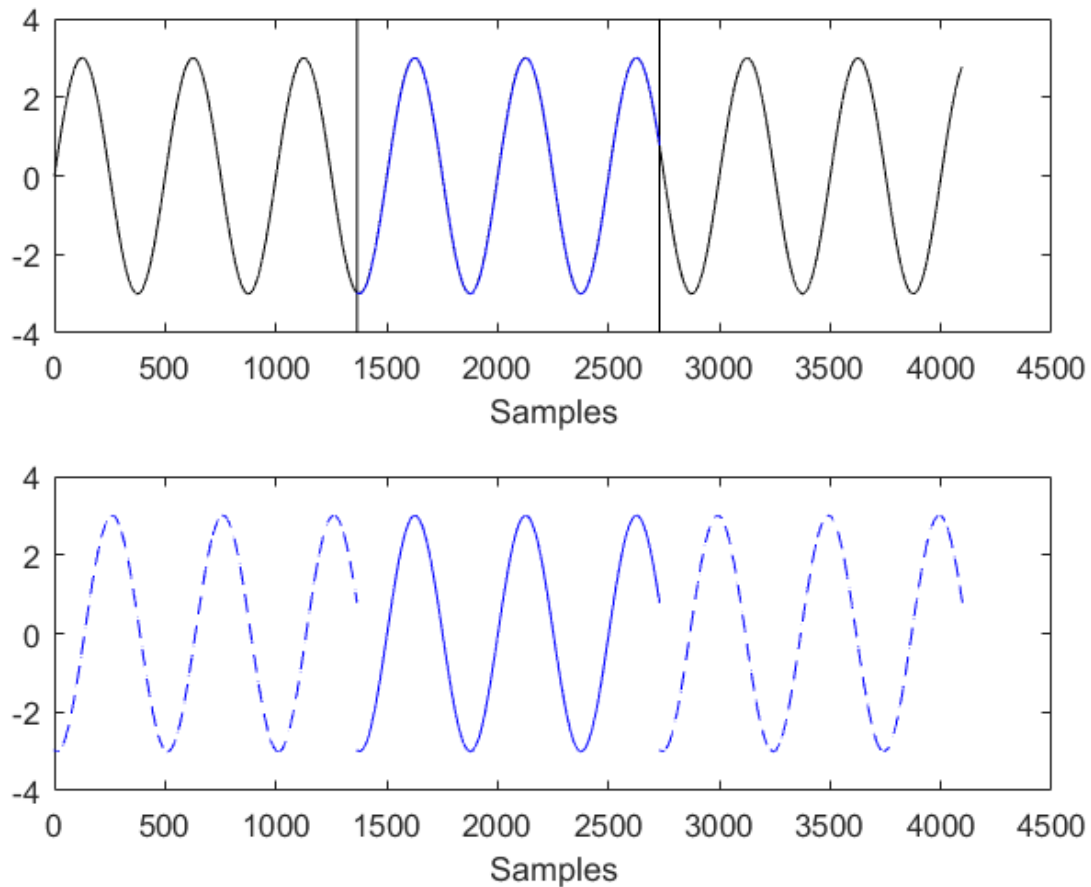
ここで、N はFFT長、L はウィンドウ長（またはそれぞれのオーバーラップ）、nBins は設定されたスペクトル値の数を示しています。さらに、D はデシメーション係数を示します。ファンクションブロック FB_CMA_SlidingDFT の入力データバッファの長さは固定のサイズに制限されません。オーダースペクトルの長さL,V は、特に振動と位置データに適合させる必要があります。

2.1.2 データストリームの解析

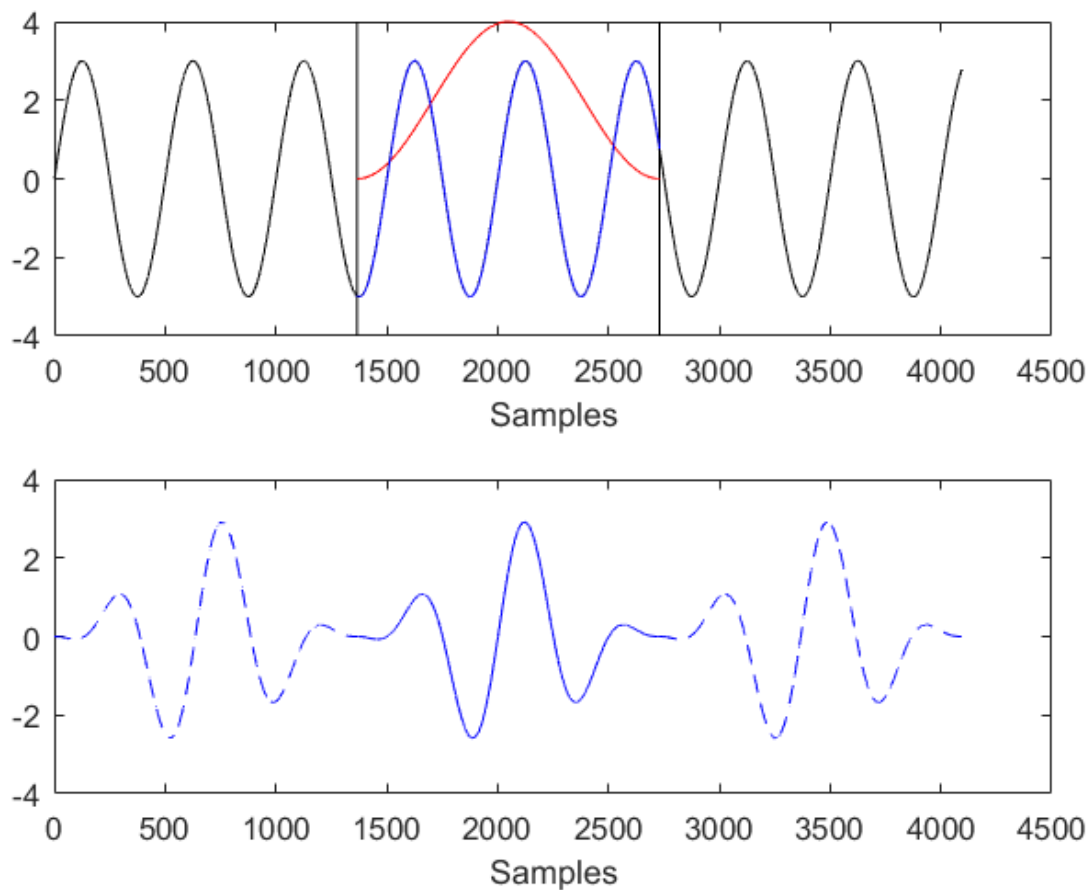
データストリームからブロックごとにFFT解析

DFT/FFTは、継続した周期的な信号に対して定義されます。これは、最初は意外な結論につながります：長い信号のFFT解析が必要な場合、入力信号を単純にセクションに分割してDFT変換できません。なぜなら、このようなセクションの最後の値が最初の値と一致しない場合、FFTはこれを周期シーケンスの不連続と解

釈し、スペクトルにはっきりと現れるからです(スペクトル漏れ)。次の図はその原理を説明しています。全信号(黒)から部分信号(青)を切り出します。FFTは、部分信号の周期的な継続を意味し(下図)、スペクトルに明確に示されているように、変換される信号の段階的な変化を仮定します。

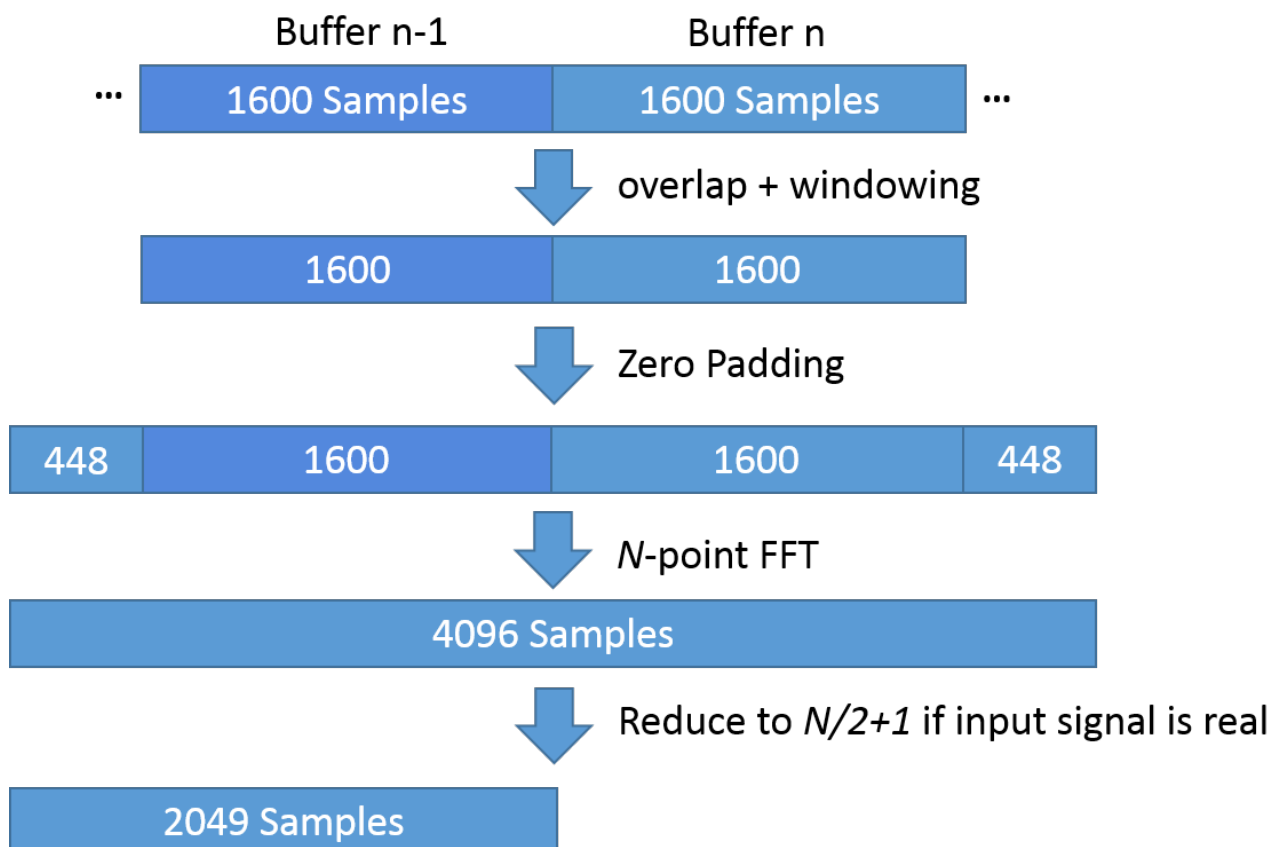


この状況は、適切なウィンドウ関数を用いて変換前の信号セクションに重み付けをすることで修正できます(詳細は次の項を参照)。適切なウィンドウでは、開始と終了の近傍の時間値は、係数ゼロまたはゼロに近い値が乗算されます。次の図は、同じシナリオをウィンドウ関数(赤)で示したものです。ウィンドウ処理は、周期的継続の段階的な変化を除去します。ただし、ウィンドウの特性はウィンドウ化された部分信号のスペクトルに現れることに注意してください。しかし、ウィンドウの特性がスペクトルに与える影響は一般的にはかなり小さいです。



ウィンドウ処理の問題点は、ウィンドウの端の値がスペクトルでほとんど考慮されないことです。この領域に損傷の可能性を示す信号特性がある場合、重要な情報が失われる可能性があります。情報の損失を防ぐため、TwinCAT Condition Monitoringライブラリは、ウィンドウ処理にオーバーラップした信号セクションを使用します。標準として使用されるHannウィンドウでは50%のオーバーラップが推奨されています。その結果、あるウィンドウセクションの端にあるサンプルは、次のウィンドウセクションの中央に位置します。

次の図は、オーバーラップ50%のデータストリームからFFT解析を行う手順を模式的に示しています。



最初に、1600の値で定義された長さのバッファがデータストリームで満たされます。前のバッファはバッファnからのデータの評価に含まれるため、ウィンドウ化されたデータパッケージには3200の値が含まれることになります。ウィンドウ関数の最大値は、2つのバッファのちょうど真ん中にあり、2つのバッファの端に向かうにしたがってゼロになります。ゼロパディングは、データパッケージを4096の値の長さに拡張するため、長さが2の累乗となり、FFTアルゴリズムで効率的に計算できます。FFTの結果は4096値のデータパッケージで、入力データが全て実数値であれば、必要に応じて2049値まで減らすことができます（[画像周波数 \[▶ 15\]](#)を参照）。

バッファnの評価中にバッファn+1が満たされ、バッファnはその評価に含まれます。このアプローチでは、ウィンドウで区切られた時間領域が常に50%オーバーラップする結果になります。

● TwinCAT 3でのデータストリーム解析

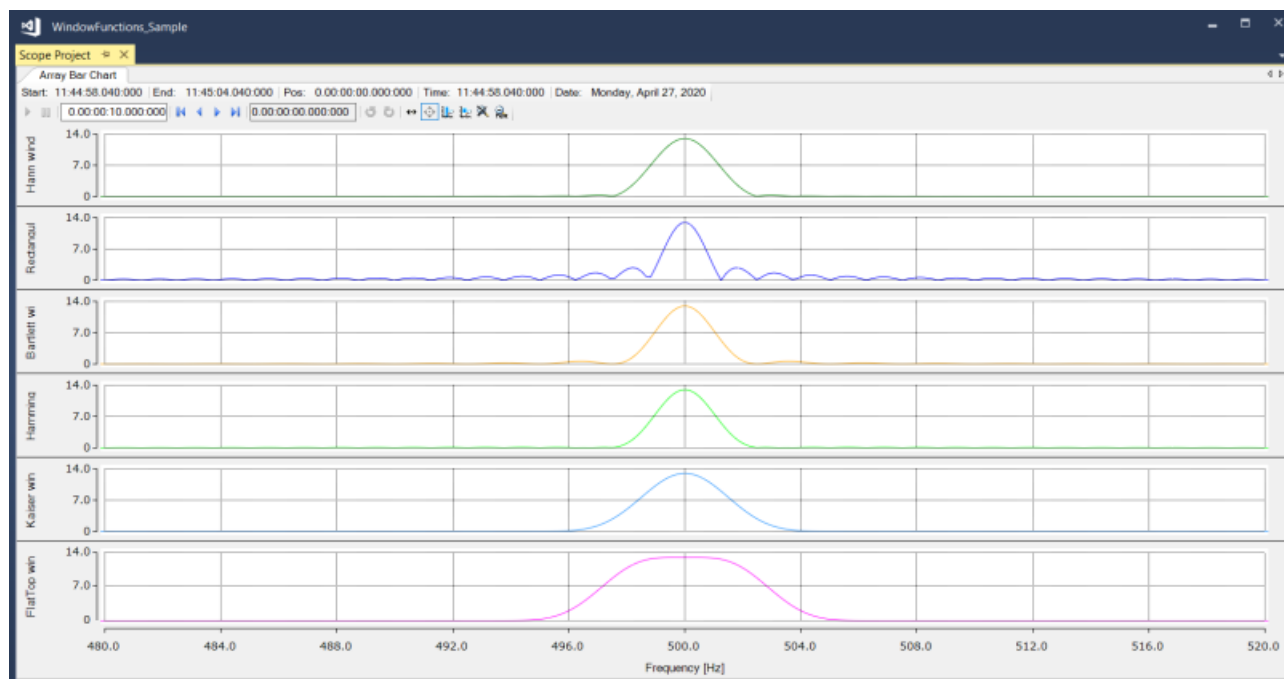
1 上図に示す信号解析スキームは、FB_CMA_MagnitudeSpectrumとFB_CMA_PowerSpectrumのCondition Monitoringライブラリに実装されています。ユーザは、パラメータ（バッファの長さ、FFTの長さ、ウィンドウ関数、オーバーラップなど）の設定と、データバッファの提供だけで十分です。例：[ST_CM_PowerSpectrum_InitPars \[▶ 295\]](#)を参照してください。

ウィンドウ関数

使用したウィンドウ関数の特性は、変換結果に示されます。フーリエ変換されるのは信号 $x[n]$ ではなく、 $w[n]$ をウィンドウ関数の時間値とした信号 $x[n] \cdot w[n]$ です。ウィンドウ関数の基本的な特徴に注意してください。

ウィンドウ関数が"使用されない"場合、すなわち信号セクションがより長い信号全体から取り出される場合、これは短径窓（rectangular window）の適用に該当します。例として、短径ウィンドウ（rectangular window）を用いたウィンドウ関数の特性を比較します：振幅13、周波数500Hzの高調波正弦波を10kHzのレートでサンプリングし、3200サンプル長のウィンドウ関数でウィンドウをかけ、その後にマグニチュードスペクトルを計算します（[スペクトルスケールリングのオプション \[▶ 371\]](#) eCM_PeakAmplitude）。以下の図では、ウィンドウ関数を使用した場合のマグニチュードスペクトルを比較しています。

以下のグラフィック再構築のためのサンプルはこちらを参照してください：[ウィンドウ関数 \[▶ 318\]](#)



これはウィンドウ関数の2つの重要な特徴を示しています：

- メイン・ローブの幅、この場合は約500Hz
- メインローブの最大値に対するサイドローブの減衰量
- ハーモニック信号の振幅精度は、振幅13の振動を指定した場合の最大値を比較

様々なウィンドウタイプはそれぞれ、これら3つの本質的な特性の間でトレードオフを形成しています。

メインローブの幅は、達成可能な周波数分解能に影響します。サイドローブの高さはスペクトル漏れを示しています。これは、解析する信号ではなく、ウィンドウによってのみ引き起こされるためです。最大値付近の-0.5ビンから+0.5ビンの範囲におけるメインローブの曲率は、高調波信号で発生しうる最大振幅誤差(スカロッピング損失)を特徴づけています。

短径ウィンドウ (rectangular window) は、非常に優れた周波数分解能を実現しますが、スペクトル漏れが強くなります。例えば、500Hzのピークに加えて550Hzに振幅0.5の周波数成分が発生した場合などに問題となります。また、最大振幅誤差は-36.34%と非常に大きくなります。**Hannウィンドウ**はサイドローブをかなり減少させますが、達成可能な周波数分解能も低下させます。正弦波信号の振幅を正確に決定する必要がない限りは、これは良い妥協点になります。ここで可能な最大振幅誤差は-15.12%です。Hannウィンドウは最もよく知られたウィンドウ関数の1つであるため、Condition Monitoringライブラリではデフォルトで設定されています。高調波信号で振幅精度が必要な場合は、中央のメインローブ領域で可能な限り平坦な曲率を持つ**フラットトップウィンドウ (SFT5M)**を使用します(最大振幅誤差-0.045%)。しかし、ここではメインローブが非常に広くなるため、このウィンドウは純粋に高調波信号の解析にのみ推奨されません。

ウィンドウ関数を使用する場合、周波数分解能の重要なパラメータは等価雑音帯域幅 (ENBW) です。

$$ENBW = N \frac{\sum w^2[n]}{(\sum w[n])^2} \Delta f = \frac{\sum w^2[n]}{(\sum w[n])^2} f_s$$

Δf の値は、FFT長 N とサンプリングレート f_s から導かれます(フーリエ解析 [▶ 11] 参照)。 Δf の前の式は、ウィンドウの特性により定義されます。例えば、短径ウィンドウ (rectangular window) の場合は1、Hannウィンドウの場合は1.5、SFT5Mの場合は3.885です。

使用するウィンドウ、パラメータ、オーバーラップの選択は、初期化パラメータを持つ各ファンクションブロック固有の構造体(例：ST_CM_MagnitudeSpectrum_InitPars [▶ 288])を介して、当該ファンクションブロックに対して実現されます。

利用可能なウィンドウ関数

長さNのウィンドウに対して、Condition Monitoringライブラリでは以下のウィンドウ関数が利用できます。ここで、 I_0 は第1種の変形ベッセル関数であり、次数はゼロです。eCM_KaiserWindow、eCM_FlatTopWindowは、initPars構造体のT_CM_WindowParameters [▶ 275]を介してパラメータ設定されます。最後の列は、推奨されるオーバーラップをウィンドウの長さのパーセントで指定します。例として、

$a_0 = \beta = 4\pi$ のカイザー4ウィンドウがeCM_KaiserWindowのために選択され

$a_0 = 0.209671, a_1 = -0.407331, a_2 = 0.281225, a_3 = -0.092669, a_4 = 0.0091036$

のSFT5MのウィンドウがeCM_FlatTopWindowのために選択されています。

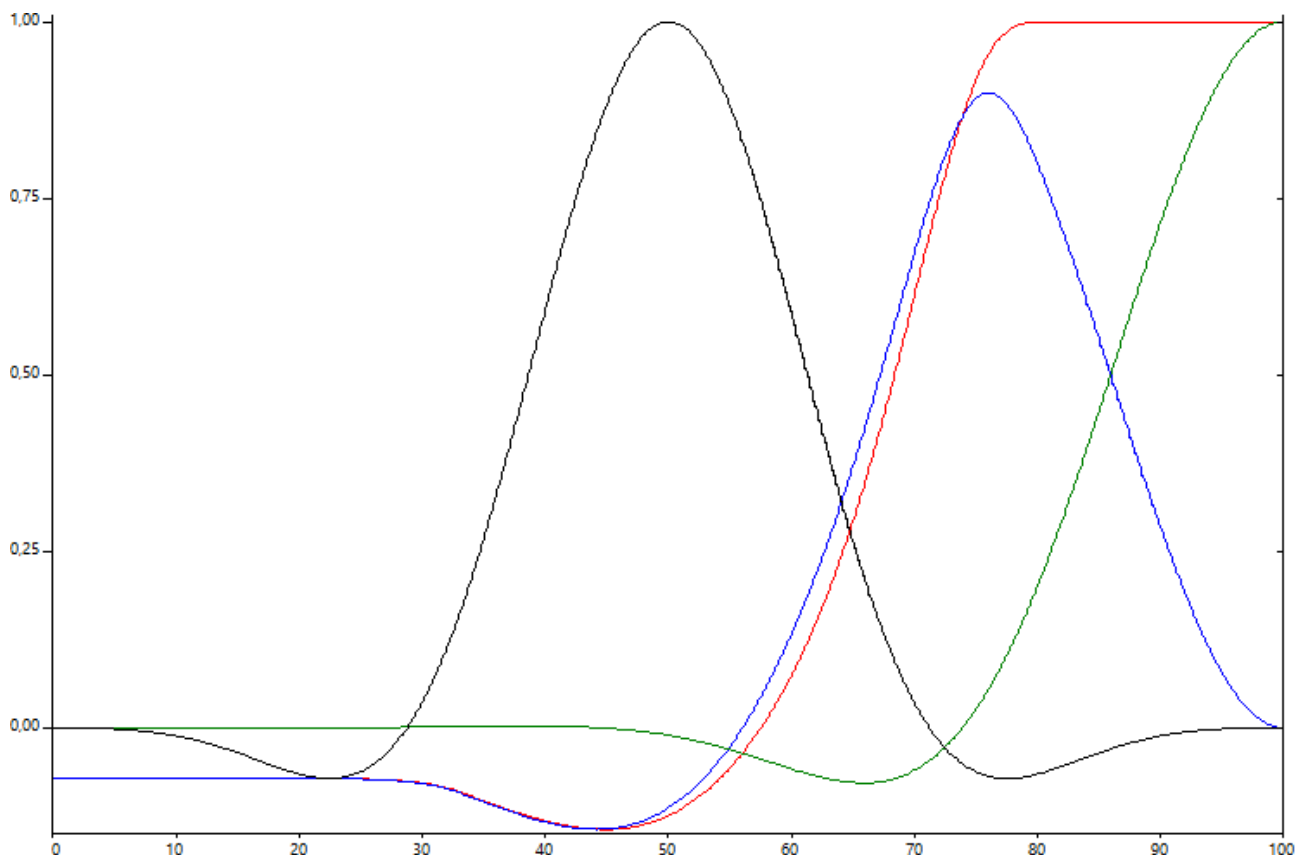
識別子	PID	定義	オーバーラッピング
eCM_HannWindow	16#05300901	$w[n] = 0.5 \cdot (1 - \cos(2\pi n/N))$	50%
eCM_RectangularWindow	16#05300902	$w[n] = 1$	0%
eCM_BartlettWindow	16#05300905	$w[n] = 1 - 2n/N - 1 $	50%
eCM_HammingWindow	16#05300906	$w[n] = 0.54 - 0.45 \cos(2\pi n/N)$	50%
eCM_KaiserWindow	16#05300907	$w[n] = I_0 \left(\beta \sqrt{1 - (2n/N - 1)^2} \right) / I_0(\beta)$	67% (カイザー4)
eCM_FlatTopWindow	16#05300917	$w[n] = a_0 + \sum_{k=1}^4 \{(-1)^k a_k \cos(2k\pi n/N)\}$	76% (SFT5M)

推奨オーバーラップの計算

推奨されるウィンドウ付き信号区間のオーバーラップは、ウィンドウ関数の以下の特性を考慮した結果です：

- 振幅の平坦度：オーバーラップウィンドウ関数の平坦度、すなわち個々のサンプルの重み付け
- オーバーラップの相関：重複する関数の個々のサンプル間の相関

推奨されるオーバーラップは、2つの特性の差が最も大きくなる点にあります。言い換えれば、可能な限り小さな相関関係で、最高の平坦性が達成されます。これを以下の図でSFT5Mのウィンドウ（eCM_FlatTopWindow）に示します。



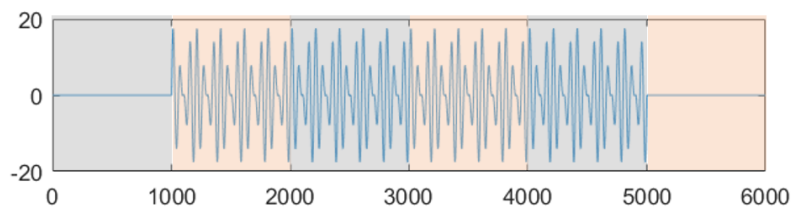
ウィンドウ関数（黒）、振幅平坦度（赤）、オーバーラップ相関（緑）、推奨オーバーラップ（青）。

詳細は以下を参照：

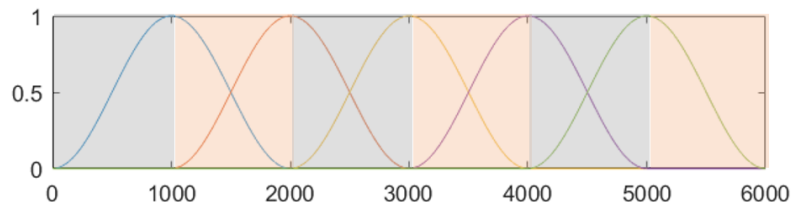
- G. Heinzel et.al：離散フーリエ変換（DFT）によるスペクトルおよびスペクトル密度の推定、ウィンドウ関数の包括的なリストと新しいフラットトップウィンドウを含む。（G. Heinzel et.al.: Spectrum and spectral density estimation by the Discrete Fourier transform (DFT), including a comprehensive list of window functions and some new flat-top windows. 2002.）
- Condition Monitoringライブラリでは、推奨オーバーラップの計算は `F_CM_CalculateRecommendedOverlap` [▶ 270] を介して実施する。この関数は明示的に使用することもできるが、適切に設定されていれば、アルゴリズムは設定したウィンドウ関数に応じて暗黙的に使用することもできる。もちろん、オーバーラップを自由に設定することも可能。例えば、`ST_CM_PowerSpectrum_InitPars` [▶ 295] の `eWindowType`、`eWindowTypeParameters`、`nOverlap` を比較。

オーバーラップ加算法（重畳加算法）

Condition Monitoringライブラリのいくつかのファンクションブロックは、入力信号のスペクトルを操作することで動作します。つまり、上述のように、入力信号はまず、オーバーラップする部分信号に分解され、フーリエ変換されます。その後、スペクトルの操作が行われ、逆フーリエ変換が計算されます。使用するウィンドウ関数によっては、ウィンドウの影響を補正するための補正関数が必要になります。重なり合った個々の結果は、ファンクションブロックの出力で合計され、ファンクションブロックの出力で再びデータストリームが作成されます。この手順はオーバーラップ加算法と呼ばれ、信号包絡線（エンベロープ）（`FB_CMA_Envelope` [▶ 137]）の計算に基づく例として以下の図に示します。



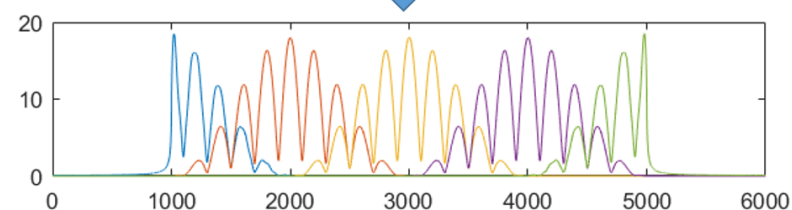
complete data stream
sectioned into buffers



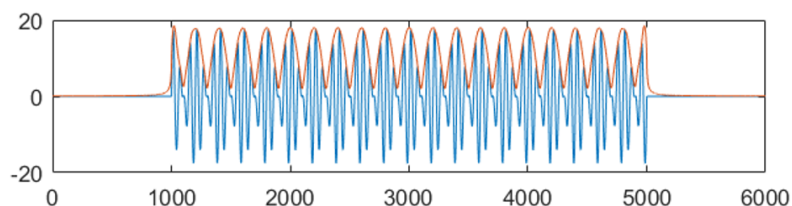
windowing of 2 data buffers
with 50% overlap

Individual processing of each windowed signal part

- (1) Apply FFT
- (2) Manipulation of signal
- (3) Apply inverse FFT



result of each
windowed signal part



overlap add for reconstruction
into stream analysis

● TwinCAT 3でのオーバーラップ加算

i この方法は、ライブラリのいくつかのファンクションブロック内で使用します。ユーザ自身が実装する必要はありません。必要なのは、パラメータ（バッファの長さ、FFTの長さなど）の設定とデータバッファの提供だけです。

2.1.3 一定期間のトリガー解析

動機

連続的に回転するシャフトの振動挙動など、プロセスの連続時間解析に加えて、定義された時間枠の解析も頻繁に応用されます。応用例としては、ドリルヘッド、ミリングユニット、あるいは特定の時間だけ回転するシャフトの振動信号の解析などがあります。

この場合に特にコントローラに統合された解析のメリットは明らかです。制御は通常、穴あけなど特定の加工ステップを開始します。したがって、逐次機械制御は加工ステップだけでなく、解析ステップのトリガーとしても使用できます。

Condition Monitoringライブラリでの実装

解析機能に関しては、定義された時間枠の評価と、データストリームの連続的な解析との間に、ほとんど違いはありません。唯一の違いは、それぞれのトリガー解析が独立していること、つまり連続的でないことです。したがって、TC3 Condition Monitoringライブラリの全ての解析ファンクションブロックは、連続解析およびトリガーによる時間ウィンドウ解析に使用することができます。解析チェーンを構成する際に注意すべきは2点のみです：

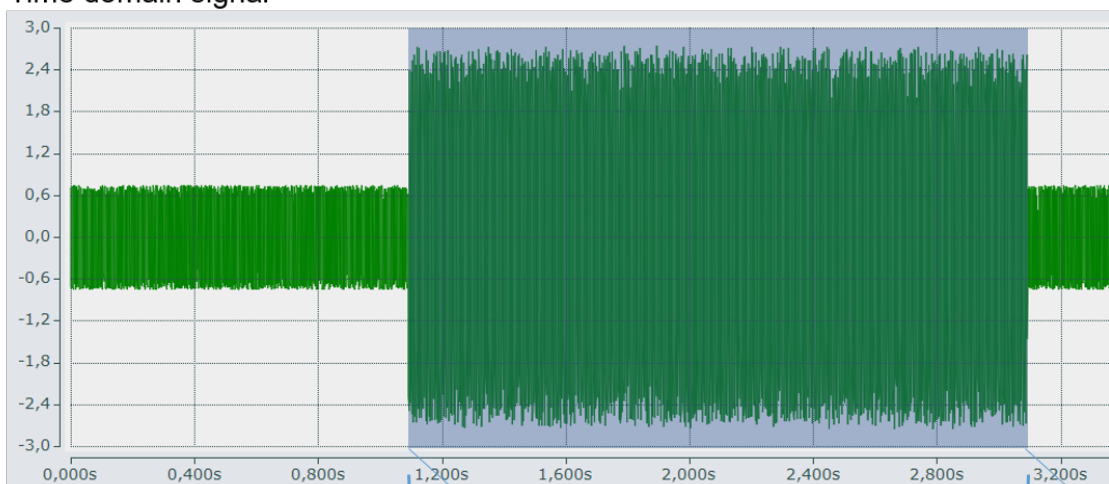
- 有効な結果を算出するためには、FB_CMA_Sourceを介して十分なデータ（十分に大きなデータウィンドウ）が解析チェーンに送信されなければならない。例えば、FFTベースのアルゴリズムのウィンドウの長さを観察しなければならない。
- 各解析を明確に分離するためには、メモリ特性を持つ全ての解析ファンクションブロックが、解析が完了したらリセットされるようにするだけで十分（各アルゴリズムのドキュメント「メモリ特性」の項を参照）。FB_CMA_SourceはこのためにメソッドResetAnalysisChain()を提供。あるいは、ResetData()というメソッドを個々の解析ファンクションブロックに使用することも可能。この場合、メソッドは個々の解析ファンクションブロックでのみ機能する。

サンプルの実装

合成信号に基づくサンプルを以下に説明します。合成信号は、バックグラウンドノイズと、周波数200Hz、振幅2の正弦波信号が加算的に重畳されたもので、正弦波信号は常に2秒ごとにオン・オフされます。

このような信号に対して連続評価が選択された場合、評価に使用される信号セグメントがどの時間間隔にあるかを決定することはできません。したがって、正弦波信号がオンになった時点で、定義された計測時間の評価ウィンドウを常に開始することをお勧めします。以下の模式図は、説明した合成信号と、指示された評価ウィンドウに基づく振幅スペクトルを示しています。

Time domain signal

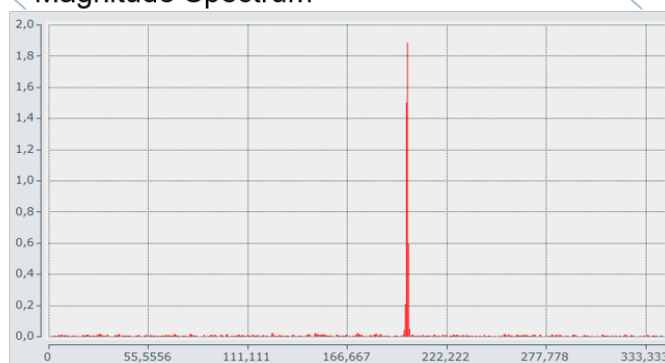


start

window of interest

stop

Magnitude Spectrum



ソースコードとサンプルのより詳細な説明は、以下を参照：[イベントベースの周波数解析 \[▶ 347\]](#)

2.1.4 スペクトルのスケーリング

マグニチュードとパワースペクトル

スペクトルを評価するための一般的な方法はいくつかあります：

- `FB_CMA_MagnitudeSpectrum` [▶ 161] は、複素数値のスペクトル値 $|X[k]|$ の大きさを線形にスケーリングした値を使用します。量スペクトル、振幅スペクトルとも呼ばれます。
- `FB_CMA_PowerSpectrum` [▶ 189] の値はマグニチュード値 $|X[k]|$ の2乗を表します。

2乗スペクトル値 $|X[k]|^2$ は、パースバルの定理により、時間信号のRMS値に正確に関連するため、パワー値が加算または統合される場合、パワースペクトルを使用することは理にかなっています。

パースバルの定理によれば、時間表現における信号 $x[n]$ のパワーは、フーリエ変換における信号のパワーに等しくなります：

$$\sum_{n=0}^{N-1} |x[n]|^2 = \frac{1}{N} \sum_{k=0}^{N-1} |X[k]|^2$$

ここで、信号 $x[n]$ のRMS値を計算すれば、時間領域でも周波数領域でも実現できます。どちらの表現もパワーに関しては同じためです：

$$\text{RMS}\{x[n]\} = \sqrt{\frac{1}{N} \sum_{n=0}^{N-1} |x[n]|^2} = \sqrt{\frac{1}{N^2} \sum_{k=0}^{N-1} |X[k]|^2}$$

実際には、例えば信号の限られた周波数領域についてRMS値を計算することができます。Condition Monitoringライブラリの実用的なスペクトルスケーリングのオプション [▶ 371] に、このセクションで言及されるプロパティに関連するもの `eCM_ROOT_POWER_SUM` および `eCM_RMS` が含まれます。

パワースペクトル密度

スペクトル解析におけるもう1つの重要な概念は、パワースペクトル密度（PSD）です。等価雑音帯域幅（ENBW）で示される実効周波数分解能に基づく出力値を指します。

$$\text{PSD} = \frac{1}{\text{ENBW}} |X[k]|^2$$

信号、マグニチュードスペクトル、PSDの物理単位を見れば、その関係がよく分かります。信号 $x[n]$ がボルト（V）で計測される場合、離散マグニチュードスペクトル $|X[k]|$ もVで記述されます。2乗は、パワースペクトルが V^2 で表記されていることを意味します。定義では、パワー密度スペクトルは、Hz単位の周波数に基づくパワー値（ V^2 ）です。パワースペクトルをヘルツ（Hz）単位の有効周波数分解能に関連付けると、 V^2/Hz という単位になります。

この表現はマグニチュード値にも使用できます。これに対応して、線形スペクトル密度（LSD）は次のようになります。

$$\text{LSD} = \sqrt{\frac{1}{\text{ENBW}}} |X[k]|$$

デシベルスケール

振動解析や機械音響学では、線形から対数の「デシベルスケール」への数値変換が一般的です。デシベルスケールは、スペクトル中に非常に大きな値と非常に小さな値の両方が発生し、大きな値と小さな値の両方の領域で評価される場合に良好な解釈を可能にします。マグニチュードスペクトルは、以下のとおりデシベルスケールに変換されます：

$$|X[k]|_{\text{dB}} = 10 \log_{10} |X[k]|^2 = 20 \log_{10} |X[k]|$$

デシベルスケールは、パワースペクトルの10倍対数とマグニチュードスペクトルの20倍対数の両方で計算できます。したがって、[FB_CMA_MagnitudeSpectrum \[▶ 161\]](#)と[FB_CMA_PowerSpectrum \[▶ 189\]](#)の計算結果は、デシベル単位では同じです。

デシベルスケールへの結果の変換は、Condition Monitoringライブラリによって、ファンクションブロックの初期化パラメータ内のbool変数（例：[ST_CM_PowerSpectrum_InitPars \[▶ 295\]](#)）を介して、便利に有効化されます。

信号タイプに応じたスケーリングオプション

適切なスペクトルスケーリングオプションを[\[▶ 371\]](#)選択することで、[FB_CMA_PowerSpectrum \[▶ 189\]](#)または[FB_CMA_MagnitudeSpectrum \[▶ 161\]](#)ファンクションブロックによって計算されたスペクトル値を、目的の参照変数に自動的に調整できます。ここで特に重要なのは、参照変数の正しい解釈です。

スケーリングオプションではまず、実際に定常信号を仮定して、決定論的信号と確率論的信号を区別することが重要です。

決定論的信号は、定義された周波数を持つ周期的振動で構成されます。ここで決定的なのは、周波数分解能（ENBW）が高調波周波数よりも広いことです。したがって、信号のこの周波数成分の全パワーは、この周波数チャンネルに集約されます。よってスペクトル値は、振幅（[スペクトルスケーリングオプション \[▶ 371\]](#) `eCM_PeakAmplitude`）または等価な正弦波信号のRMS値に直接スケーリング可能です。信号が周波数チャンネルの中心にこない場合、いわゆるスカロッピング損失が発生し、観測される最大振幅を減少させます（[ウィンドウ関数 \[▶ 19\]](#)の項と比較）。フラットトップウィンドウの使用とは別に、例えば、隣接する周波数チャンネルの電力値を合計値として評価するHannウィンドウを使用することで、これを逆及的に補正することができます（[スペクトルスケーリングオプション \[▶ 371\]](#) `eCM_ROOT_POWER_SUM` および `eCM_RMS`を参照）。

確率的または広帯域の信号では、定義された周波数領域にわたって全ての周波数が信号パワーを含むため、パワースペクトル密度（PSD）または線形スペクトル密度（LSD）を評価する必要があります。この場合、決定されるパワー値は、FFTの周波数チャンネルの有効幅に依存します。論理的には、評価パラメータに依存しない結果を得るためには、この帯域幅を参照する必要があります。ウィンドウ関数を使用する場合の周波数チャンネルの有効幅は、ウィンドウ関数の長さや形状に依存するため、この場合は前述の等価雑音帯域幅（ENBW）を使用する必要があります（[スペクトルスケーリングオプション \[▶ 371\]](#) `eCM_PowerSpectralDensity`を参照）。

PSDに基づくスケーリングでは、"DC成分"の一貫したスケーリングはできません。必要であれば、ローパスフィルタまたは平均化によって決定すべきです。

信号が決定論的な部分と広帯域の部分の両方を含む場合、処理パラメータに依存しない値を得るためには、両方のスケーリングを互いに独立して使用しなければなりません。例として、高調波正弦波と帯域制限ノイズからなる信号の評価があります。高調波正弦波の振幅を評価する場合は、決定論的信号に対するスケーリングを実施する必要があります。確率的なバックグラウンドノイズを評価しようとするれば、PSDまたはLSDでスケーリングしなければなりません。

● Condition Monitoringライブラリによるスペクトルのスケーリング

I 様々なスケーリングオプションは、Condition Monitoringライブラリにすでに実装されており、初期化パラメータでファンクションブロック固有の構造を介してパラメータ設定できます。[E_CM_ScalingType \[▶ 273\]](#)および[スペクトル・スケーリング・オプション \[▶ 371\]](#)を参照。これに関するチュートリアルは：[スペクトルのスケーリング \[▶ 320\]](#)を参照。

参照

スケーリングの分類

計測技術において、絶対的な計測値の比較は非常に重要ですが、ISO10816-3に準拠した[振動評価 \[▶ 33\]](#)や機械保護では、トレンドや比較ベースの状態監視に絶対的な校正は不要です。

多くの場合、特定の機械に合わせたものではない一般的な制限値は、損傷の早期診断検出にはあまり適していません。計測ポイントの選択（計測場所、センサのカップリングなど）は伝送リンクの減衰係数に大きな影響を与えるため、**トレンド監視**では、選択したテストポイントとカップリング条件を一貫して維持することがより重要になります。多くの場合、最初は信号レベルが低い信号成分が重要になります。信号が周期的である場合、可能な限り狭い帯域幅で適切な統計関数を用いた高分解能FFTスペクトルを用いると、特に明

瞭かつ早期に現れます。Condition Monitoringでは、通常、長期にわたるトレンド監視とデシベルスケールの相対比較が、個々の絶対値よりもはるかに重要な役割を果たします。これは、センサの場合、一般的には高価で高精度な絶対校正と滑らかな周波数応答は、高い長期安定性と十分に低い温度依存性よりも重要ではないことを意味します。ただし、これは校正を完全に無視できるという意味ではありません。

参照信号に基づくスケーリング

多くの場合、計測値の数学的参照（参照によるスケーリング）は、一見したところよりもはるかに複雑です。様々なパラメータに非線形に依存する複数のステップを含む処理を行う場合、校正装置の助けを借りてスケーリングを行う方が簡単で、何よりも誤差が生じにくい場合が多いです。ここでは、計算されたスペクトルのマグニチュード値が常に入力値に対して線形であるという事実を利用します。したがって、信号を正しくスケーリングするためには、よく知られた基準入力値に基づいて、関連する線形係数を決定するだけです。専門的には、校正装置を使用して定義された振幅（またはRMS値）を持つ物理信号を生成し、出力値を計測し、入力と出力の比として必要な補正係数を決定することによって行われます。基準信号に基づいてスケーリングすることの大きな利点は、加速度センサの損傷や計測システムの誤った設定などの物理的欠陥を確実に発見できることです。評価時に多数のパラメータの組み合わせをテストする場合、この方法には限界があります。

2.1.5 統計解析

状態監視は、制限値のモニタリングに使用します。値の超過はメッセージや警告の原因になります。実際には、FFTの個々の値はしばしば大きく変動するため、平均化やその他の統計解析が必要になります。個々の値を解析すると、高い値が制限値を超過する結果になります。

基本概念

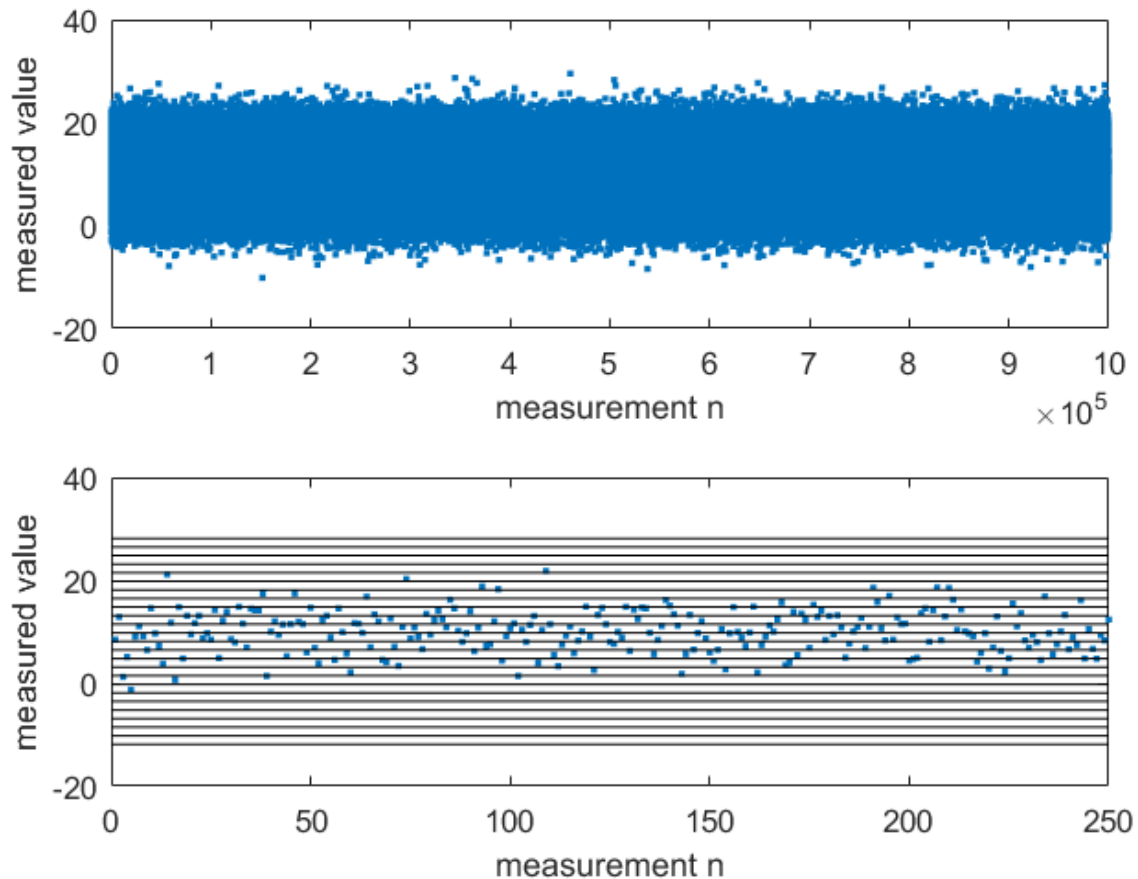
実際のプロセスで、量（温度、圧力、電圧など）が計測される場合、繰り返し計測では、前の計測値が繰り返し計測で決定された値と一致しない可能性が非常に高いです。ランダムに変動する数量のシーケンスは決定論的に（つまり具体的な方程式を介して）決定することができないため、このような信号を記述するために統計的パラメータが使用されます。実際には、決定論的信号と確率論的信号がしばしば重畳される（例えば、直流電圧に計測ノイズが重畳される）という事実は関係ありません。要約結果はランダムであり、したがって確率信号です。

ランダムに変動する数量の個々の計測は、ランダム事象です。個々の計測は、ランダム実験の実現と呼ばれます。ランダム実験で N 個の無作為サンプルを取った場合、この実現数がサンプルサイズとなります。

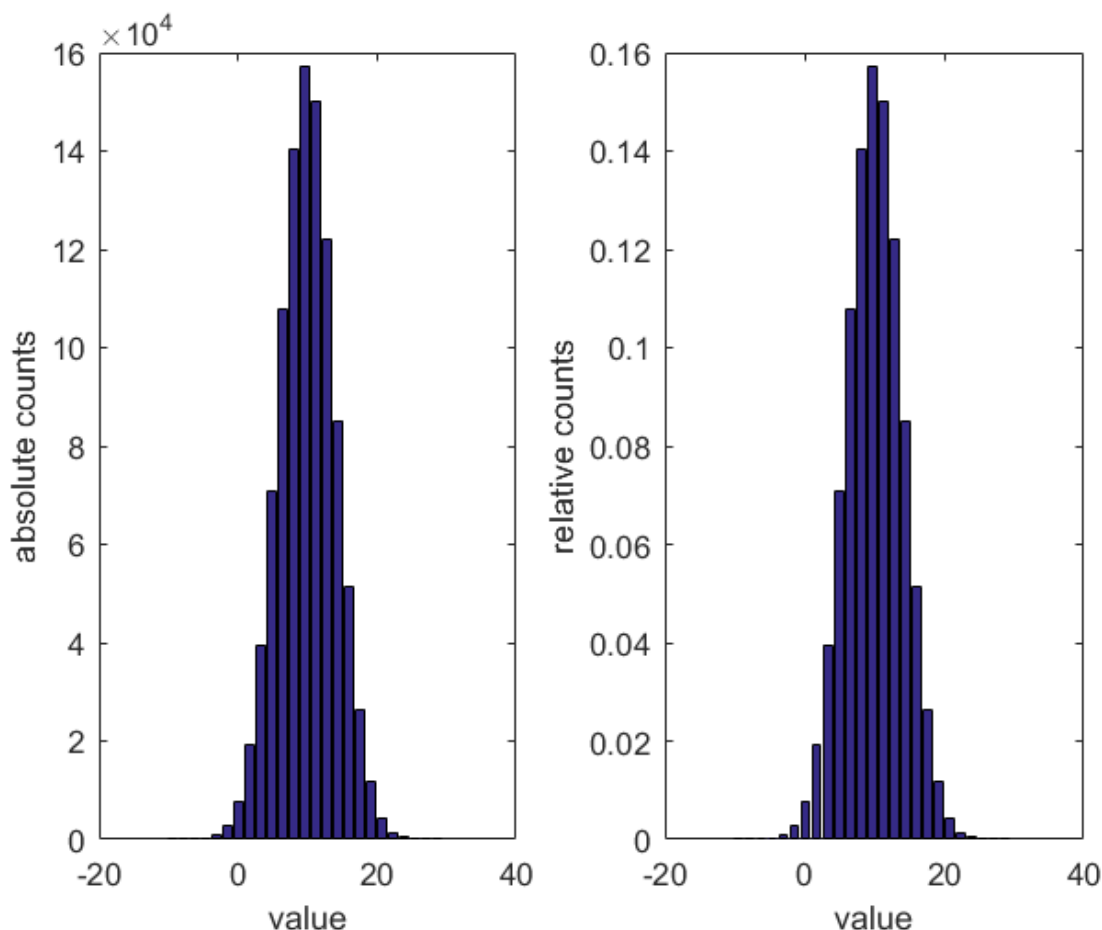
ヒストグラム

ランダム事象の中心的な特性は、計測されたパラメータが特定の値をとる確率です。これは絶対または相対度数分布によって記述され、ヒストグラムで表されます。

簡単な例：10Vの計測変数に正規分布ノイズ（平均値0V、標準偏差4V）が重畳されていると仮定します。このパラメータの計測を100万回繰り返すと、下図（上部）になります。100万回のランダム実験の実現値をヒストグラムで示すと、概要がよく分かります。絶対度数分布は、計測された変数範囲がクラス（ビン）に細分化されるように生成できます。図の上部は、個々の計測において計測された変数を示し、下部は最初の250の計測とヒストグラムのクラス限界のみを示しています。



絶対度数分布は、単純にクラス（ビン）内にある計測値の数から得られます（下図左を参照）。この分布は、考慮されるクラスの数に基づいてパラメータ設定され、クラスが多ければ多いほど、分布は細かくなります。相対度数分布は、サンプルサイズを参照することにより、絶対度数分布から計算できます（下図右を参照）。これは、計測回数に依存せず、値が計測された確率を示します。例えば、10V付近のクラスの値は、 $0.157=15.7\%$ の確率で計測されました。



実験的に調査されたプロセスは、最初は度数分布に基づいて簡単に視覚的に評価できます。3つの疑問があります：

- 計測値の散乱の強さ
- 計測値は（上記の10V付近のように）1つの値の周辺に散らばっているか、その他の値の周辺に散らばっているか
- どのように分散しているか - 正規分布、スチューデントのt分布、カイ2乗分布

● TwinCAT 3での絶対度数分布の計算

I Condition Monitoringライブラリを使用すると、ファンクションブロックFB_CMA_HistArray [▶ 145]で、絶対度数分布を簡単に計算できます。ファンクションブロックのパラメータ設定に必要なのは、対象範囲とクラス数だけです。TwinCAT Scope Viewの配列棒グラフでグラフィック表示が可能です。サンプルは、[ヒストグラム](#) [▶ 325]からダウンロードできます。
統計的手法 [▶ 327]のサンプルは、データ評価のためのCondition Monitoringライブラリのその他のオプションを示しています。

通常モーメントと中心モーメント

ランダムなプロセスの複数の観測に基づいて、実際の値にできるだけ近い値を推定することができます。これは最良推定値と呼ばれます。この目的のために、特性の異なる推定量（例えば算術平均）を使用することができます。最良推定値の算出に加えて、多くの場合、推定値の不確実性を表現することも重要であり、これは通常、実験的標準偏差（経験的標準偏差とも呼ばれる）を用いて算出されます。

例えば、モーメントは与えられたサンプルから統計変数を計算するのに非常に適しています。平均値、分散、歪度、尖度などは、特に与えられた操作回数から統計パラメータを計算するのに適しています。平均値は適切な推定値を提供しますが、他のモーメントはこの推定値周辺の値の分布についての洞察を提供します。

サンプルに基づく図：

ヒストグラムの項で説明した上記のサンプルは、10Vの「真値」を持ち、後からノイズが加えられました。100万個のサンプルから平均値を計算すると、9.9977Vになります。これが真値の最良の推定値です。この平均値に対する分散は16.01V²です。分散の平方根は標準偏差に対応し、4.0013Vです。計測値の分布がこのケースのように正規分布であれば、計測値の分布はこの2つのモーメントで完全に記述されます。つまり、歪度と尖度は（理論的には）ゼロとなります。歪度は平均値を中心とした分布の対称性を表し、尖度は分布関数の急峻さ（ピーク性）を表します。

推定結果の不確実性を評価する：

1995年、計量標準合同委員会（JCGM）は、計測の不確実性の記載に関するガイドを発表しました。JCGMは、BIPM、IEC、IFC、ISOなどの中央統括組織で構成され、共同でこのガイドを作成しました。基本論文「計測における不確実性の表現ガイド」（GUM）は、BIPMのウェブサイトから無料でダウンロードできます。以下に、その中心的な考えを紹介します。

上記のように、与えられた N 個の観測値の集合から最良推定値を計算できます（平均値 = 標本平均）。最良推定値の分散は、観測値の集合の分散（標準偏差）ではなく、不確実性の値として計算され、使用されます。推定値の不確実性評価が目的であるため、これは理にかなっています。最良推定値の分散は、観測値の集合の標準偏差を N の平方根で割ることによって単純に計算できます。サンプルサイズが十分に大きければ、不確実性値に2倍（あるいは、より大きな係数）をかけることで拡張不確実性の値を計算できます。この拡張不確実性の値をプラス/マイナスした平均値は、真の計測値を95%の確率で含みます。

したがって、Condition Monitoringライブラリのアルゴリズムは、計測の不確実性についてGUMに準拠した記述を行うために使用できます。

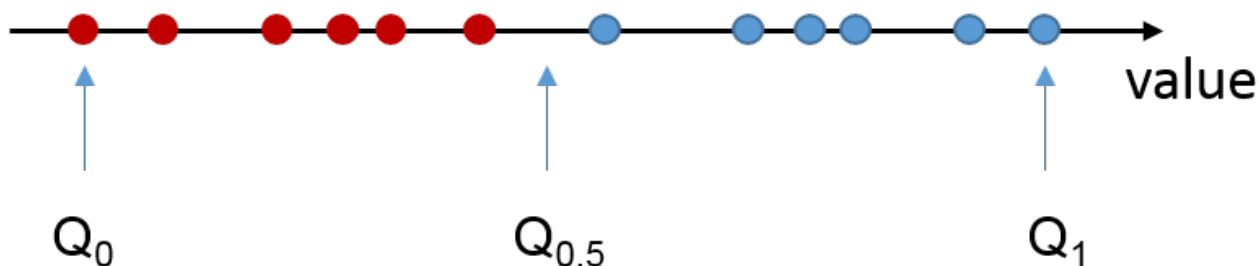
● TwinCAT 3でのモーメントの計算

1 Condition Monitoringライブラリでは、サンプルの1次から4次モーメント（平均、分散、歪度、尖度）をファンクションブロックFB_CMA_MomentCoefficients [▶ 172]を使用して計算できます。このファンクションブロックは、使用するサンプルサイズのみパラメータ設定する必要があります。

分位数

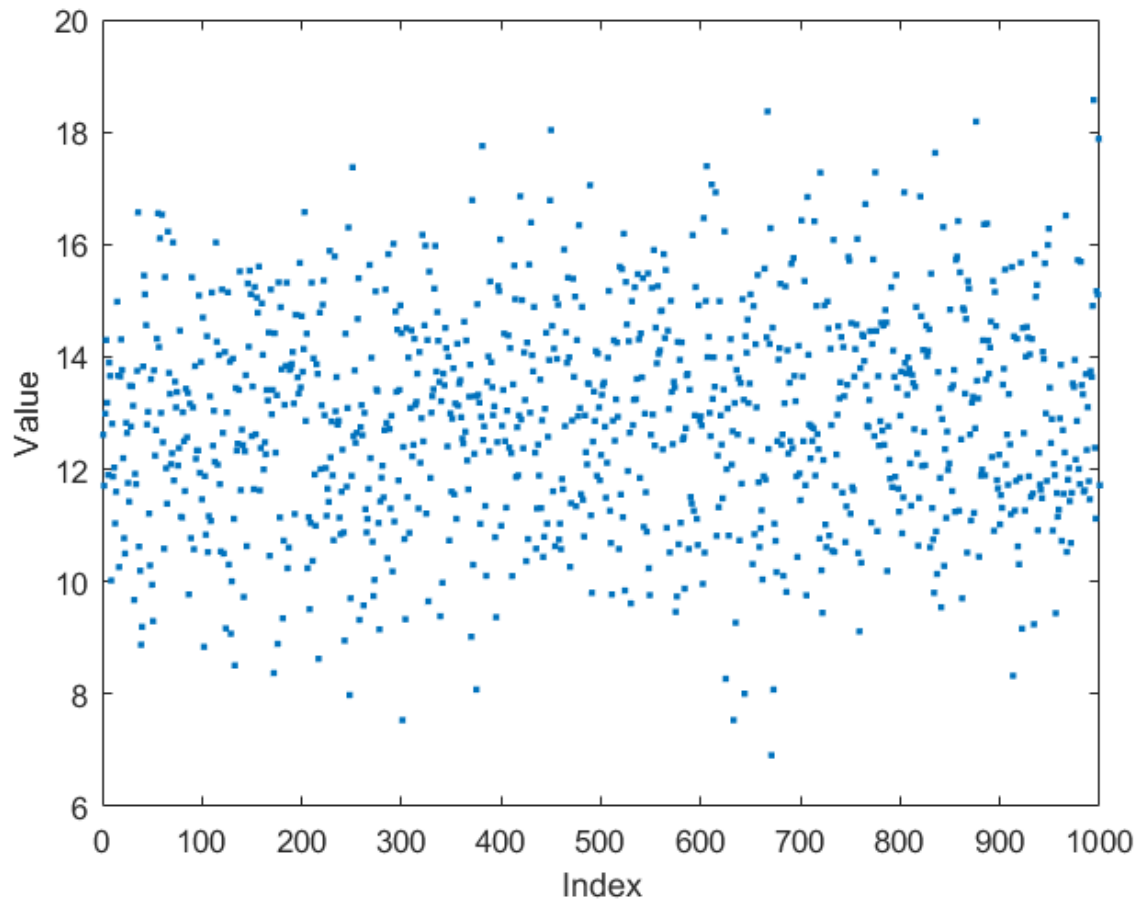
確率変数 x の p 分位数 Q_p は、 x の全ての実現数のうち、 p の割合に対して Q_p が x より大きい値です。もう少し記述的に定式化すると有限個の値が与えられた場合、 p -四分位数はデータを2つの領域に分割します。例えば、50%分位値（中央値）は、全値の少なくとも50%以下の値を示します。この値をサンプルの平均値と混同してはいけません。

p の値は0から1の間にあります。 p パーセントで指定されている場合は、パーセンタイルが考慮されます。 $Q_{0.5}$ は中央値、 $Q_{0.9}$ は90%パーセンタイル、 Q_1 は一連の観測値の最大値です。

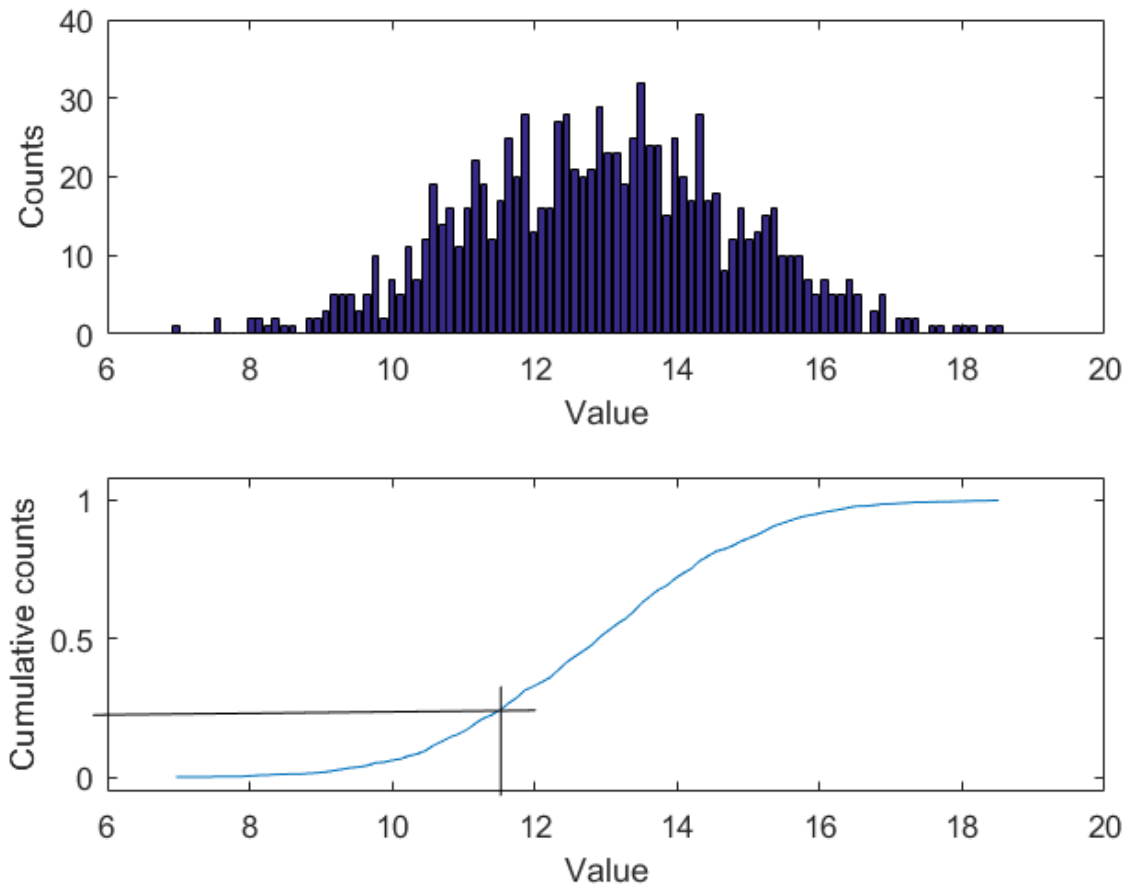


p が1の値に近づくほど、 Q_p は外れ値や極端な個体値によって決定され、 p が0.5の値に近づくほど、 Q_p は中央値に近づきます。中央値は外れ値に対して非常に頑健です。実行時にTwinCATで設定できる p の値を使用すると、個々の値に対するサンプルの評価感度を動的に変更できます。

次の図は、分位の基本的な考え方を説明するために平均値13を中心に1000個の値が散らばった列を示しています。



値列から、ある値が対象列（サンプル）においてどれくらいの頻度で出現するかを示すヒストグラムを計算できます。ヒストグラムに示された絶対度数を積分し、対象となるシーケンスの値の総数（この場合は1000）を参照することで、経験的総和度数分布を計算することができます。それに基づき分位数を簡単に読み取ることができます。例えばこの場合、25%分位値は11.8です。つまり、1000個のサンプル値のうち、少なくとも25%はこの値より下にあります。



FB_CMA_Quantiles [▶ 193]を計算するライブラリのファンクションブロックは、2つのサブステップで動作します。これらは、一緒に呼び出すことも、別々のサブステップで呼び出すこともできます。最初のステップでは、内部ヒストグラムに値が追加されます。このヒストグラムのパラメータは事前に設定できます。このステップに必要な計算量はごくわずかです。第2のステップでは、保存されたヒストグラムから、先に選択された分位数が計算されます。構成にもよりますが、この2つ目の操作は、より複雑な操作によって定義されるため、計算量ははるかに多くなります。しかし、実行頻度ははるかに少なくて済みます。

● TwinCAT 3での分位の計算

i ファンクションブロックFB_CMA_Quantiles [▶ 193]は分位の計算に使用できます。1回のファンクションブロック呼び出しで、複数の分位数を計算できます。このファンクションブロックは、ヒストグラムのファンクションブロックと同様にパラメータ設定します。計算する分位数や使用するサンプルサイズもパラメータ設定します。

2.2 アプリケーションの概念

ここでは、Condition Monitoringタスクのアプリケーションとソリューションの基本的なパターンを説明します。プログラミングやインターフェースの詳細については触れず、代わりに基本的なアプローチや概念を説明しています。各コンセプトの最後には、Condition Monitoring ライブラリを用いた実装例を示しており、ライブラリを使用して何ができるのか、その全体像を把握できるようになっています。

学習内容：

- ISO 10816-3に準拠した振動監視はどのように機能するか
- 周波数帯域での閾値監視はどのように機能するか
- ローラーベアリングの状態監視はどのように構成されるか
- ギアユニットの状態監視はどのように構成されるか
- 損傷状態の監視はどのように実装するか

2.2.1 振動評価

導入による曖昧さの解消

振動評価は、振動の計測によって機械の運転状態を評価し、機械の信頼性と安全性を確保することを目的としています。機械部品のローカル診断/解析は、本書の範囲外です。ローラーベアリングやギアユニットのような部品の診断による状態監視ソリューションは、以下で個別に説明します。

共通規格への参照

機械の振動を評価するための規格には以下があります：

- ISO 5348、機械的振動および衝撃-加速度計の機械的取付け
- ISO 10816、機械振動-非回転部品の計測による機械振動の評価（旧VDIガイドライン2056）この規格は複数のパートで構成される
 - ISO 10816-3は、定格容量が15kWを超え、定格回転数が120rpmから15000rpmの産業機械を対象とし、現場で計測される
 - ISO 10816-7は産業用の遠心ポンプを対象とする
 - ISO 10816-21は、水平軸とギアボックスを備えた風力タービンを対象とする
- ISO 7919、機械振動-回転軸の計測による機械振動の評価 この規格は複数のパートで構成される
 - ISO 7919-3は、連結された産業機械を対象とする
 - ISO 7919-2は、公称運転速度が 1500min^{-1} 、 1800min^{-1} 、 3000min^{-1} 、 3600min^{-1} で、容量が50MWを超える定置蒸気タービンと発電機を対象とする
- ISO 20816-1、機械振動-機械振動の計測及び評価 ISO 7919-1とISO 10816-1の統合

DIN ISO 10816-3に基づく機械振動の評価

この規格の適用範囲には、50MWまでの蒸気タービン、電気モータ、ファンが含まれます。その範囲はかなり広いので、この基準の詳細については以下で詳しく説明しています。この規格は、受入計測と運転監視のための振動データによって、機械の状態を4つの異なるクラスに分類することを目的としています。

規格の評価基準は、振動速度のRMS値と振動変位のRMS値です。通常は振動速度を計測すれば十分です。低周波成分が発生した場合は、振動変位の追加評価が推奨されます。両方の振動パラメータが記録、解析された場合、決定された2つのクラスのうち悪い方が適用されます。

振動を捉えるための周波数領域は、機械の速度に依存します：

- 10 Hz～1000 Hz（600rpm以上の場合）
- 2 Hz～1000 Hz（600rpm未満の場合）

適切な計測ポイントは、機械の動的な力を可能な限り純粋に反映する場所です。例えば、局所的な共振が発生する場所は不適切です。適切な計測場所は、ベアリングスタンドやベアリングカバーとなる場合が多いです。計測は通常、2つの直交方向で行われます。

この分類では、機械の下部構造も考慮されており、剛性下部構造と弾性下部構造に細分化されています。機械と下部構造で構成されるシステム全体の最低固有振動数が、主加振周波数（一般に回転周波数）より少なくとも25%高い場合、下部構造は剛性とみなすことができ、そうでない場合は弾性とみなすことができます。この評価は、各計測方向（直交する2方向、上記参照）に対して別々に実施されるべきです。

DIN ISO 10816-3:2009では、4つの評価ゾーン（A、B、C、D）が規定されており、制限値は以下の表に記載されています。

機械グループ		1		2	
設置方法		剛性	弾性	剛性	弾性
振動速度のRMS値 (単位：mm/s)	11,00 .. ∞	D	D	D	D
	7.10 .. 11.00	D	C	D	D
	4.50 .. 7.10	C	B	D	C
	3,50 .. 4,50	B	B	C	B

	2.80 .. 3.50	B	A	C	B
	2.30 .. 2.80	B	A	B	B
	1.40 .. 2.30	A	A	B	A
	0.00 .. 1.40	A	A	A	A
機械グループ		1		2	
設置方法		剛性	弾性	剛性	弾性
振動変位のRMS値 (μm)	140 .. ∞	D	D	D	D
	113 .. 140	D	C	D	D
	90 .. 113	D	C	D	C
	71 .. 90	C	B	D	C
	57 .. 71	C	B	C	B
	45 .. 57	B	B	C	B
	37 .. 45	B	A	B	B
	29 .. 37	B	A	B	A
	22 .. 29	A	A	B	A
	0 .. 22	A	A	A	A

- ゾーン A

最近立ち上げた機械の振動はこのゾーンにある傾向がある。
- ゾーン B

このゾーンの振動を持つ機械は、通常、制限のない連続運転に適しているとみなされる。
- ゾーン C

このゾーンに振動がある機械は、通常、連続運転には適さないといみなされる。機械は通常、改善措置の適切な機会が訪れるまでの限られた期間、この状態で運転することができる。
- ゾーン D

このゾーンの振動値は通常、機械に損傷を与える可能性があるという意味で危険といみなされる。
- 機械グループ1

定格出力300kW～50MWの大型機械およびシャフト高さが315mmを超える電気機械。
- 機械グループ2

定格出力15kW～300kWの中型機およびシャフト高さ160mm～315mmの電気機械。

ISO 10816-3に則った振動評価は、3つの異なる例で実施されています。以下を参照：

- [ISO 10816-3に準拠した振動評価 \[▶ 328\]](#)
- [ISO 10816-3に準拠した振動評価（コンパクト版） \[▶ 331\]](#)
- [ISO 10816-3に準拠した振動評価（拡張版） \[▶ 333\]](#)

2.2.2 周波数解析

動機

機械の診断/解析モニタリングで最も重要な手法のひとつは、加速度計による振動の記録と、それに基づく周波数解析です。機械は金属でできているため、事実上常に周期的な力を受ける弾性構造であるためです。これにより、それぞれの構造物の周波数と加振力、特性周波数が反映された振動が発生します。したがって、振動計測によって、機械の構造や力に関する結論を導き出すことができます。ベアリングなどの機械要素の損傷や構造変化は、振動パターンに変化をもたらします。

振動は、音波（構造伝播音）の形で機械部品に伝播します。機械は多数の部品から構成され、それらは一方では他の部品に弾性的に振動を伝え、他方ではそれ自体振動するため、個々の振動成分のフィルタリングや重ね合わせが発生します。振動信号はそれに応じて、いくつかの信号成分で構成されており、これらは移動距離に依存する異なる時間遅延と減衰を伴い、総合的な信号として加算されます。そのため、個別に求められた振動成分は、さらなる処理を行わない限りは、全信号の時系列では認識できません。周波数解析の利点は、直線的に重畳する振動を周波数成分に分割できることです。これらの周波数成分は、特定の機械状態、部品、プロセスに容易に割り当てることができます。

成分を周波数選択により監視する概念は、次のように分類されます：

- スペクトルの計算
- 結果の統計的評価
- 閾値の監視

周波数解析の実践的要素

フーリエ解析の主要な側面については、すでにフーリエ解析 [▶ 11] の項で扱いました。ここでは、最も重要な実践的側面についてもう一度おさらいをします。

フーリエ解析のためのファンクションブロック (FB_CMA_MagnitudeSpectrum [▶ 161] や FB_CMA_PowerSpectrum [▶ 189] など) のパラメータ設定をする際、以下の質問が最も重要です。

- 解析対象となる最大周波数はどのくらいか。
サンプリング周波数は、ターミナルのオーバーサンプリング係数によって設定され、関連するタスクサイクルタイムもそれに応じて設定すべきです。エイリアシング対策フィルタの設定にも注意が必要です。サンプリングの定理 [▶ 13] を参照。
- 周波数分解能の要件
計測時間の長さ (入力配列のサイズ) は、それに応じて設定する必要があります。詳細は周波数分解能 [▶ 13] を参照。ウィンドウ関数を適用した場合の周波数分解能の劣化も考慮すべきです。詳細はウィンドウ関数 [▶ 19] を参照。
- FFTの長さは入力配列の長さより大きく、2の累乗である必要があります。余った要素はゼロで埋められます。ゼロパディングまたは周波数分解能 [▶ 13] を参照。
- スペクトルの適切なスケーリングを選択する必要があります。詳細はスペクトルのスケーリング [▶ 25] を参照。



上記の点は、特にFB_CMA_MagnitudeSpectrumとFB_CMA_PowerSpectrumの使用について言及しています。特殊なバリエーションのスペクトル計算を使用する場合、いくつかのポイントを最適化できます。スペクトル値計算の概要 [▶ 15] を参照してください。

統計的評価

フーリエスペクトルは、信号中のノイズや干渉に対して非常に敏感です。したがって、フーリエ変換された実ノイズ信号は、直接的な解析や評価には適していません。これを補うため、マグニチュードスペクトルは通常、平均化されるか、分位数によって評価されます (統計解析 [▶ 27] 参照)。このアプローチは、解析する信号の時間的安定性または周期的反復性を前提としています。この方法で決定されたパラメータは、干渉に対してより頑健であり、視覚的に評価しやすくなります。以下では、複数のスペクトルの平均に基づく評価を例として考えます。



マグニチュードスペクトルの統計的評価

複数のマグニチュードスペクトルを作成し、平均化や分位点計算などで統計的に解析することは理にかなっています。これにより、決定値の不確実性が減少し、閾値解析の信頼性が高まります。

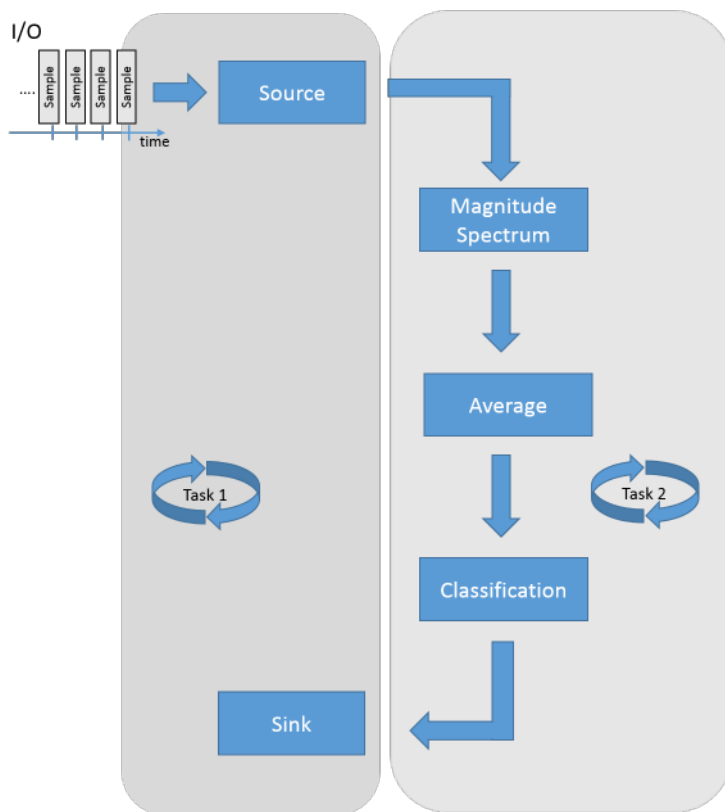
別の方法として、算出されたフーリエ係数を周波数を介して平均化する、つまり隣接する周波数ビンを平均化する方法があります。この目的のため、FFTは上述のスペクトルの経時的な連続平均の方法よりも、高い周波数分解能で計算するべきです。隣接する周波数ビンの平均は、スペクトルの時間平均とほぼ同じですが、計算がより複雑になります。

閾値の監視

ここで説明する概念の最終ステップは、閾値の自動監視です。各周波数チャンネルの閾値は、「通常動作」「警告」「アラーム」など、優先順位の異なる複数のカテゴリーに割り当てられています。これらの閾値は、経験に基づいて設定し、運転中に調整できます。

プロセスの概念

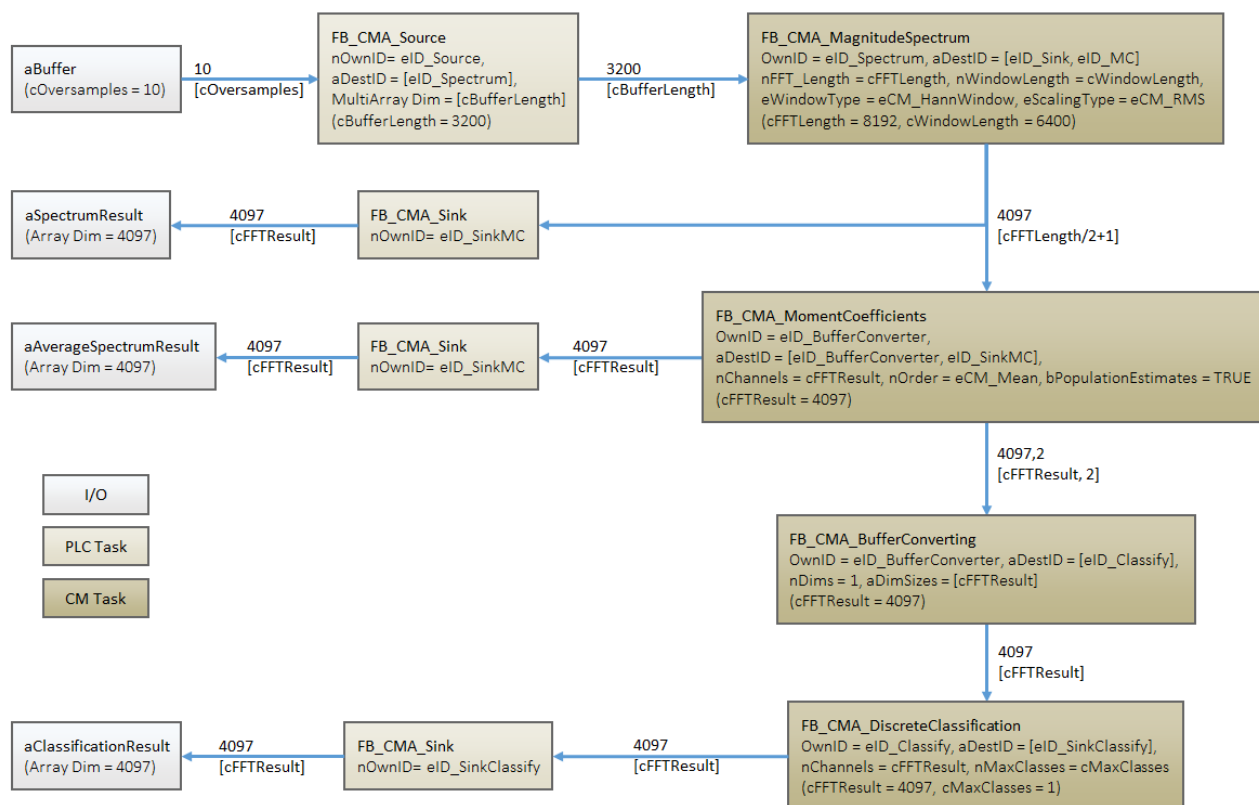
上記の概念は、提供されるファンクションブロックのパラメータ設定により、TwinCAT Condition Monitoringライブラリで簡単に実装できます。以下に設定の例を示します。



平均化したマグニチュードスペクトルの閾値監視を実装します。Condition Monitoringライブラリの以下のコンポーネントを使用します：

- FB_CMA_Source
 - 入力データのバッファ
- FB_CMA_MagnitudeSpectrum
 - 入力バッファのオーバーラップセクションへの配置
 - 計算セクションのウィンドウ処理
 - フーリエ変換の計算
 - フーリエ係数の絶対値の計算
 - 結果のスケーリング（RMS）
- FB_CMA_MomentCoefficients
 - 算術平均値の形成
- FB_CMA_BufferConverting
 - FB_CMA_MomentCoefficientsとFB_CMA_DiscreteClassification間の転送のためバッファ寸法を調整します。
- FB_CMA_DiscreteClassification
 - 計算された各周波数の閾値違反の監視

上記の図をよりソースコードに近い形で表現すると、次のような実装が考えられます：



ここに示した概念のサンプルプロジェクトは、こちらからダウンロードできます：[平均化したマグニチュードスペクトルに対する閾値の考慮 \[▶ 340\]](#)。

マグニチュードスペクトル計算のパラメータ設定

サイクルタイムとオーバーサンプリング係数は、結果のサンプリングレートが10kHzになるように設定されます。次の設定は、MagnitudeSpectrumファンクションブロックとソースファンクションブロックのパラメータ設定のサンプルで使用されています。

```
// constant for input
cOversamples : UDINT := 10; // number of oversamples
cFSample : UDINT := 10000; // lms task with 10 oversamples = 10kHz

// constants for FFT (Magnitude Spectrum)
cBufferLength : UDINT := 3200; // buffer size
cWindowLength : UDINT := 2*cBufferLength; // 50% overlap
cFFTLength : UDINT := 8192; // length of FFT for mag. spectrum, power of 2
cFFTResult : UDINT := cFFTLength/2+1; // result of mag. spectrum
```

周波数分解能は $10 \text{ kHz} / 8192 = 1.22 \text{ Hz}$ です。ゼロパディングと周波数分解能 [▶ 13]で説明したように、これは、隣接する2つの周波数を区別できる周波数分解能には対応しません。この場合、 $10 \text{ kHz} / (2*3200) * 1.5 = 2.34 \text{ Hz}$ となります。2*3200は、FFTの計算に使用する信号区間の長さ（サンプリング値での計測時間）に相当します。拡張係数1.5は、Hannウィンドウ（MagnitudeSpectrumファンクションブロックのウィンドウ処理）の選択によって定義されます。FFT長の8192は、2の累乗を表す2*3200より大きい最小の数です。マグニチュードスペクトルの計算結果の配列の長さは4097です。これはFFTの対称性によって定義されます。

マグニチュードスペクトルの平均化

MagnitudeSpectrumファンクションブロックの結果はFB_CMA_MomentCoefficientsに転送され、平均値（第1中心モーメント）を結果として返すように設定されています。このファンクションブロックは、デフォルトでは、中心モーメントの計算に使用されたサンプルサイズも提供します。このため、結果の配列は2次元になります。本サンプルでは、ファンクションブロックのCallEx()メソッドを使用して25のマグニチュードスペクトルを平均化し、ファンクションブロックをリセットします。この場合、サンプルサイズは常に25なので、この情報は必要ありません。したがって、対応するシンクファンクションブロックは、ファンクションブロックの結果からPLCタスクのコンテキストに平均値だけがコピーされるようにパラメータ設定

されています。さらに、分類ファンクションブロックとMomentCoefficientsファンクションブロックの間でバッファ交換 (FB_CMA_BufferConverting) が適用され、サンプルサイズ情報を含む列も省略されます。

分類

このケースでは、ファンクションブロックFB_CMA_DiscreteClassificationが単純な閾値分類器として使用されます。したがって、2つのクラスしか定義されていません ($nMaxClasses = 1$)。各離散周波数に対して個別に適用される閾値は、実行時に設定されます。サンプルでは、配列インデックス30から50 (約36Hzから61Hzに相当) の閾値は $6V_{RMS}$ に設定され、残りの周波数は $2V_{RMS}$ に設定されています。値が閾値を下回った場合、対応する周波数に対して-1が結果として返されます。閾値を超えた場合は0が出力されます。

サンプルコードの詳細

このプロジェクトには、3軸を持つスコープを含む計測プロジェクトが含まれています。上図は、FB_CMA_MagnitudeSpectrumの結果、すなわち入力信号のマグニチュードスペクトルを示しています。入力信号はファンクション発生器で生成され、周波数50Hz、振幅25Vのノイズの多い正弦波を表します。したがって、非平均化マグニチュードスペクトルの結果は、時間変動 (不確実) です。平均化することで、結果は顕著に安定します。平均化したマグニチュードスペクトルは、スコープの中央に示されます。各周波数の値が閾値以下の場合には-1、定義された閾値を超える場合は0が表示されます。

周波数解析による状態監視 - その他の例

例のセクションには、いくつかのコード例があります。周波数解析による状態監視 [▶ 335] の項には、本項で説明したものと同様の例が記載されています。これは、Condition Monitoringライブラリを使用して作成できる、各ソリューションの柔軟性を説明するためのものです。

2.2.3 ベアリングの監視

動機

ベアリングは、最も一般的で、最も応力のかかる機械要素のひとつです。多くの場合、ベアリングはプラント運転にとって極めて重要です。大型ベアリングの交換部品調達にはダウンタイムだけでも高いコストがかかりますが、小型ベアリングの故障も交換部品のコストをはるかに超える費用が発生することがあります。

損傷の原因

ローラーベアリングの故障には、様々な原因が考えられます：

- ローラーベアリングの自然故障原因は、運転中に転動体の接触面に発生する高い応力による材料疲労です。一定の時間が経過すると、素材のひび割れや走行面の剥離につながります。その結果、小さな欠陥は非常にゆっくりと成長し、疲労寿命の終わりに近づくにつれて速度を増しながら大きくなっていきます。材料疲労のメカニズムは理論的に理解されており、統計的に記述することができます。これらは通常の摩耗の一部です。通常のベアリング設計では、機械の疲労寿命中に重大な損傷が発生する確率が低くなるような寸法が選択されます。したがって、通常の状況下では、正しく寸法決めされ、メンテナンスされたベアリングは、非常に長い疲労寿命を持つことが期待できます。実際に達成される疲労寿命は、かなり短くなる場合が多いですが正確には予測できず、以下の要因で大きく変動します。
- 潤滑剤は応力の一部を分散させ、ベアリングの過熱を防ぐ役割もあるため、不適切な潤滑によって転動体や走行面にかかる応力は著しく増大します。
- さらに損傷の原因として、不良シールや金属切粉などによる汚れがあります。少量の水でも潤滑油は機能しないため、水の浸透は潤滑の失敗にもつながります。
- 取付時の芯出し不良や、組立時の不適切な応力も、原因として挙げられます。
- 過度の応力は走行面の塑性変形 (ブリネリング) を引き起こします。同じような状況は、ベアリングが停止しているときの振動によって引き起こされることがあり、潤滑膜による緩和がない場合にも、これに似た現象 (偽ブリネル摩耗 / false brinelling) が発生することがあります。
- 電気機械では、電流の流れが走行面を破壊する可能性があります。
- 腐食が最初の表面損傷の原因となることもあります。

これらの原因に共通するのは、ローラーベアリング接触面の損傷の早期発見が可能なことです。ベアリングの故障は、圧倒的多数の場合、材料疲労によるものではないという事実から、損傷の早期発見、主要原因の解析と追跡（根本原因故障解析、RCFA）により、中期的な損傷を予防回避し、ダウンタイムコストを削減するだけでなく、ベアリングの耐用年数を大幅に延長することが可能になります。

損傷の結果

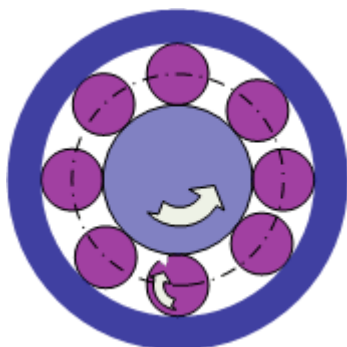
走行面の初期損傷に続き、応力の増大により欠陥が拡大します。走行面以外にも、転動体の保持器など、他の部品も影響を受ける可能性があります。振動は通常、損傷の原因ではなく症状を示すもので、必ずしも損傷の初期段階を示すものではありません。とはいえ、全ての損傷は、遅かれ早かれ接触点の欠陥につながり、それは振動の増加という形で表れます。

監視のアプローチ

最初に直接原因を認識することは困難であるため、ベアリングの走行面に生じる損傷を早期に認識することに重点を置いています。早期発見と調査は、初期の損傷を発見し、その原因に基づいて修正できる可能性が高くなります。このアプローチは、長期的には持続可能なコスト削減につながる人が多いです。さらに、早期発見によってメンテナンス計画が立てやすくなり、プラント運転員にとって何よりもメリットにつながります。もう1つのアプローチは、振動信号を解析することによって関係する要素を特定することです。

以下の信号解析オプションの理解を促すため、欠陥のあるローラーベアリングにおける振動形成について現象学的に紹介します。

ローラーベアリングの概略断面図：



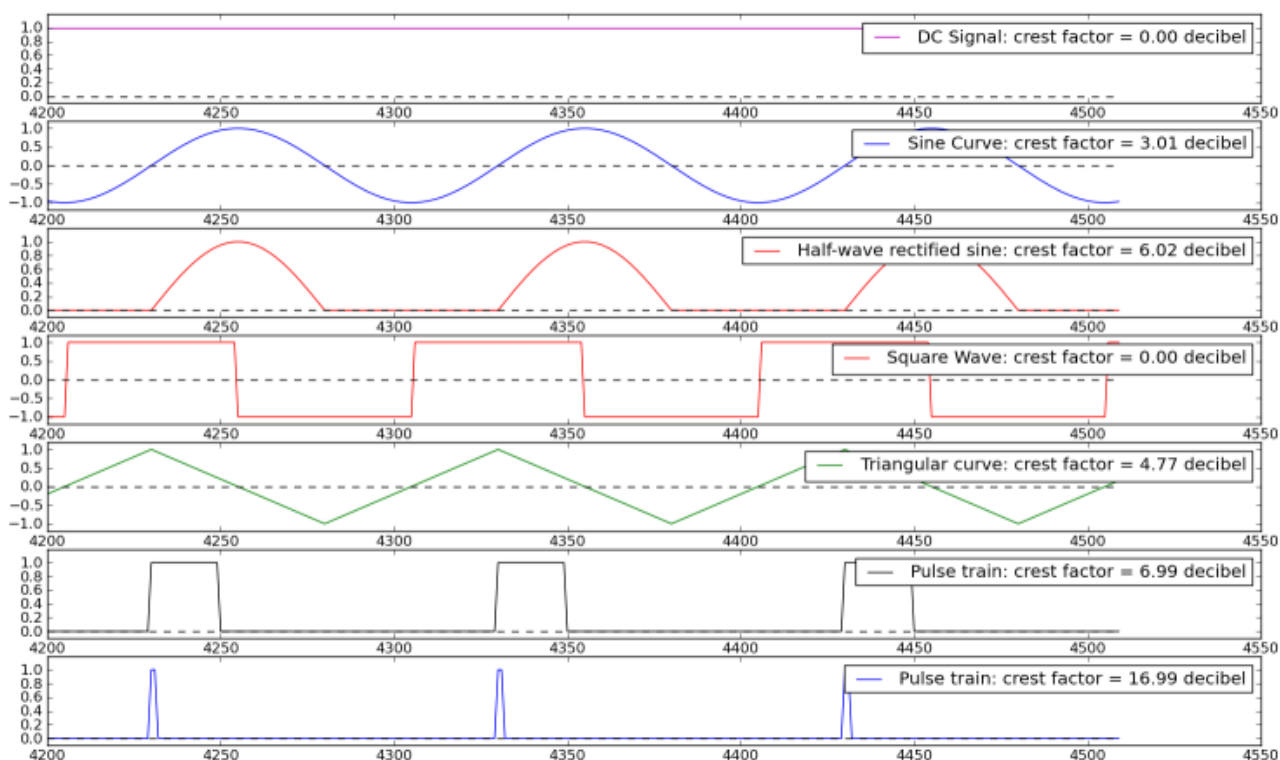
ローラーベアリングの重要な部分は、互いに接触する可動面です。これらは、転動体の表面、内輪の接触面、外輪の接触面にあたります。接触面の局所的な損傷の上を転がると衝撃パルスが発生し、加速度計で拾うことができます。損傷が大きいほど、ショックパルスは強くなります。

時間領域での振動評価

ローラーベアリングの状態を評価する簡単な方法は、振動信号のパルス内容を解析することです。一般的な手法としては、クレストファクタや尖度値の計算が挙げられます。

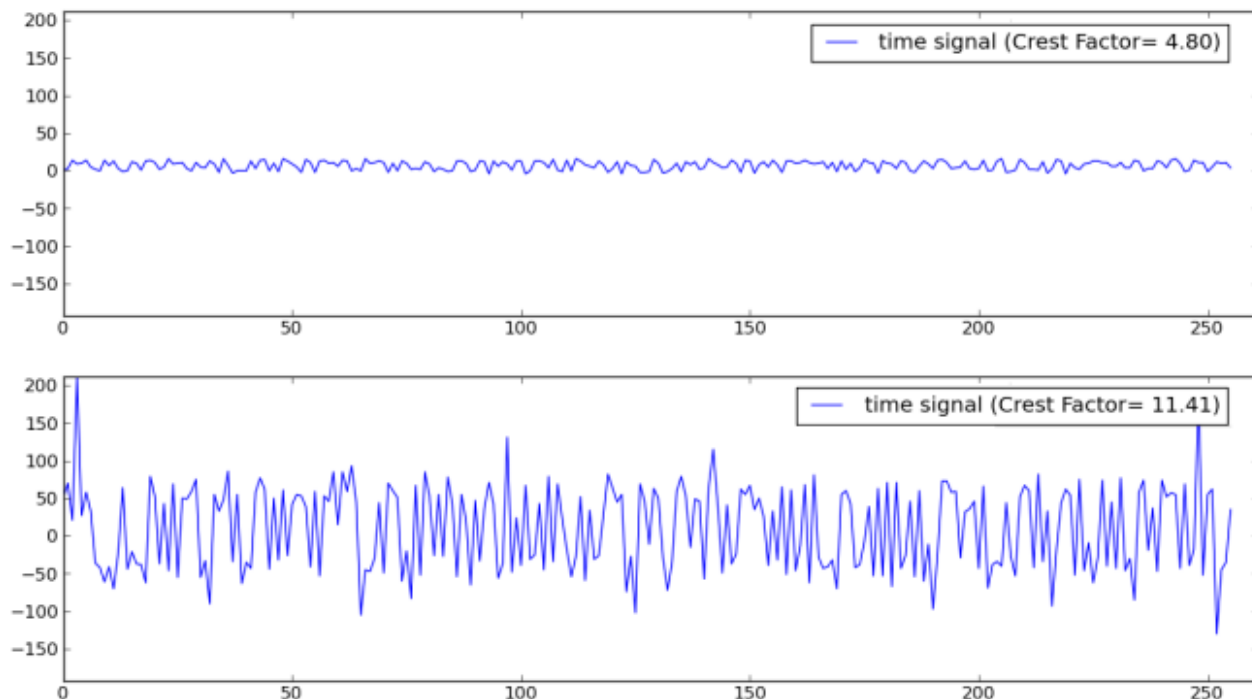
クレストファクタ

クレストファクタは、信号の最大振幅とRMS値の比として定義されます。デシベル単位で指定され、0以上の数値となります。このようにクレストファクタは、最大振幅と実効平均振動振幅の関係を決定します。初期損傷から生じる衝撃パルスは、クレストファクタの上昇につながります。次の図は、信号のパルス含有量の増加に伴うクレストファクタの増加を示しています。



下図は、急性の衝撃パルスに遭遇したときに典型的なクレストファクタの急激な上昇を示しています。クレストファクタの上昇は、損傷の良い指標となります。そのため、この変数は損傷の早期発見や傾向解析に適したツールです。

ローラーベアリングからの信号は次のように解釈できます。



上の図は、摩耗の異なるベアリングからの2つの振動信号を、それぞれのクレストファクタとともに示したものです。下部の信号列は、損傷に起因するピークを明確に示しています。損傷していないベアリングのクレストファクタは4.8dBであるのに対し、損傷した部品のクレストファクタは11.4dBであり、損傷の存在を明確に示しています。

クレストファクタは、計算が非常に効率的で、解釈しやすいという利点があります。加えて、経時変化を簡単に図に表示できます。これを正しく使用するには、この種の評価の基本的な限界を理解することが重要です：

- クレストファクタは信号の最大値に強く影響されるため、統計的な意味で頑健なパラメータではありません。
- 局所的な損傷が大きくなるほど、クレストファクタは大きくなります。しかし、損傷がある程度を超えると、局所的な欠陥の数は増加するものの、衝撃パルスのピーク値はあまり増加しません。その結果、信号の最大値は増加しないが、信号のRMS値は増加し続けます。結果的に、損傷の激しいベアリングの場合、クレストファクタは再び低下します。

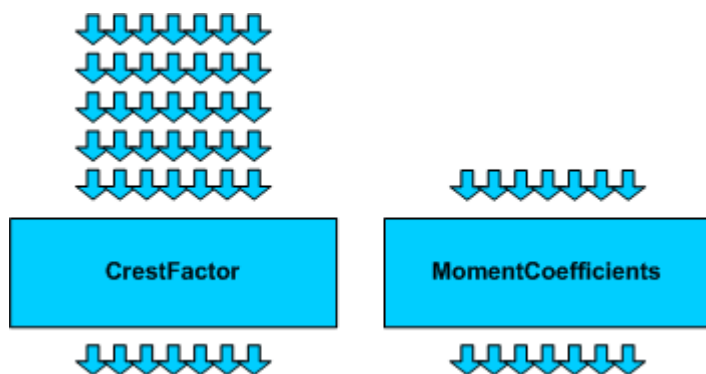
このため、クレストファクタを継続的に計測し、その結果を傾向の観点から解析することが望ましいです。

尖度の値

クレストファクタの統計的な頑健性が限定されていることに起因する問題が発生する場合もあります。より頑健だが、計算量がやや多くなるパラメータに尖度値があります（曲率、第4中心モーメントとも呼ばれる）。平均値や分散と同様に、尖度はいわゆるモーメント係数であり、パラメータを統計的に記述するために使用できます。尖度は、分布の極値（平均値から遠い値）と平均変動との比率を表します。計測系列に時折見られる異常値は結果に大きな影響を与えないため、尖度はクレストファクタよりも統計的にはるかに頑健です。

実際には、尖度はクレストファクタと同様に使われる傾向があります。尖度（または過剰度）およびその他の一般的な統計モーメント係数は、TwinCAT 3 Condition Monitoringライブラリで **MomentCoefficients** [▶ 172] ファンクションブロックを使用して計算します。

プロセスの概念



上図は、クレストファクタと尖度を計算するために利用可能なファンクションブロックを示しています。FB_CMA_CrestFactor [▶ 93] ファンクションブロックは、各チャンネルの個別値の数が同じであれば、複数のセンサからのデータを同時に評価できます。戻り値は、各チャンネルの個別の値で構成されます。個々の値はベクトルで返されます。上図では、水平方向と垂直方向の矢印で示されています。各ケースのクレストファクタファンクションブロックは、7チャンネル（水平）に対して5つの個別値（垂直）を含み、7チャンネルそれぞれのクレストファクタを返します。

尖度はFB_CMA_MomentCoefficients [▶ 172] のファンクションブロックで評価できます。ここでは、値が全てのチャンネルおよび個別の時間ステップについて交互に転送されるか、複数の時間ステップにわたってブロック単位で転送されます。後者は、オーバーヘッドが小さいため、シングルチャンネル信号に対してより効率的です。

包絡線スペクトル（エンベロープ）

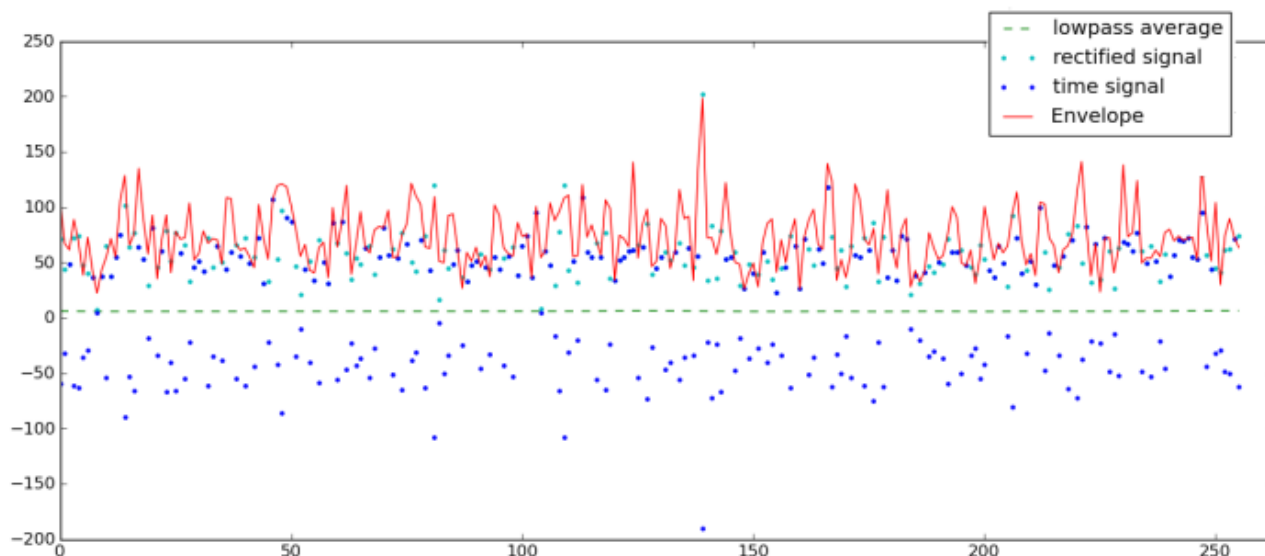
理論

クレストファクタや尖度を計測することで、わずかな出費で損傷の有無を早期発見できます。部品の分解検査には、場合によっては多大な費用がかかります。また、多数のベアリングが存在する可能性があることから、損傷したベアリングや部品をより正確に特定するための診断手段が注目されています。欠陥の特定は、接触面の損傷に起因する衝撃の評価に基づいています。回転部品の損傷の場合、衝撃パルスは周期的に発生します。その周期の長さは、欠陥が接触面に接触する頻度に依存します。この衝撃パルスの周期は、ベアリングの回転速度と要素部品の形状の両方に依存します。よって、衝撃パルスの周期から不良部品を特定できます。

衝撃パルスは、作動した機械部品の振動に起因する高周波信号成分と、重畳され、場合によっては変調された低周波成分とを含み、衝撃の周期的繰り返しに関する情報を含みます。これら信号の低周波部分は、包絡線計算で決定できます。包絡線は、周波数領域でヒルベルト変換を適用することで効率的に計算できます。TwinCAT 3 Controller Toolboxで提供されるハイパスフィルタによる事前フィルタは有効ですが、必須ではありません。包絡線の計算に続いて、包絡線信号のパワースペクトルが決定されます。この包絡線スペクトルの特徴的な周波数は、衝撃の周期を特定します。

アプリケーション

包絡線スペクトルは、特にベアリングのどのユニットまたはどの部品に欠陥があるかを診断するのに役立ちます。それに加えて、信号の重要な部分を評価し、干渉する部分を除外することが、損傷の早期発見に役立つ可能性があります。早期発見のために使用する場合、ベアリング形状から問題の損傷頻度を決定し、監視する必要があります。



上図は、すでに使用され破損したローラーベアリングの信号の包絡線を示しています。時間信号は青い点で、包絡線は赤い線で示されています。より理解しやすくするために、緑の点線は時間信号の移動平均を示しており、これは正確にはゼロではありません。水色の線は時間信号における負の値の大きさを示しています。包絡線は常に、時間信号の最大値または時間値の大きさに非常に近いところにあることが分かります。時間信号のピークや負の偏向は包絡線のピークにつながりますが、時間信号の「バックグラウンドノイズ」は包絡線の形成によってほとんど変化しません。

包絡線スペクトルの解析

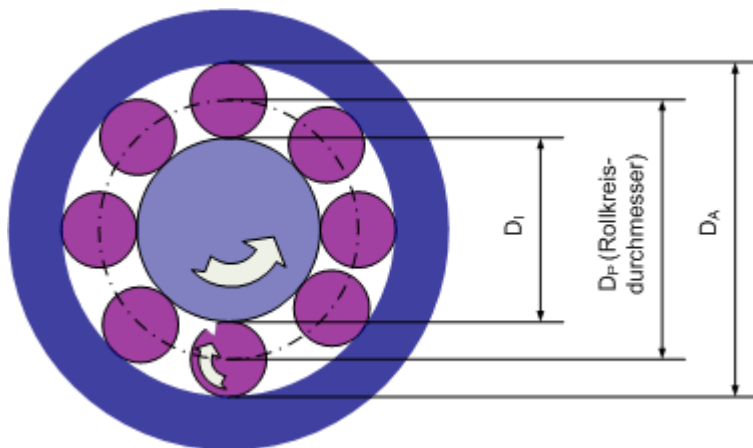
周期的な衝撃の連続（パルス列）は、多くの高調波を持つ信号に対応するため、包絡線スペクトルは、基本周波数と基本周波数の整数倍の両方を含みます。パワースペクトルやマグニチュードスペクトルと同様に、スペクトル値に関連する周波数は、結果配列のインデックスにFFTの周波数分解能を掛けたものから導き出されます。詳細は周波数分解能 [▶ 13]を参照してください。FFTの長さを N 、サンプリングレートを f_s とすると、次のようになります： $\Delta f = f_s / N$ 。そして、インデックス m を持つ周波数ビンの周波数は、 $f_m = (m-1) \Delta f$ です（配列インデックス m が1から始まると仮定しています）。

診断のためには、パルス列の基本周波数を特定する必要があります。高調波は、均等な間隔で並んだ鋭い最大値のコーム状の列によって認識できます。基本周波数はこれらの最大値の間の距離であり、通常はこの系列の最初の最大値の周波数です。その逆数が衝撃の周期となり、逆数の単位は時間差となります。計測すべき軸の回転速度と、ベアリングの形状から求めることができる損傷周波数の速度比とともに、損傷が発生した可能性のある部品を特定できます。

ローラーベアリングに特徴的な損傷周波数

下図は、単純なローラーベアリングで起こりうる速度比の例です。原理的には、衝撃パルスは、2つのベアリング要素の接触点が損傷面を通過する周波数（図の一番下の回転体上側）で発生します。この接点も、要素同士の相対的な動きによって移動します。接触点の回転速度または角速度は、正常に機能するベアリングではほとんど滑りが発生しないという原則に基づいて求めることができます。つまり、ベアリングの要素同士がほぼ完全に転がり合うことを意味します。

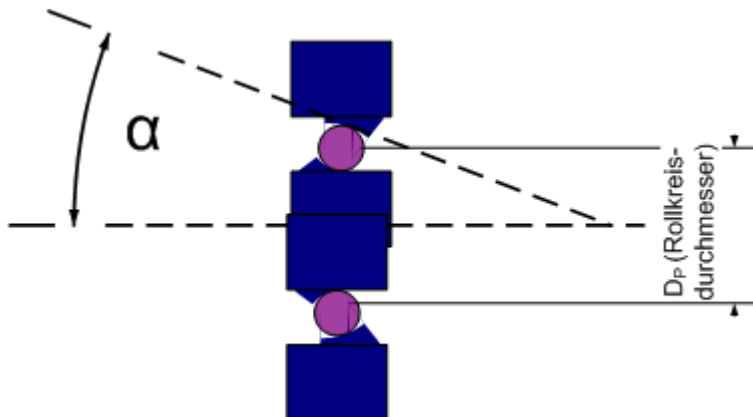
ローラーベアリング形状のパラメータ



内輪に連結された車軸の速度 f_{red} が計測され、ベアリング部品の直径が次のとおりであるとして：内輪の直径 D_I 、玉の直径 D_B 、外輪の直径 D_A 。ボールの数は Z です。 D_I と D_A はピッチ径を導きます： $D_P = (D_I + D_A) / 2$ 。ここで、内輪が f_{rot} の速度で回転する場合、パルス周波数はここから特定できます。周波数の呼称としては、以下の略語が一般的です：

- **BPFO**：転動体通過周波数（外輪）
- **BPFI**：転動体通過周波数（内輪）
- **BSF**：ベアリング回転周波数（転動体）
- **BPF**：ボール通過周波数
- **FTF**：基本周波数

接触角：



軸方向荷重を受けるベアリングの場合、正確な計算のためには、ボールが走行面に接触する接触角 α を用いてボールの直径を補正する必要があります： $D_b = \cos(\alpha) D_B$ 。ラジアルベアリングの場合、この角度は 0° であり、その結果、実際に使用される式は以下のとおりです：

$$\begin{aligned} \text{BPFO} &= Z * f_{red} / 2 * (1 - D_b / D_P) \\ \text{BPFI} &= Z * f_{red} / 2 * (1 + D_b / D_P) \\ \text{BSF} &= f_{red} / 2 * D_P / D_B * (1 - (D_b / D_P)^2) \\ \text{BPF} &= 2 * \text{BSF} \end{aligned}$$

内輪が回転している場合：

$$\text{FTF} = f_{red} / 2 * (1 - D_b / D_P)$$

外輪が回転している場合：

$\text{FTF} = f_{red} / 2 * (1 + D_b / D_P)$ ここで、 $(\text{BPFI} + \text{BPFO}) / f_{red}$ は常に転動体の数 Z に等しくなります。例えば、接触角 α は荷重によって変化する可能性があるため、実際にはこれらの式から若干のずれが生じます。単純な経験則では

$$f_{\text{BPFI}} = 0.6 * f_{red} * Z$$

が内輪の欠陥の指標周波数として使用され、

$$f_{BPFO} = 0.4 * f_{red} * Z$$

が外輪の欠陥の指標周波数として使用されます。ベアリング形状の特定には、ベアリングメーカーのデータを参照することが有効です。また、メーカーがインターネット上で提供している計算プログラムを使用することも有効です。

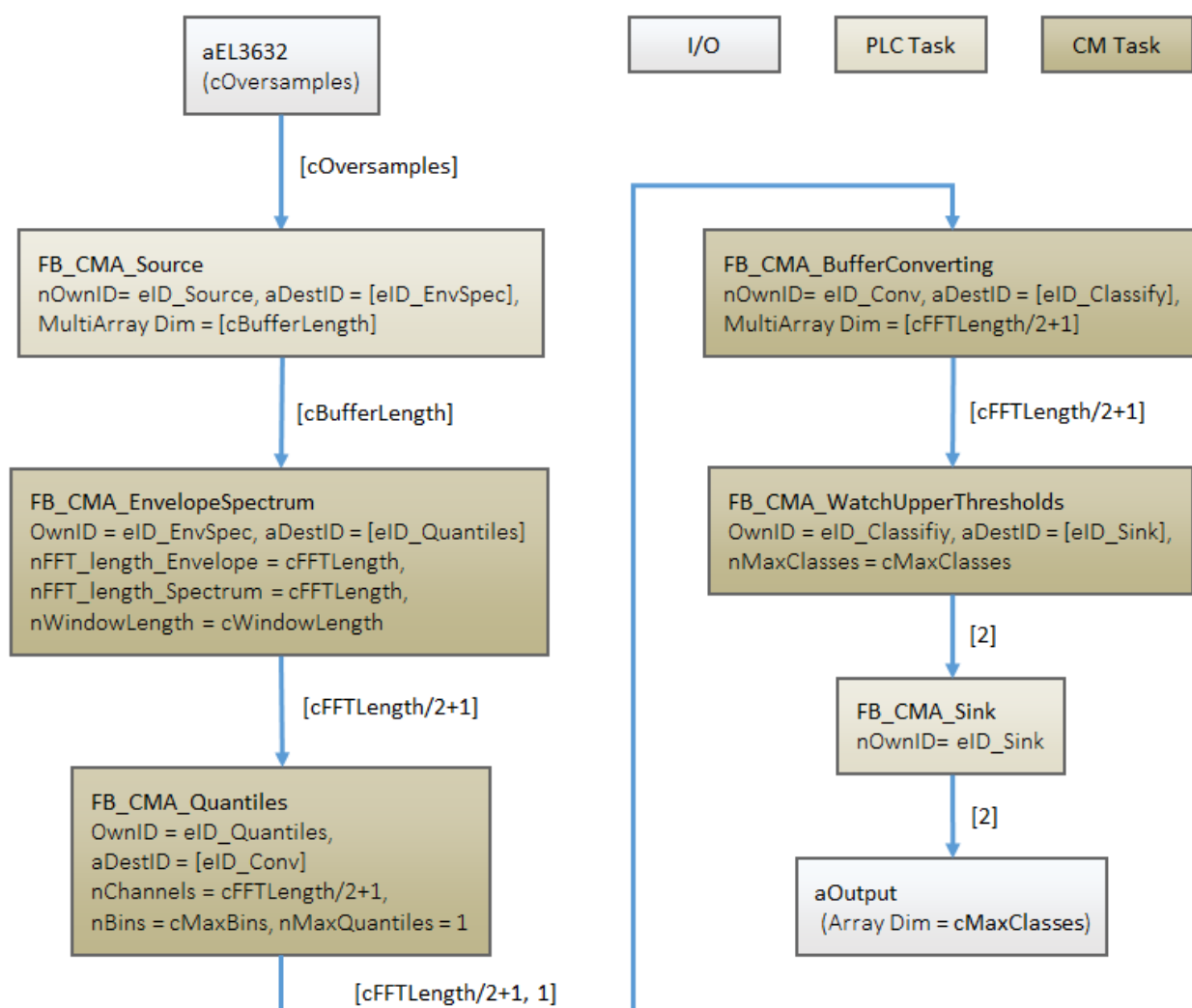


ローラーベアリングの型番からは、ベアリングの形状について明確な結論を導き出すことはできません。転動体数のようなパラメータは変更される可能性があります。

プロセスの概念

周波数解析処理ステップ

解析のステップ：



上図は、包絡線スペクトルの処理ステップと、ここで使用できるファンクションブロックを示しています。まず最初に、包絡線 [▶ 137] ファンクションブロックを使って包絡線を計算します。その後、時間信号のスペクトルと同じように、パワースペクトルが計算されます（パワースペクトル [▶ 189]）。得られた包絡線スペクトルは、非定常信号では比較的強く変動するため、周波数解析 [▶ 34] の項で前述したように、分位点計算法（分位数）を用いて統計的に評価することが推奨されます。得られた値は、WatchUpperThreshold [▶ 246] ファンクションブロックを使用した制限値監視によって、特定の閾値内であるかどうかを自動的にチェックできます。

2.2.4 ギアボックスの監視

動機

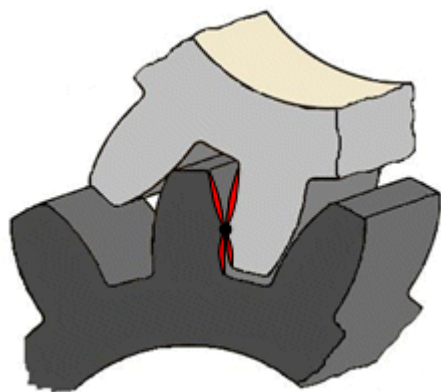
本項では、ギアボックス監視の概念について説明します。ローラーベアリングと同様に、ギアボックスも最も一般的な機械要素のひとつです。様々なドライブに使用されるため、通常、システムの信頼性において重要な役割を果たします。典型的なギアボックスの損傷は、ローラーベアリングの損傷とは異なります。これは一方で、ギアボックス内で高い応力がかかる部品が互いに直接摺動するため、潤滑と表面品質の両方に特別な要求が課されるためです。通常の運転から生じる力を吸収しなければならないため、ギアボックスは比較的大きく、したがって高価であり、メンテナンスが正しく行われていたとしても、機械の疲労寿命の間に交換が必要になる可能性があります。ここでも十分な潤滑と正しい組み立てが重要です。しかし、発生する損傷パターンは、決してこれらのエラーだけに起因するものではありません。接触点での過剰な電圧や、腐食と過熱の相互作用は、表面損傷（ピitting、マイクロピitting、剥離、摩耗）を引き起こし、最終的には歯面の欠けや変形に至る可能性があります。機械的な衝撃や過負荷は、ギアホイールの直接的な破損を引き起こす可能性があります。ローラーベアリングに比べ、ギヤユニットの欠陥は突然の故障を引き起こし、その結果多額の費用が発生する傾向があります。これは、ギアホイールの場合、最大の張力が歯底にかかるためです。その結果、疲労症状が早い段階で起こり、時間の経過とともに深い亀裂につながり、最終的には歯が折れてしまいます。後者の場合、極端な状況では、事前の警告なしにギアボックス全体が完全に停止し、例えば車軸の破損による大規模な二次損傷を引き起こす可能性があります。先に述べた原因と結果の挙動から、ギアボックスの監視には2つの目的があります：

- 第一に、摩耗の症状を長期的に監視し、傾向観察を通して問題を早期に認識し、損傷が生じる前に迅速に修正することが重要です。
- 第二に、監視によって急性の損傷を即座に認識できるため、早期に修理対策を開始でき、故障やダウンタイムを軽減できます。

理論

ギアボックス損傷の早期発見の理論的背景を以下に説明します。

噛み合い振動

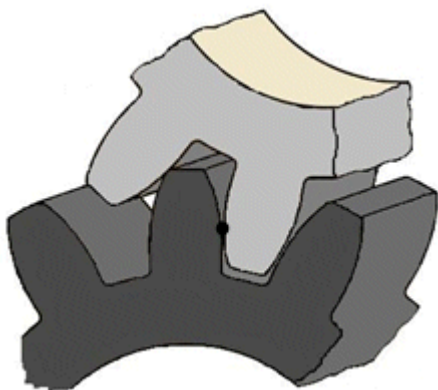


ギアボックスでは、歯車が互いに転がりながら動作し、その過程で個々の歯が周期的に接触し、力を伝達した後、再び分離します。新しく設計が優れたギアボックス（インボリュート歯形）では、これが正確に一定の変速比とほぼ一定の力で行われることは可能ですが、この転がり運動が滑り運動を全く伴わずに行われることは不可能です。上の図が示すように、歯面の中心部では主に回転運動が起こり、中心部からの距離が長くなるにつれてスライド運動の部分が大きくなります。また、このような歯形では、速度比はほぼ一定ですが、伝達トルクは変化します。歯車の歯は硬く弾性のある材料で作られているため、わずかに変形し、噛み合う周期に応じて振動を引き起こします。この振動がいわゆる噛み合い周波数です。

噛み合い周波数の高調波

噛み合い振動は、正弦波のような波形を持つものではなく、比較的急激に力が発生して減衰することに基づく強制振動であるため、そのスペクトルには噛み合い周波数の整数倍の周波数を持つ多数の高調波が含まれます。トルクが歯を弾性的に変形させるため、振動はギアホイールにかかる負荷に依存します。したがって、ギアホイールの振動は負荷に依存します。

摩耗の結果



摩耗が進むにつれて、歯形は理想的な形状からずれていきます。これは、歯面が互いに摺動することによって素材が摩耗するためです。この現象は、上図が示すように、歯面の中心から離れれば離れるほど顕著になります。そのため、摺動運動そのものが大きくなり、トルクがより強く変化するため、噛み合い振動、特に高調波が増幅されます。したがって、高調波の解析は、ギアボックスの状態を評価する鍵となります。明らかに損傷しているギアボックスの高調波が突然減少した場合は、アラーム信号として受け取ってください：歯面の破損が進行し、歯形の弾性が増加している可能性があります。この場合、ギアボックスが完全に故障していることが即座に推測されます。

ケプストラム

ケプストラムは、ギアボックスの高調波、変調、および振動を解析するための最も重要なツールです。これは、信号スペクトルの周期性を強調する操作です。

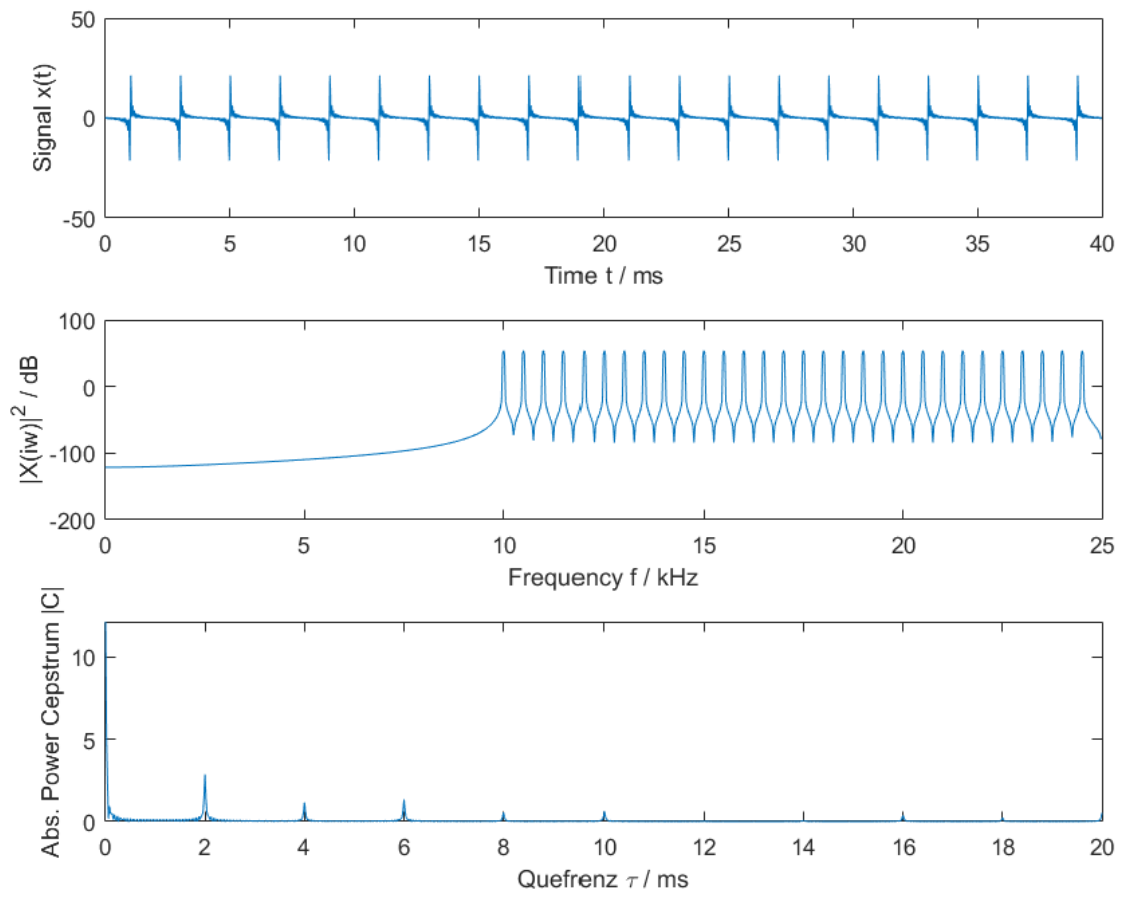
信号 $x(t)$ のパワーケプストラムは次のように定義されます：

$$C_p(\tau) = X^{-1} \{ \log(|X(\omega)|^2) \} = X^{-1} \{ 2 \log |X(\omega)| \}, \quad \text{Quefrency } \tau$$

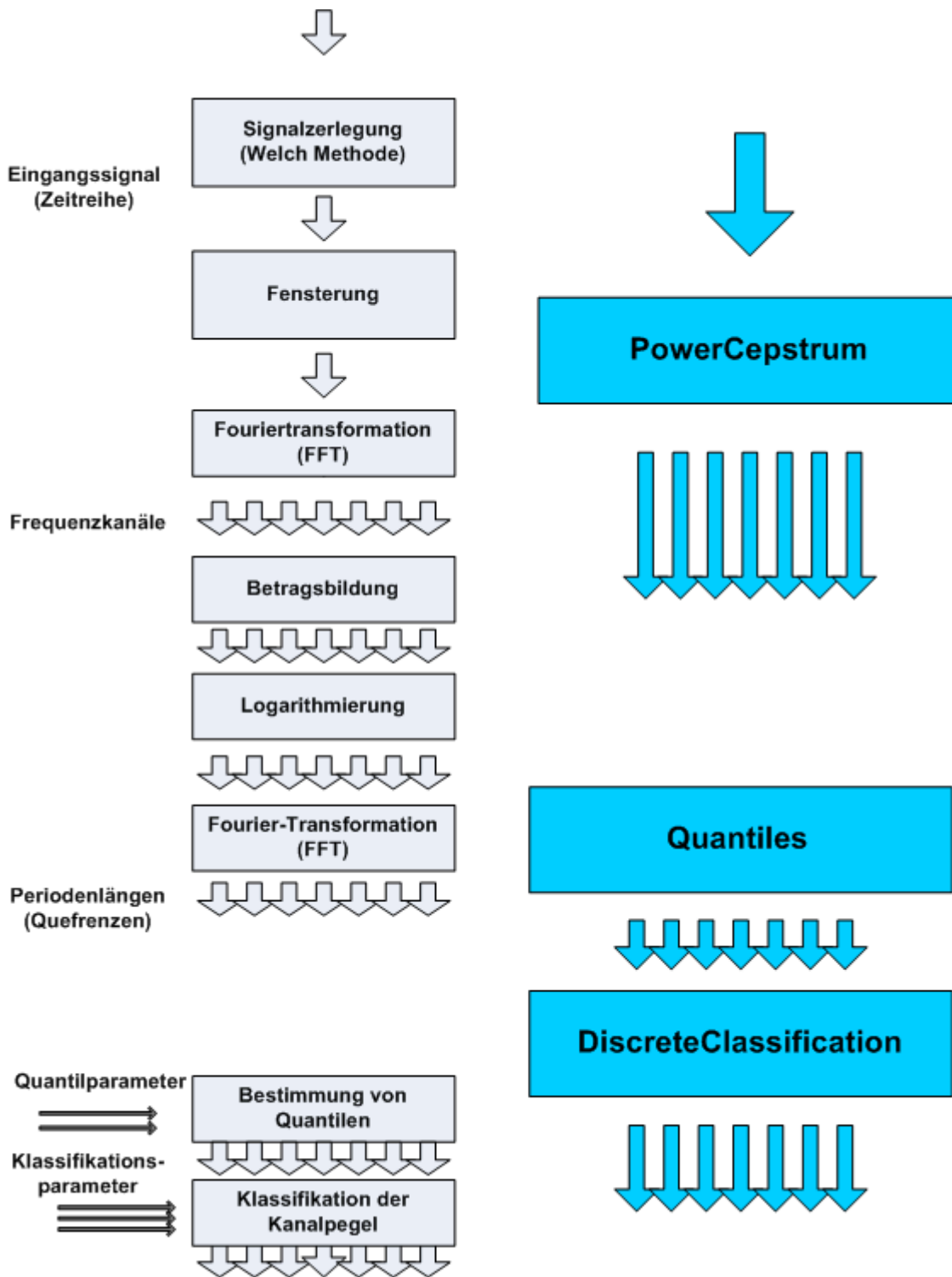
解釈

フーリエ解析が信号の時間領域における周期性を示すのに対し、ケプストラムは周波数領域における周期性を示します。逆フーリエ変換は、その結果を時間領域にマッピングします。しかし、関連する値インデックスは、 t の元の時間軸を表すのではなく、発生したスペクトル期間を示します。このパラメータは時間単位を持ち、反転と逆変換の組み合わせを示すため、ケフレンシー（時間的スケール）と呼ばれます。例えば、調和、フィルタリング、位相解析などのエンティティや操作に対しても、類似した区別的な名称があります。採用する2つのフーリエ変換の長さ N が長ければ長いほど、ケプストラムの計算に参照される入力値が多くなり、ノイズや（非系統的な）揺らぎの影響を低減できます。時間分解能を大きくするには、サンプリングレートを上げる必要があります。

例として、以下の図は、いわゆる調和音複合体のパワースペクトルとパワーケプストラムを示しています。信号の時間領域は、2msごとに繰り返されるパルスを示しています。個々のパルスは重畳高調波で構成されています。この状況は前述のギアユニット破損のケースと同様（粗いモデル）であることを意味します。中央の図はパワースペクトルを示しています。パワースペクトルの周期性がはっきり確認できます。最大値は0.5kHz間隔です。下の図はパワーケプストラムの大きさを示しています。最大の（全体的な）最大値は0msですが、実際には特に意味はありません（単にパワースペクトルの平均値を示しているだけです）。この最大値を除いて、最も大きな最大値は2msで見られます。これは、時間信号の時間的繰り返しに正確に対応しており、またパワースペクトルの局所的な最大値の距離の逆数である $1/0.5 \text{ kHz} = 2 \text{ ms}$ にも相当します。



プロセスの概念（計算手順）



パワーケプストラムの計算

ケプストラムの計算は、定義から導かれるように、通常の周波数解析に基づいています。したがって、セクションデータストリームの解析 [▶ 16] で説明したように、最初に信号をセクションに分割し、その後、ウィンドウ処理と呼ばれるウィンドウ関数の乗算を行う必要があります。パワーケプストラムは、フーリエ変換、絶対値計算、対数計算、さらにフーリエ変換という、前述の計算手順に基づいて計算します。ここで重要なのは、ゼロによる除算と同様、ゼロの対数が定義されていないため、値の範囲を超えないようにすることです。

最初の計算結果は複素数値を持ちます。通常、マグニチュードまたはマグニチュードの2乗がさらなる解析に使用されます。

サンプルはこちらからダウンロードできます：[パワーストラム \[▶ 346\]](#)

分位の計算

通常、ケプストラムの短期的な値は、その元となるFFTの値のように非常に強く変動します。したがって、次に推奨される処理ステップは、得られた各周期、つまり各ケフレンシーに対して分位数を計算することです。例えば監視タスクの場合、95%の分位数が決定されることが多いです。これは、全てのケースの95%で計測値が超過しない値です。この計算は、[分位数 \[▶ 193\]](#)ファンクションブロックを使った周波数解析と同様に行われます。

閾値の監視

さらなる処理は目的により異なります：

- 傾向解析には、長期間にわたり得られた値を保存し、その推移を表示することが有用です。
- 機械の自動監視には、閾値や制限値を設定して分類することが有効です。これは、[DiscreteClassification](#)ファンクションブロック [\[▶ 113\]](#)に記載されたファンクションブロックで実施します。
- 機械の保護など、解析範囲が限られているタスクには、[WatchUpperThreshold](#)ファンクションブロック [\[▶ 246\]](#)を使用します。これは、最高限度カテゴリの番号を自動計算するものです。例えば、状態「全て正常」をカテゴリ0、状態「警告」をカテゴリ1、状態「アラーム」をカテゴリ2に割り当てた場合、出力がレベル1であれば警告をテキストメッセージで送信することができ、出力がレベル2であれば設備を自動的に停止させることができます。

2.2.5 疲労寿命解析と損傷計算

動機

風力発電機の塔、橋梁のグライダー、荷重クレーン、または機械部品など、周期的な荷重にさらされる構造部品の疲労寿命はどのくらいか。

計算疲労分析は、こうした疑問に対する答えを与えてくれます。構造物の疲労強度を決定するには、使用荷重、形状、材料データを使用します。部品や構造物が稼働中であれば、実稼働荷重の継続的な監視を行うことで、疲労寿命の推定を行うことができます。

軽量構造の目標は、何よりも、生産と稼働における原材料とエネルギーの省資源化です。軽量化はコスト削減を可能にするだけでなく、より複雑な形状を持つ新製品も実現します。こうした目標は当然、軽量構造の分野に限定されるものではなく、機械設計のあらゆる分野に広がっています。以前は寸法オーバーだった部品は、運転時に想定される荷重に合わせて調整されます。したがって、荷重に対する安全マージンは減少し、部品は疲労耐性設計ではなくなります。逆に、疲労最適化設計とは、定義された目的と荷重プロファイルで使用した場合に、故障することなく望ましい疲労寿命を達成するように部品を設計するものです。

[サンプル \[▶ 307\]](#)セクションの疲労分析 [\[▶ 351\]](#)エリアに示すように、疲労寿命解析用のソフトウェアは、TwinCAT 3 Condition Monitoringで簡単に実装できます。さらに、ベッコフシステムは、必要な計測変数を記録し（例：ELM35xx）、それらをEtherCATを介して長距離でも通信し、信頼性の高いIPCで処理し、最終的にそれらを上位システムに転送して保存または表示するための優れたプラットフォームを提供します。

以下の項は、背景知識を提供し、対象分野の分類に役立ちます。対象のファンクションブロックの説明およびサンプルは、実用的な実装例として役立ちます。

疲労寿命解析の背景

本項の学習内容：

- 部品の疲労寿命の計算方法および損傷累積
- レインフロー法とは
- ウェラー曲線とは
- ウェラー曲線はどのように決定されるか

基本の概念

疲労計算が開始されたのは、現在のようにシミュレーションツールや計測技術が標準的に使用されるようになるはるか以前の約200年前のことでした：クラウスタールのエンジニアであったウィルヘルム・アルバートは、1837年の時点で、鉄製チェーンの疲労に関する最初の調査を行っていました。彼の観察によれば、チェーンが破損するのは過負荷だけでなく、より低い振幅での頻繁な繰返し荷重によるものでもあり、疲労強度を系統的に調査するための基礎を築きました。1860年頃、ドイツの鉄道技師であったアルブレヒト・ウェラーは、金属の疲労強度に関する数式、いわゆる**ウェラー曲線（S-N曲線、応力-寿命曲線と呼ばれることもある）**を定式化しました。このために、彼は材料サンプルに繰返し荷重をかけ、破壊するまで試験を実施しました。これらの計測結果から、アウグスト・ウェラーは破壊までのサイクル数に対する機械的応力を図に示し、被試験材の特性曲線を導き出すことに成功しました。

これらの基本的な観察に加え、ガーバーとグッドマンは、一定の機械的予荷重など、平均応力が疲労寿命に及ぼす影響を詳細にテストしました。彼らは計算に平均応力効果を考慮する計算方法を開発しました。

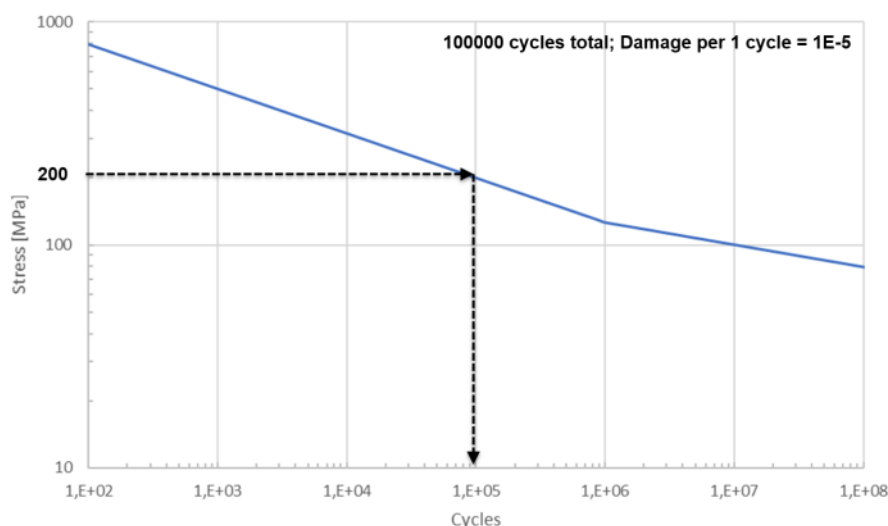
線形**損傷累積**の仮説は、1924年にA.パルムグレンが提唱し、1945年にM.A.マイナーにより出版されました。これは、検査対象の部品が全損傷 $D = 1$ で故障することを記述しています。

$$D = \sum_{i=1}^k \frac{n_i}{N_i}$$

N_i は、部品が破損するまでの、あるサイクル応力範囲における最大負荷サイクル数を表し、 n_i は、部品が既に受けた負荷サイクル数を表します。 n_i/N_i の比は、部分損傷と呼ばれ、一定の応力範囲による損傷のみを考慮します。そして、発生した全ての応力範囲にわたる全ての部分的損傷の合計が、全損傷 D となります。それに応じて、 i インデックスは k 個のクラスにまたがり、それぞれが応力範囲を定義しています。

平均応力と応力範囲を定義します：ある部品に一定の振幅 y_p の高調波信号で周期的な応力をかけると、各サイクルの応力範囲は $2 \cdot y_p$ に等しく、平均応力は $y_m = 0$ になります。この振動に対して、追加の平均応力 y_m （または初期応力）を加えると、結果として生じる応力範囲は変わりませんが、各サイクルの平均応力は y_m に変化します。平均応力の方向（引張または圧縮）と振幅によって、部品の疲労寿命は大きく変わります。ここで、平均応力補正を使用し、サイクル応力範囲を平均応力ゼロの損傷相当サイクルに補正します。

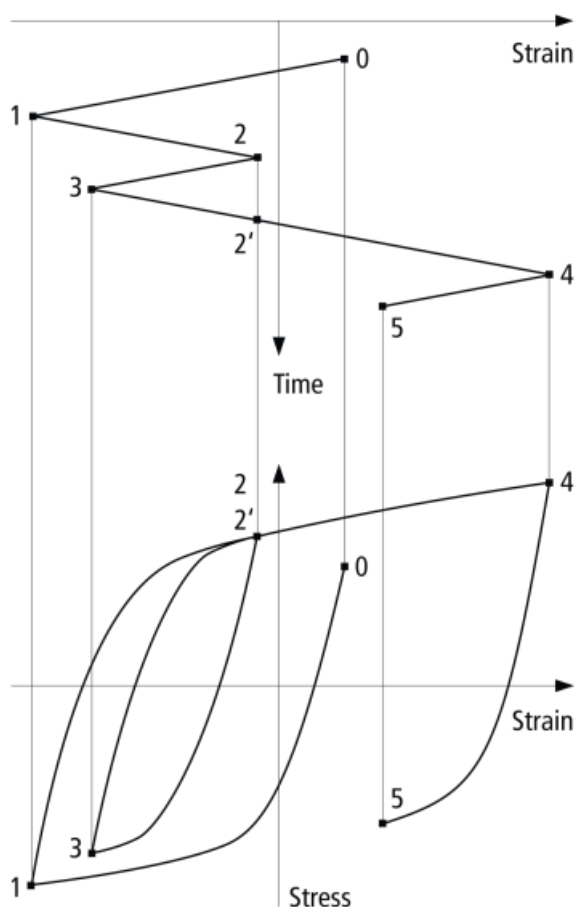
以下の例では、ある部品に200MPaの応力範囲で応力を加えています。この応力に対して、ウェラー曲線は破断までのサイクル数を100,000（1E5）サイクルと定義しています。こうして、1回のサイクルで疲労寿命の1/100,000が消費されます。ウェラー曲線は両対数で示されます。



材料試験とは対照的に、実際の部品にかかる応力はほぼランダムな信号曲線を持っています。そのため、適切な方法で時系列の計測信号をクラス別にビン化し、それを上記のマイナー法にしたがって損傷領域に変換して累積する必要があります。

松石正典と 遠藤達雄は1968年に、時間軸を90度回転させた時間軸上に時間信号を表す方法である**レインフロ－法**を発表しました。この方法は、逆転点で雨水が屋根から流れ落ちるパゴダの屋根に似ていることから、その名前が付けられました。まず、引っ張り方向などのハーフサイクルがカウントされ、反対の応力方

向、例えば押し方向のハーフサイクルと組み合わせられ、閉サイクルが形成されます。この方法はASTM E 1049-85（疲労分析におけるサイクル計数の標準的实施方法）で標準化されており、広く使用されています。



ウェラー曲線の決定と意味

ウェラー曲線を決定するために、一定の試験片形状（通常は直径10mmの円形試験片）を持つ材料試験片を試験機に固定します。試験機は、試験片に対して周期的な力を加えます。この力は通常、ゼロ値（平均応力なし）を中心に対称的に振動し、一定の最大力を持ちます。しかし、他の平均応力値を持つ材料パラメータを求めるために、周期的な圧縮または引張範囲に変化を加えることも可能です。試験は一定の最大振幅で実施され、破損（＝疲労破壊）または技術的な初期亀裂が生じるまで行われます。この負荷段階で到達したサイクル数が図表に記録されます。

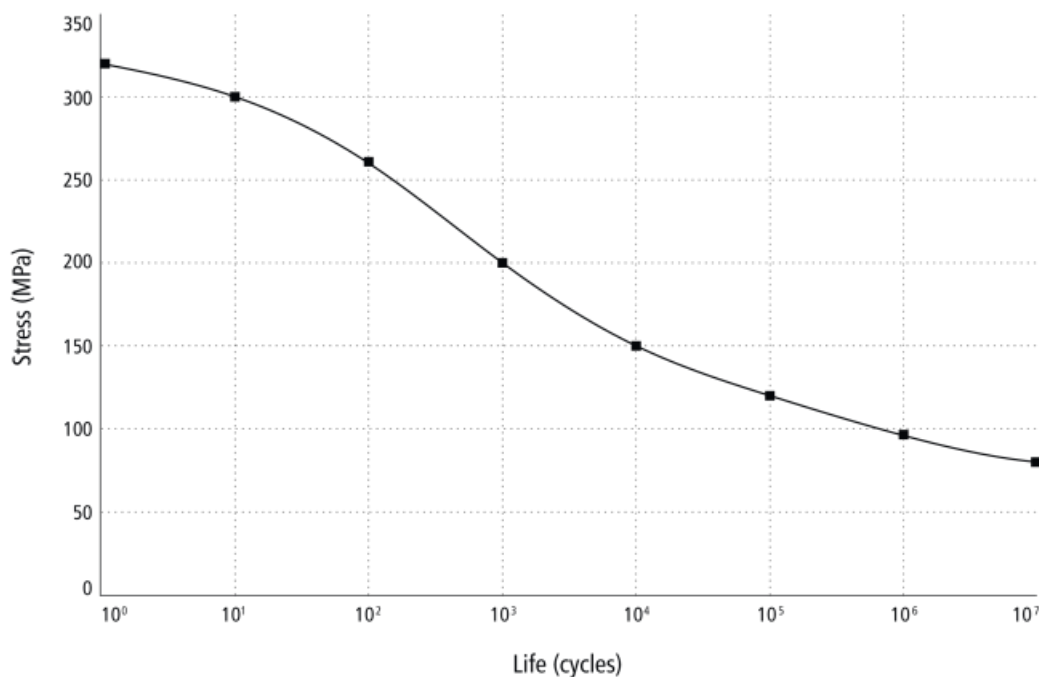
材料は外部応力、すなわち加えられた力 F に対して、その断面積 A に基づいて反応します。この応力は機械的応力として定義されます。既知の（初期の）試験片直径と既知の作用力が分かっている場合、結果として生じる部材応力は F と A の比として求めることができます。単位は通常メガパスカル（MPa）です。

したがって、機械的応力値は、達成可能なサイクル数に対してウェラー曲線に示されます。試験結果のばらつきを特定するために、同等の材料および同じ試料形状を用いて、負荷段階ごとに複数回の反復試験を実施します。

疲労強度試験の実施は、主にDIN 50100などで標準化されています。また、ASTMでは、ASTM E466-15で一定振幅による疲労試験の標準手順を公表しています。

例として、アルミニウムの単純な対数表現によるウェラー曲線を以下に示します。

S-N Curve for Brittle Aluminum with a UTS of 320 MPa



ウェラー曲線は3つの領域に分類されます：

低サイクル疲労（LCF）：応力振幅の大きい領域

($N = 1 \dots \text{約} 10^4$ サイクル)。下の図では黄色で示されています。

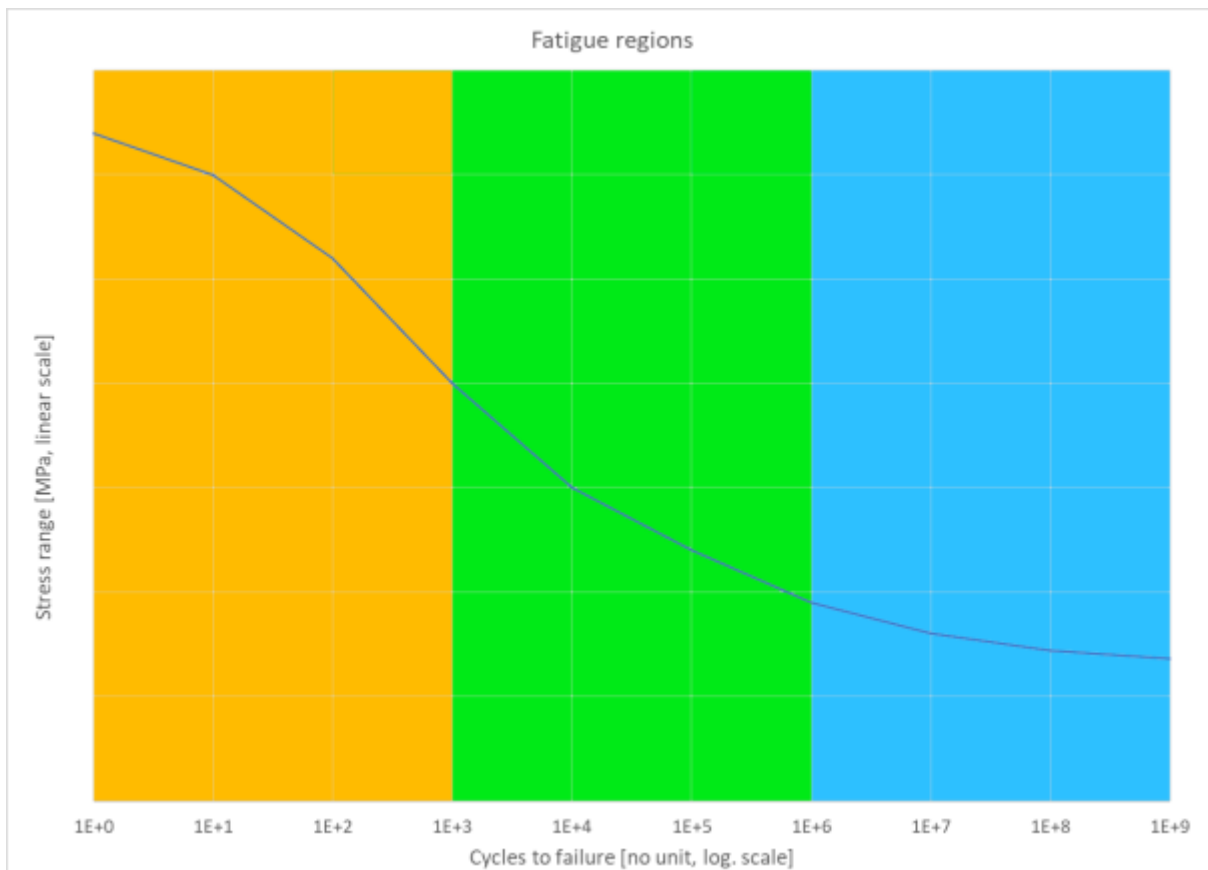
- この部分で高い応力が発生し、材料に強い塑性変形が生じます。この領域では、ひずみ制御の材料試験が挙動をよりよく示しているため、いわゆるひずみウェラー線が使用されます。
- 実際には、低サイクル疲労領域用に設計された部品はわずかです。

高サイクル疲労（HCF）：ウェラー曲線をx軸とy軸に対数示したときの一定の傾きの面積（ $[N = \text{約} 10^4 \dots 10^6]$ ）。下の図では緑色で示されています。

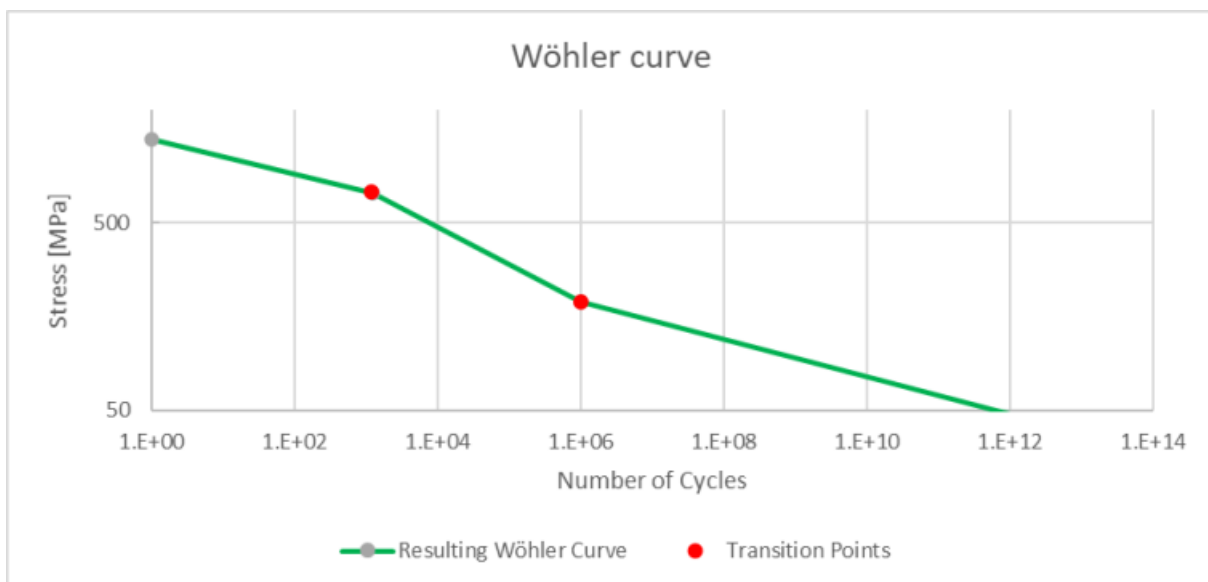
- 動作信頼性の高い部品設計の定番の領域です。
- 産業界や研究機関、多くの用途で広く使用されています。
- 多くの材料パラメータが文献に掲載されています。

超高サイクル疲労（VHCF）：ある振幅 $[N > \text{約} 10^6]$ からウェラー曲線が著しく低下しない領域。下の図では青で示されています。

- 疲労が観察されない理論的な応力限界（ただし、腐食や高温などにより疲労の兆候が現れることもある）。



上記の3つの領域に基づいて、ウェラー曲線の使用可能な近似値を以下に示します。このため、ウェラー曲線を両対数表現に変換し、3つの線形セクションに細分化します。この表現は頻繁に使用され、わずか数個の材料パラメータから作成できます。

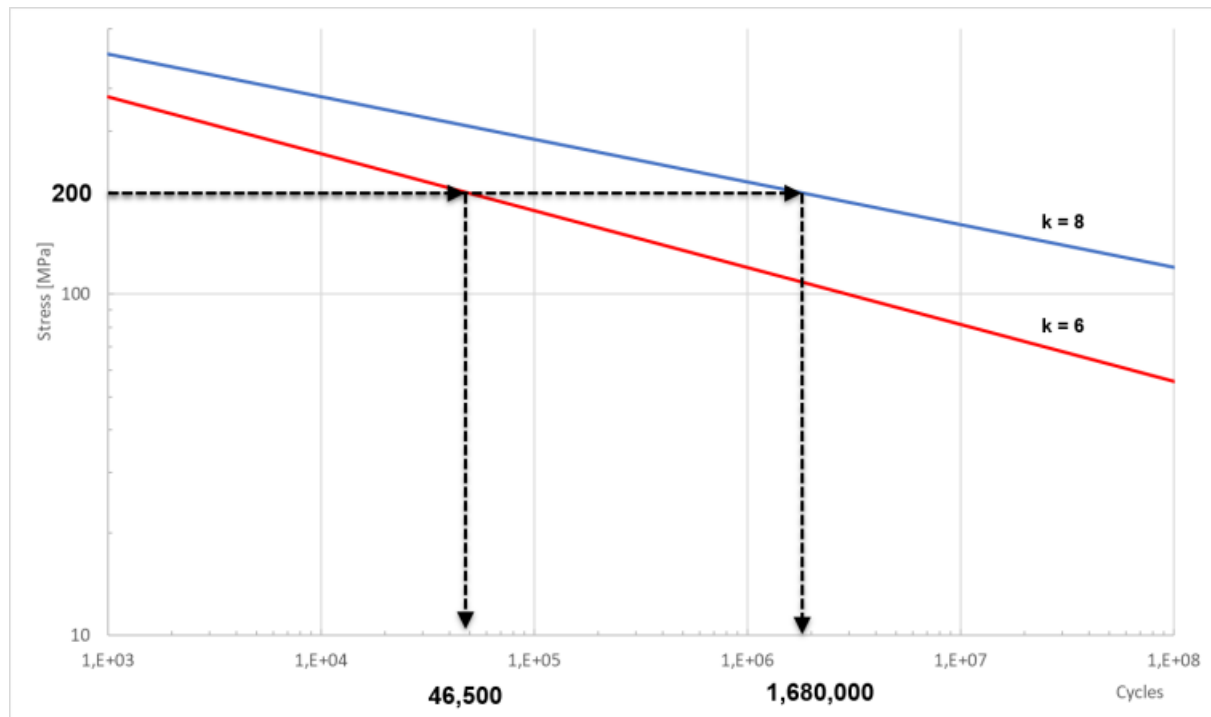


ウェラー曲線の傾き k の一般的な値は以下の通りです：

- 溶接部品： $k = 3$
- 電子部品: $k = 4$
- 応力集中のない部品： $k = \sim 10$

ウェラー曲線の勾配が平坦であればあるほど（ k 値が高ければ高いほど）、疲労挙動において材料の感度が低くなります。より小さい応力範囲は材料への損傷を少なくします。つまり、一定の応力範囲までの破壊に耐えられるサイクル数は、ウェラー曲線の傾きが急な材料よりも多くなります。

溶接構造、応力集中のある部品、電子部品など、応力範囲が小さくても疲労の影響を受けやすい部品には、勾配が急な方が適しています。以下の例では、傾き $k = 8$ と $k = 6$ の2つのウェラー曲線を、同じ応力交点で示しています。故障までの許容サイクル数は大きく異なります ($46,500 \sim 1.68 \times 10^6$)。



ウェラー曲線の解釈に関する実践メモ

材料疲労の分野で作業する際には、計算が常に統計的な性質を持つことを念頭に置いてください。ウェラー曲線は生存確率50%以上で定義されることが多いです。これは、p50のウェラー曲線の指定された時点で、試験した試験片の50%がすでに破壊していることを意味します。より高い生存確率が必要な場合は、それに対応する生存確率の曲線を材料試験から導き出す必要があります。

材料パラメータの決定は、散乱した計測結果をいくつかの単純な曲線パラメータに抽象化するだけでなく、実際の材料サンプル成分とは大きく異なる場合があります。特に、応力集中による部品の寿命短縮が重要な役割を果たします。

形状：

- 試験片には鋭利なエッジがなく、研磨されている（切欠き、気泡などがない）
- 部品の鋭いエッジやパンチアウトは、局所的な応力の増加と亀裂の原因となる

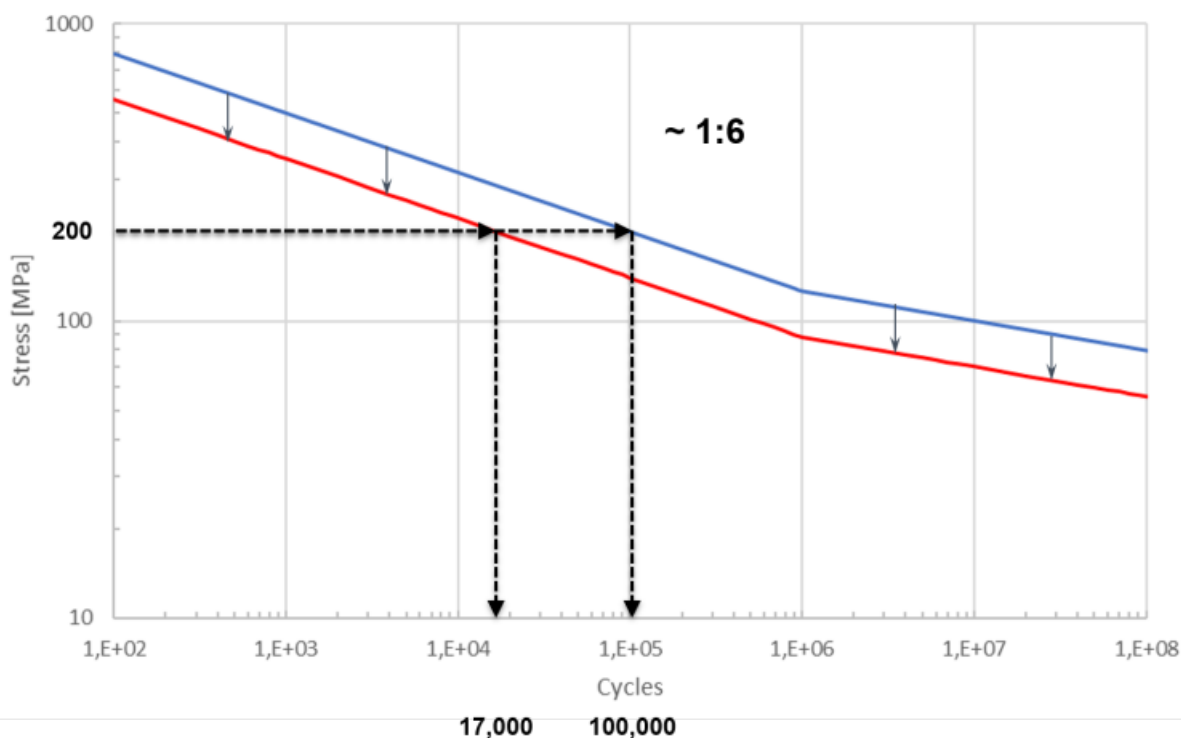
材料の品質：

- 試験片の材料品質が高いこと
- 生産される実際の材料品質は、しばしば変動する（気泡、製造工程のばらつき、酸化、電食、変形による損傷など）

機械加工：

- 実際の部品における溶接の接合部
- 機械加工では、鋭いエッジ、切り抜き、切り欠きが生じる 表面の手直しや設計変更によって、こうした応力集中を軽減できる。

応力集中は、材料の破損を著しく早めます。以下では、応力集中係数がウェラー曲線に及ぼす影響を示しています。これは、曲線の低下をもたらし、破壊までの許容サイクル数を6分の1に減少させます。



構造部品/コンポーネントの疲労強度について正確な定義が必要な場合、部品のコスト、サイズ、部品および試験設備の利用可能性に応じて、いわゆる「部品ウェラー曲線」を決定できます。ここでの欠点は、テストにかかるコストが高いことに加え、現在の部品での有意性は高いものの、設計変更があった場合に限られた領域でしか使用できないことです。

監視アプローチ：疲労分析または耐用年数解析

負荷データは、ひずみ、力、または圧力センサなどによる計測によって試験片上で直接決定されます。レインフローアルゴリズムによるサイクルカウントの後、負荷の観点から必要な全ての情報が利用可能になります。マイナーによる疲労評価に関しては、材料特性（ウェラー曲線）を参照することになります。使用する材料の疲労特性の決定は困難であることが多いです。材料特性がほとんど、あるいは全く入手できない、または、材料特性を推定しなければならないことがよくあります。

実践的には、以下の考慮が有用です：

- 同一部品の相対比較の制限
- 文献データの活用と近似したウェラー曲線への変換

部品の相対比較

特定の勾配を持つ一般的なウェラー曲線は、特に同一部品、例えば量産機械の部品の相対比較に適しています。絶対的な損傷の総和はここでは意味はありませんが、「機械部品Bは機械部品Aよりも40%損傷している」といった記述を導き出すことができ、その結果、メンテナンス箇所を特定できます。

文献データの活用

絶対損傷値を求めるために、ウェラー曲線は通常、試験機内で実際の材料サンプルを用いて決定されます。しかし、ユーロコード3とFKM指令により、鋼材の材料パラメータも利用できるようになり、そこからウェラー曲線を計算することができるようになりました。



以下では、応力範囲は常に特定され、いくつかの情報源で通常見られるようなピーク振幅はありません。応力範囲はピーク振幅の2倍です。

FKM指令（機械工学研究管理委員会、VDMA所属）は、様々な種類および状態の材料に対して保守的な材料特性を規定しています。FKM指令の一例（詳細は付録 [▶ 57] を参照）。

Sort	UTS [応力範囲、MPa]	超高サイクル疲労時の応力範囲 [MPa]	k	NC	SRI [MPa]
S185	620	280	5	1E+06	4438

ファンクション `F_CM_CalculateWoehlerCurve` [▶ 270] の材料パラメータに変換すると、次のようになります：

```
fSRI : LREAL := 4438;
fUTS : LREAL := 620;
bUseUTSCorrection : BOOL := TRUE;
fK1 : LREAL := 5;
fK2 : LREAL := 0;
nNC1 : ULINT := 18790;
nNC2 : ULINT := 1E+06;
```

そのため、fSRI と fUTS を直接適用できます。勾配は規定通り 5 であり、UTS 補正から超高サイクル疲労限界への移行の間の領域を示しています。超高サイクル疲労限界は NC 用に定義され、nNC2 と入力されます。超高サイクル疲労域での勾配は、もちろん fK2 = 0 です。ファンクション `F_CM_CalculateWoehlerCurve` [▶ 270] への入力としてのみ、UTS 補正から勾配 5 の領域への遷移点を、次式によって計算する必要があります。

$$NC1 = \left(\frac{UTS}{SRI} \right)^{-k}$$

ユーロコード 3 (BS EN1993-1-9:2005) は、異なる設計の鋼材、特に圧延/押出製品、溶接/ボルト接合部、T 形鋼のような異なる形状に対してウェラー曲線を定義しています。全てのウェラー曲線には、共通して 3 つのセクションの定義があります：

- $N \leq 5 \cdot 10^6$ の場合、 $k = 3$
- $5 \cdot 10^6 \leq N \leq 10^8$ の場合、 $k = 5$
- $N \geq 10^8$ の場合、 $k = 0$ (疲労臨界ではない)

ユーロコード 3 の例 (詳細は付録 [▶ 57] を参照)。

設計の詳細	$\Delta\sigma_c$ [応力範囲、MPa]	k1	k2	N ($\Delta\sigma_c$ の場合)	NC1	ND (超高サイクル疲労限度)	SRI (N = 1 [MPa] の場合)	$\Delta\sigma_{NC1}$ (NC1 [MPa] の場合)	$\Delta\sigma$ (ND [MPa] の場合)
160	160	3	5	2E+06	5E+06	1E+08	20159	118	65

板、中空円筒または類似の形状として製造される圧延または押出製品は、ユーロコードでは設計細目 160 として定義されています。

提供された疲労分析 [▶ 351] から材料パラメータに変換すると、次のようになります：

```
fSRI : LREAL := 2580;
fUTS : LREAL := 20159;
bUseUTSCorrection : BOOL := TRUE;
fK1 : LREAL := 5;
fK2 : LREAL := 0;
nNC1 : ULINT := 5E+06;
nNC2 : ULINT := 1E+08;
```

ここでは、勾配 3、5、0 の 3 つの定義されたゾーン要件が、ファンクション `F_CM_CalculateWoehlerCurve` [▶ 270] を使用したときに満たされるように、パラメータ変換されています。最初のセクションで勾配が 3 になるように UTS 補正が使用されています。これが、fUTS が fSRI よりも大きい理由です。k₂ = 5 とした場合、fSRI の値は次のようになります。

$$fSRI = \Delta\sigma_{NC1} NC1^{(1/k_2)}$$

次にfUTSをユーロコードのSRIに設定します。その結果、最初のセクションの勾配は3になります。fk1 = k2、fk2 = 0に設定した結果、他の2つのセクションが得られます。

まとめ

疲労寿命解析の**入力値**は3つのグループに分けられます：

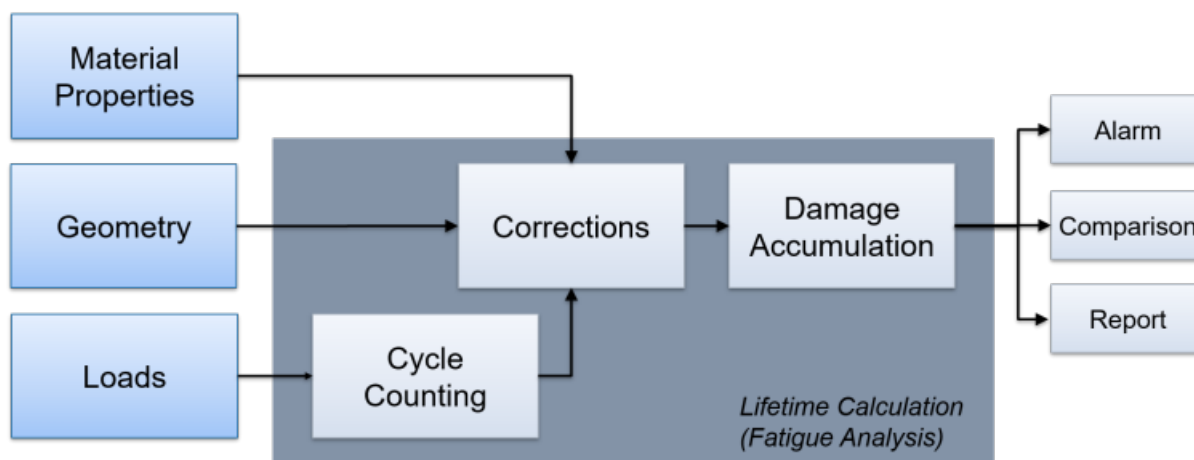
- **材料特性**は、材料の疲労強度を表すもので、通常はウェラー曲線として表され、表面のテクスチャーや加工によって補足されます。
- **幾何学的な要因**は、疲労寿命に影響する局所的な幾何学的特性（切り欠きや鋭いエッジなど）に材料パラメータを適合させるために使用します。
- **荷重**とは、部品に作用し、応力を引き起こす信号です。ストレインゲージや力変換器、トルク変換器で計測されることが多いです。

解析部分には、複雑度に応じて、以下のコンポーネントが含まれます：

- 材料パラメータの**補正**と幾何学的要因の使用
- 時間信号を損傷累積処理用のクラス（平均応力、応力範囲）に変換する際の**サイクルカウント**
- 平均応力効果による疲労寿命の低下を考慮した**平均応力補正**
- サイクルマトリックス、材料パラメータ、補正係数から個々の損傷成分を合計した**損傷累積**

その結果、損傷総和は同一コンポーネントの詳しい比較や、警告/アラームメッセージの根拠として利用できます。損傷総和が単調に増加する経過は、定められた限界までを外挿するのに適しており、致命的な応力を受けている部品を特定し、より適切なメンテナンス計画を立てることができます。

この方法は、絶対損傷数を計算するための古典的な疲労分析に適しているだけでなく、損傷強度の相対比較にも適しています。力または圧力信号、温度サイクル、またはその他の振動信号形式を使用して、静的手法（極値、クレストファクタ、尖度など）を有意義な情報によって拡張することができます。



上図の中心的な3要素である「サイクルカウント」、「補正」、「損傷累積」は、TwinCAT 3 Condition MonitoringのファンクションFB_CMA_RainflowCounting [▶ 198]、FB_CMA_MeanStressCorrection [▶ 165]、FB_CMA_MinersRule [▶ 168] によって実装されています。

付録：ユーロコードとFKM指令

ユーロコード3(EN 1993-1-9:2005 - 鋼構造の設計)

設計の詳細	$\Delta\sigma_c$ [応力範囲, MPa]	k1	k2	N ($\Delta\sigma_c$ の場合)	NC1	ND (超高サイクル疲労限度)	SRI (N = 1 [MPa] の場合)	$\Delta\sigma_{NC1}$ (NC1 [MPa] の場合)	$\Delta\sigma_l$ (ND [MPa] の場合)
160	160	3	5	2E+06	5E+06	1E+08	20159	118	65
140	140	3	5	2E+06	5E+06	1E+08	17639	103	57

設計の詳細	$\Delta\sigma_c$ [応力範囲、MPa]	k1	k2	N ($\Delta\sigma_c$ の場合)	NC1	ND (超高サイクル疲労限度)	SRI (N = 1 [MPa] の場合)	$\Delta\sigma_{NC1}$ (NC1 [MPa] の場合)	$\Delta\sigma_I$ (ND [MPa] の場合)
125	125	3	5	2E+06	5E+06	1E+08	15749	92	51
112	112	3	5	2E+06	5E+06	1E+08	14111	83	45
100	100	3	5	2E+06	5E+06	1E+08	12599	74	40
90	90	3	5	2E+06	5E+06	1E+08	11339	66	36
80	80	3	5	2E+06	5E+06	1E+08	10079	59	32
71	71	3	5	2E+06	5E+06	1E+08	8945	52	29
63	63	3	5	2E+06	5E+06	1E+08	7938	46	25
56	56	3	5	2E+06	5E+06	1E+08	7056	41	23
50	50	3	5	2E+06	5E+06	1E+08	6300	37	20
45	45	3	5	2E+06	5E+06	1E+08	5670	33	18
40	40	3	5	2E+06	5E+06	1E+08	5040	29	16
36	36	3	5	2E+06	5E+06	1E+08	4536	27	15

FKM指令

FKM指令は、様々な種類と条件の材料について、保守的な材料特性を規定しています。非合金構造用鋼、非合金細粒構造用鋼、熱処理可能な鋼、および鋳鉄の以下の値は、生存確率97.5%に適用されます。

非合金構造用鋼

Sort	UTS [応力範囲、MPa]	超高サイクル疲労時の応力範囲 [MPa]	k	NC	SRI [MPa]
S185	620	280	5	1E+06	4438
S235	720	320	5	1E+06	5072
S275	860	390	5	1E+06	6181
S355	1020	460	5	1E+06	7291
S450	1100	500	5	1E+06	7924
E295	980	440	5	1E+06	6974
E335	1180	530	5	1E+06	8400
E360	1380	620	5	1E+06	9826

焼き戻し鋼、焼き戻し状態

Sort	UTS [応力範囲、MPa]	超高サイクル疲労時の応力範囲 [MPa]	k	NC	SRI [MPa]
C22 E/R	1000	450	5	1E+06	7132
C35 E/R/-	1260	570	5	1E+06	9034
C40 E/R/-	1300	590	5	1E+06	9351
C45 E/R/-	1400	630	5	1E+06	9985
C50 E/R/-	1500	680	5	1E+06	10777
C55 E/R/-	1600	720	5	1E+06	11411
C60 E/R/-	1700	770	5	1E+06	12204
28Mn6	1600	720	5	1E+06	11411
38Cr2	1600	720	5	1E+06	11411
46Cr2	1800	810	5	1E+06	12838

Sort	UTS [応力範囲、MPa]	超高サイクル疲労時の応力範囲 [MPa]	k	NC	SRI [MPa]
34Cr4	1800	810	5	1E+06	12838
37Cr4	1900	860	5	1E+06	13630
41Cr4	2000	900	5	1E+06	14264
25CrMo4	1800	810	5	1E+06	12838
34CrMo4	200	450	5	1E+06	7132
42CrMo4	2200	990	5	1E+06	15690
50CrMo4	2200	990	5	1E+06	15690
34CrNiMo8	2400	1080	5	1E+06	17117
30CrNiMo8	2500	1130	5	1E+06	17909
35NiCr6	1760	790	5	1E+06	12521
36NiCrMo16	2500	1130	5	1E+06	17909
39NiCrMo3	1960	880	5	1E+06	13947
30NiCrMo16-6	2160	970	5	1E+06	15373
51CrV4	2200	990	5	1E+06	15690

2.3 文献注記

二次文献に関する情報を以下に示します（推奨を意図したものではありません）。このリストは全てを網羅しているわけではなく、関連文献の一部を提供するものです。

Digitale Signalverarbeitung, Fourier-Analyse, Fensterung（デジタル信号処理、フーリエ解析研究、ドイツ語）

- A.V. Oppenheim, R.W. Schaffer, J.R. Buck: Zeitdiskrete Signalverarbeitung. Pearson Studium, 2004. ISBN 3-8273-7077-9
- K.-D. Kammeyer, K. Kroschel: Digitale Signalverarbeitung – Filterung und Spektralanalyse mit MATLAB-Übungen（デジタル信号処理 - MATLABを使用したフィルタリングと信号解析） Teubner, 2002. ISBN 3-519-46122-6

Discrete-Time signal processing, Fourier-analysis, windowing（離散時間信号処理、フーリエ解析、ウィンドウ処理、英語）

- A.V. Oppenheim, R.W. Schaffer, J.R. Buck : Discrete-Time Signal Processing（離散時間信号処理） Pearson Education, 2009. ISBN 987-0131988422
- J.G. Proakis, D.K. Manolakis: Digital Signal Processing（デジタル信号処理） Pearson Education, 2013. ISBN 978-0131988422

品質保証（ドイツ語）

- J. Kolerus, J. Wassermann: Zustandsüberwachung von Maschinen（機械の品質保証） Expert Verlag, 2008. ISBN: 978-3-8169-2597-2
- DIN ISO 10816, Mechanische Schwingungen - Bewertung der Schwingungen von Maschinen due to Messung an Nichtrotierende Teil (vor VDI-Richtlinie 2056)「機械的振動 - VDI指針2056以前の非回転部品に対する測定による機械の振動評価」 規格は複数の構成要素で成り立っています。
 - DIN ISO 10816-3は、15 kWを超える出力と120 U/min～15000U/minの回転数を持つ産業用機械に適用されます。
 - DIN ISO 10816-7 は、産業用の遠心ポンプに関するものです。
 - DIN ISO 10816-21 横軸回転式風力発電機およびギアボックスに関連します。

- DIN ISO 7919, Mechanische Schwingungen - Bewertung der Schwingungen von Maschinen durch Messungen an rotierenden Wellen (機械的振動 - 回転軸の計測による機械の振動評価) 規格は複数の構成要素で成り立っています。
 - DIN ISO 7919-3は、産業用機械の連結に関するものです。
 - DIN ISO 7919-2 は、1500 min⁻¹, 1800 min⁻¹, 3000 min⁻¹および 3600 min⁻¹の定格運転回転数を持つ 50 MW 以上の定置型蒸気タービンおよび発電機を対象としています。
- DIN ISO 20816-1, Mechanische Schwingungen - Messung und Bewertung der Schwingungen von Maschinen (機械的振動 - 機械の振動の測定と評価) DIN ISO 7919-1とDIN ISO 10816-1の統合
- DIN ISO 13373-1, Zustandsüberwachung und -diagnostik von Maschinen - Schwingungs-Zustandsüberwachung - Teil 1: Allgemeine Anleitungen (機械の状態監視および診断 - 振動状態監視 - 第1部: 一般的なガイドライン)
- DIN ISO 13373-2, Zustandsüberwachung und -diagnostik von Maschinen - Schwingungs-Zustandsüberwachung - Teil 2: Verarbeitung, Analyse und Darstellung von Schwingungsmesswerten (機械の状態監視および診断 - 振動状態監視 - 第2部: 振動計測値の処理、解析、および表示)
- DIN ISO 17359, Zustandsüberwachung und -diagnostik von Maschinen - Allgemeine Anleitungen (機械の状態監視および診断 - 一般的な指針)

状態監視 (英語)

- R.B. Randall: Vibration-based Condition Monitoring (振動ベースの状態監視) Wiley, 2011. ISBN: 978-0-470-7485-8
- ISO 10816, Mechanical vibration -- Evaluation of machine vibration by measurements on non-rotating parts (機械振動-非回転部品の計測による振動評価)
 - ISO 10816-3, Industrial machines with nominal power above 15 kW and nominal speeds between 120 U/min and 15000 U/min when measured in situ (公称出力が15 kWを超え、公称速度が120 U/minから15000 U/minの産業用機械、現場計測時)
 - ISO 10816-7, Rotodynamic pumps for industrial applications, including measurements on rotating shafts (工業用回転動力ポンプ、回転軸計測を含む)
 - DIN ISO 10816-21, Horizontal axis wind turbines with gearbox (ギアボックス付き水平軸風力タービン)
- ISO 7919, Mechanical vibration -- Evaluation of machine vibration by measurements on rotating shafts (機械振動-回転軸の計測による振動評価)
 - ISO 7919-3, Coupled industrial machines (連結された産業機械)
 - ISO 7919-2, Land-based steam turbines and generators in excess of 50 MW with normal operating speeds of 1 500 r/min, 1 800 r/min, 3 000 r/min and 3 600 r/min (通常運転速度が 1 500 r/min、1 800 r/min、3 000 r/min、3 600 r/min の 50 MW を超える陸上用蒸気タービンと発電機)
- ISO 13373-1, Condition monitoring and diagnostics of machines - Vibration condition monitoring - Part 1: General procedures (機械の状態監視及び診断-振動状態監視-第1部: 一般手順)
- ISO 13373-2, Condition monitoring and diagnostics of machines - Vibration condition monitoring - Part 2: Processing, analysis and presentation of vibration data (機械の状態監視及び診断-振動状態監視-第2部: 振動データの処理、解析及び表示)
- ISO 17359:2011, Condition monitoring and diagnostics of machines - General guidelines (機械の状態監視及び診断-一般指針)

3 インストール

3.1 システム要件

以下の項目では、エンジニアリング/ランタイムシステムにおけるCondition Monitoring製品の最小システム要件を説明します。Condition Monitoringのインストールは、エンジニアリングおよびランタイムシステム上で実行する必要があります。

● 機能範囲はバージョンに依存

i ドライバとPLCライブラリの最新バージョンがインストールされている場合のみ、記載された機能の全範囲が保証されます（PLC APIの項を参照）。旧バージョンを使用した場合、機能範囲が限定されます。バージョンCM3.2で拡張された機能の詳細については、[互換性 \[▶ 61\]](#) の項を参照してください。

[概要 \[▶ 10\]](#) には、製品レベルに依存するコンポーネントのリストも含まれています。

エンジニアリング

エンジニアリングシステムとは、プログラムコードの開発に使用し、コードを実行しないコンピュータを表す用語です。エンジニアリングシステムは、以下の要件を満たす必要があります：

- TwinCAT 3.1 XAE（エンジニアリングのインストール）ビルド4022.25以上*

ランタイム

ランタイムシステムとは、プログラムコードを実行する産業用PCまたは組込み型PCを表す用語です。ランタイムシステムは、以下の要件を満たす必要があります：

- TwinCAT 3.1 XAR（ランタイムのインストール）ビルド4022.25以上*
- オペレーティングシステムWin 7、Win 10、Windows Embedded Standard 7
- TC1200 PLCおよびTF360x Condition Monitoringライセンス

i 7日間のトライアルライセンスは、テスト目的で繰り返しアクティベートできます。

エンジニアリングとランタイムが同一システム上にある場合

エンジニアリングとランタイムを1つのシステムで使用する場合、以下のシステム要件を満たす必要があります：

- TwinCAT 3.1 XAE（エンジニアリングのインストール）ビルド4022.25以上*
- TC1200 PLCおよびTF360x Condition Monitoringライセンス

i 7日間のトライアルライセンスは、テスト目的で繰り返しアクティベートできます。

*:TwinCAT 3.1 ビルド 4018から、限られた機能範囲での使用が可能になりました。

3.2 互換性

TwinCAT Condition Monitoringライブラリは、長年にわたり提供されてきました。CMバージョン3.1.xは多くのアプリケーションで使用され成功を収めています。アルゴリズムとTwinCATにおける最新要件とオプションに対応するため、新しいバージョン3.2.xが利用可能です。開発にあたり、既存アプリケーションとの互換性に特に注意を払いました。そのため、バージョン3.2.xを使用する場合、ごく少数の特殊なケースにおいてのみ、既存アプリケーションを適合させる必要があります。

- InitPars構造体で使用するEnums (はexternalTypesで、ライブラリの名前空間なしでのみ使用できます。
- ユーザが実装したライブラリでは、CMEEnum の値はデータ型の修飾子なしで使用できません。

Condition Monitoringライブラリのリリースバージョン概要

TcCM 3.1.x	3.1.16	3.1.17	3.1.18	
TcCM 3.2.x				3.2.20

バージョンCM3.2 (CMセットアップ版) での変更点

以下に説明するCondition Monitoringライブラリの変更/拡張に必要なPLCライブラリおよびドライバの最小バージョンは次のとおりです：

Tc3_CM	Tc3_CM_Base	Tc3_MultiArray	TcCM.sys	TcMultiArray.sys
2.0.30.0	2.1.18.0	2.0.14.0	3.4.17.0	3.4.17.0

基本的な変更

- アルゴリズムのマルチチャンネル機能、すなわち単一インスタンスによる複数チャンネルの連続処理。これにより、PLCアプリケーションの実装が簡素化されます。
- eCM_BartlettWindow, eCM_KaiserWindow, eCM_FlatTopWindow によるウィンドウ関数の選択の拡張と、ウェルチ法を使用する際のオーバーラップを自由に設定するオプション。ウィンドウ関数 [▶ 19]の項を参照。
- エラーや情報の出力にTC3 EventLoggerを使用します。オンライン表示 [▶ 80]の項を参照。
- ほとんどのファンクションブロックの結果をTc3 Scope Viewで直接確認できるようになりました。オンライン表示 [▶ 80]の項を参照。
- 全体的なパフォーマンスの向上と数値の改善。

アプリケーション指向のアルゴリズム

- ISO 10816-3に基づき、ファンクションブロックFB_CMA_VibrationAssessment [▶ 242]を使用して振動評価 [▶ 33]し、機械状態を分類します。

既存ファンクションブロックの拡張

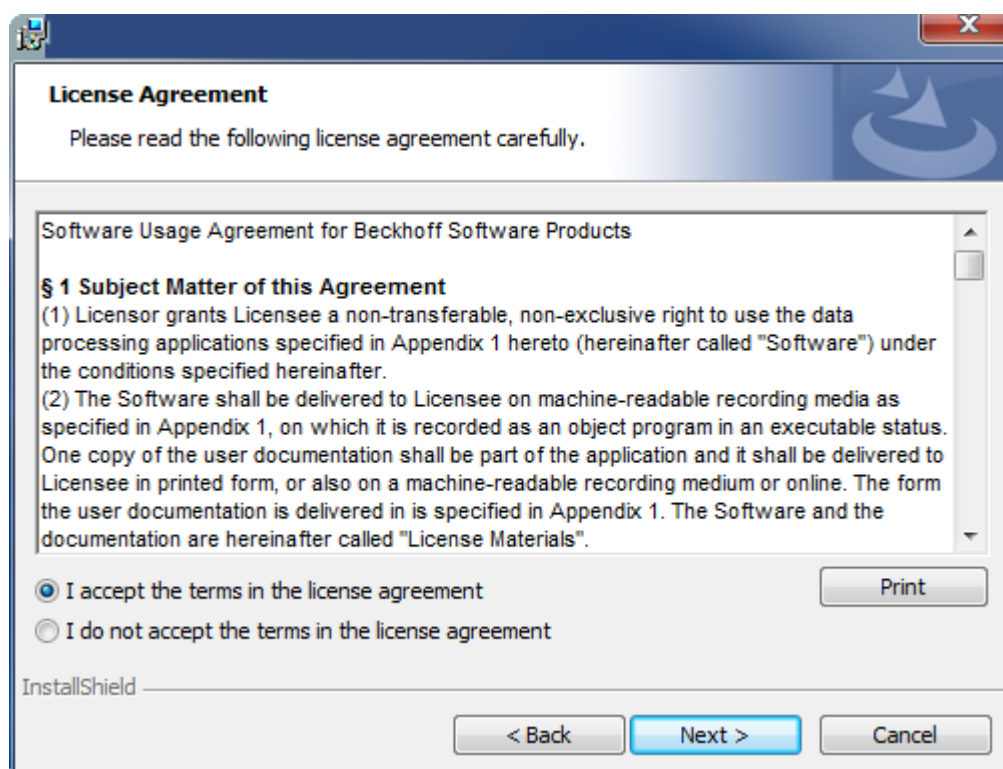
- FB_CMA_Source [▶ 230]：シングル/マルチチャンネル入力データを提供するための簡易メソッド。このファンクションブロックには、解析チェーン全体のリセットをトリガーするオプションもあります。
- FB_CMA_Sink [▶ 223]：ファンクションブロックFB_CMA_ArgSort [▶ 87]とFB_CMA_IntegratedRMS [▶ 157]に必要な3Dデータ出力メソッド、および結果出力メソッドの簡略化されたバリエーション。
- FB_CMA_IntegratedRMS [▶ 157]：設定された周波数帯域で積分RMS値を計算（オプション）。
- FB_CMA_MomentCoefficients [▶ 172]：2つのバリエーションで過剰度を計算するオプション：bExcessKurtosis := TRUEの場合、尖度は3減少。

3.3 インストール方法

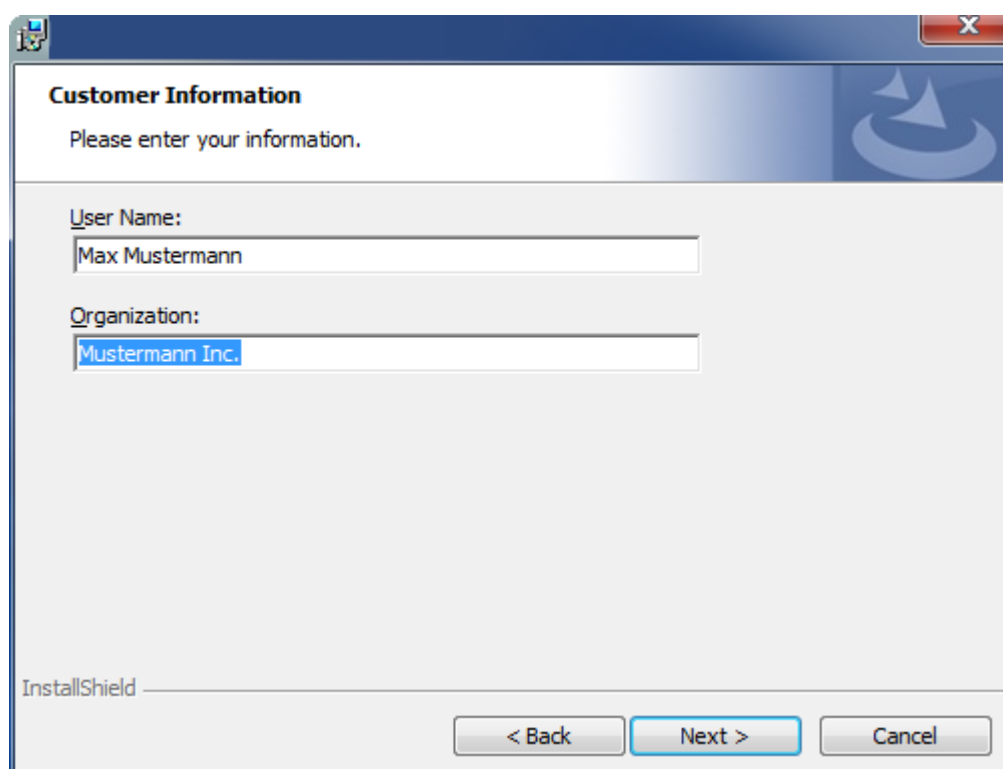
次の項では、WindowsベースのオペレーティングシステムにTwinCAT 3 ファンクションをインストールする方法について説明します。

- ✓ TwinCAT 3 ファンクションの設定ファイルは、ベッコフのウェブサイトからダウンロードしておきます。
1. 設定ファイルを管理者として実行します。これを行うには、ファイルのコンテキストメニューから**Run as administrator**を選択します。
 - ⇒ インストールダイアログが開きます。

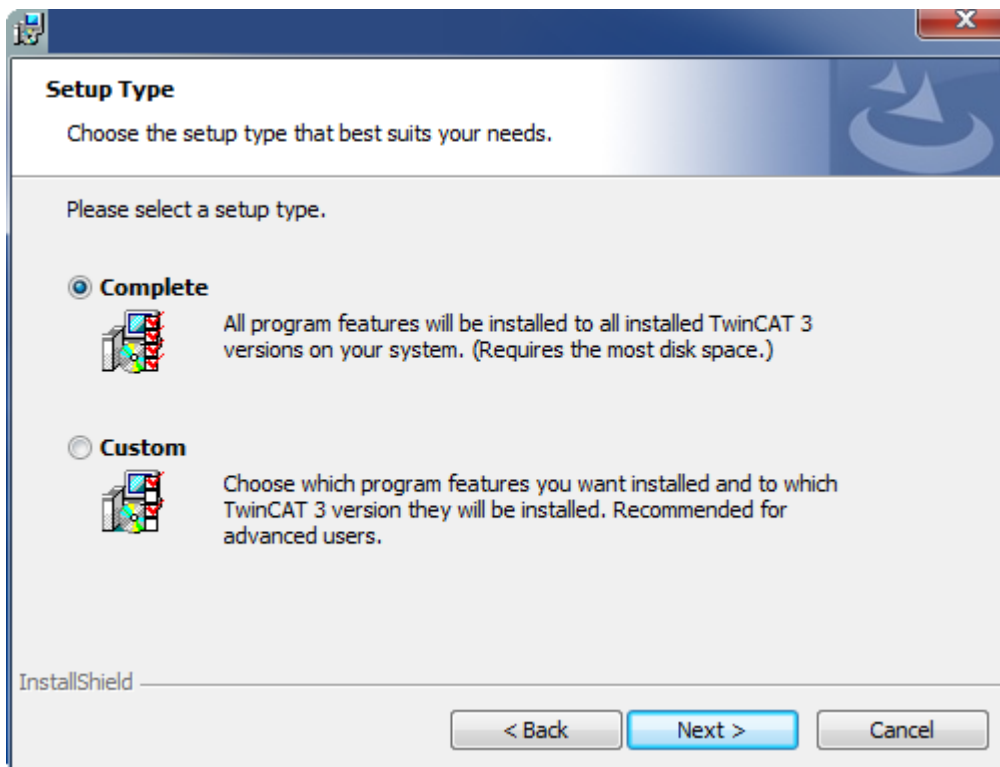
2. エンドユーザライセンス同意書に承諾し、**Next**をクリックします。



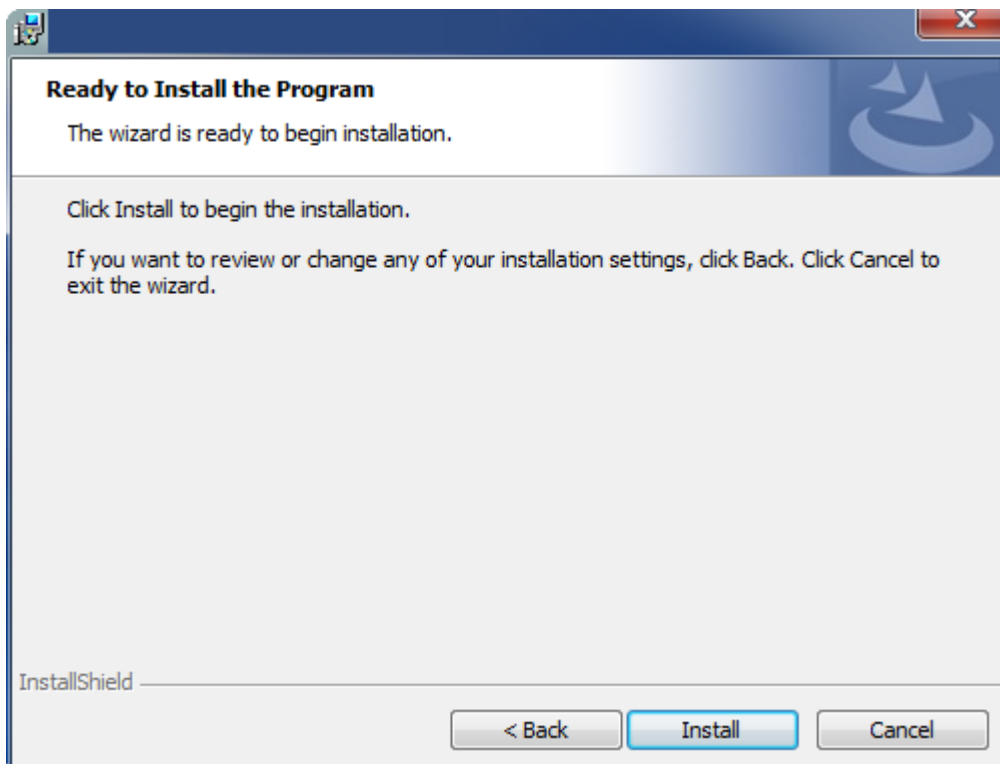
3. ユーザデータを入力します。



4. フルバージョンのTwinCAT 3 ファンクションをインストールする場合、インストールのタイプとして **Complete** を選択します。TwinCAT 3 ファンクションコンポーネントを個別にインストールする場合は、**Custom** を選択します。

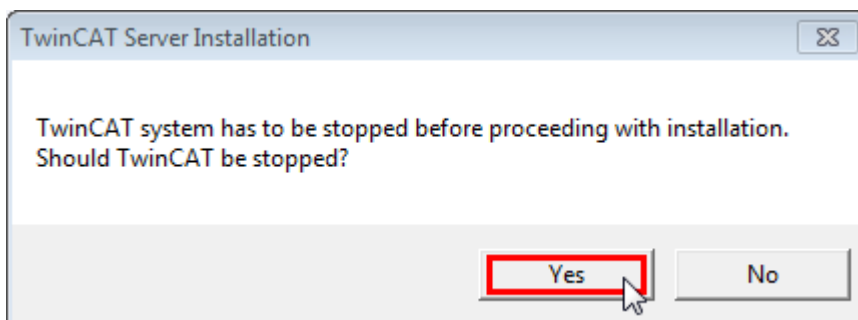


5. **Next**、**Install**の順に選択してインストールを開始します。

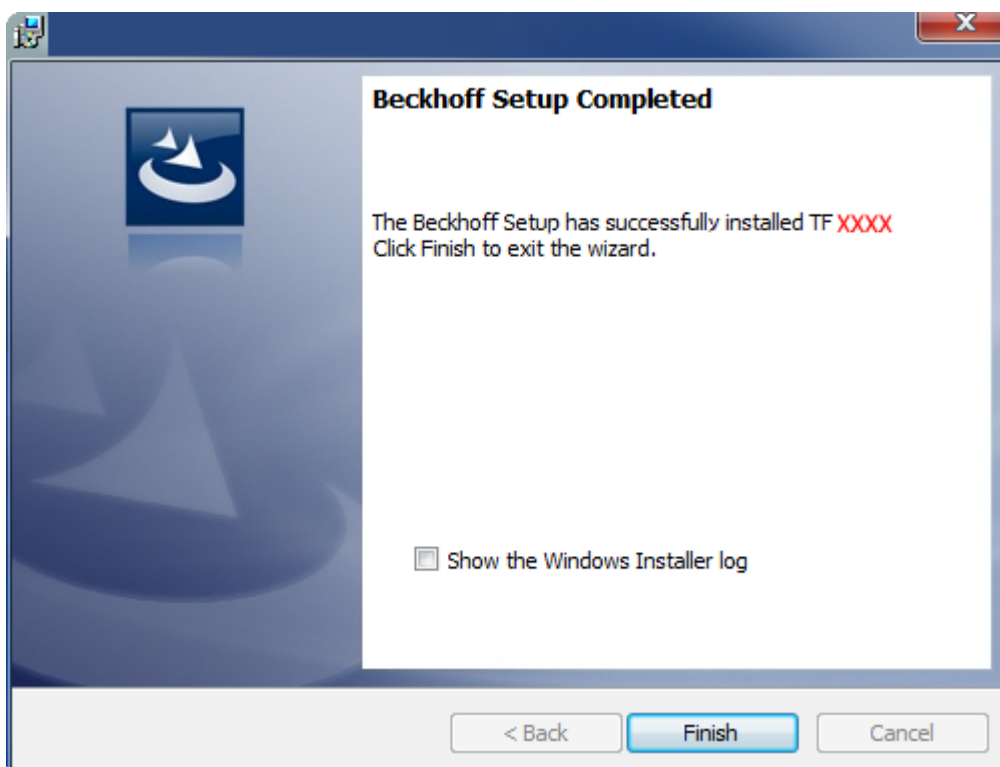


- ⇒ インストールを続行するには、TwinCATシステムを停止する必要があることを通知するダイアログボックスが表示されます。

6. **Yes**でダイアログを確定します。



7. **Finish**を選択して設定を終了します。



⇒ TwinCAT 3 ファンクションが正常にインストールされました。

3.4 インストール方法（TwinCAT 3.1. ビルド 4026）

TwinCATパッケージマネージャ：インストール方法 (TwinCAT 3.1 ビルド 4026)

製品インストールに関する詳細手順は、[TwinCAT 3.1 ビルド 4026インストール手順のワークロードのインストール](#)の章に記載されています。

本製品を使用するためには、以下のワークロードをインストールしてください：

TwinCATパッケージマネージャUI：TF3600 | TwinCAT 3 Condition Monitoring

TwinCATパッケージマネージャCLI：tcpkg install TF3600.ConditionMonitoring.XAE

3.5 ライセンス認証

TwinCAT 3 ファンクションは、フルバージョンまたは7日間のテストバージョンとしてアクティベートできます。どちらのライセンスタイプも、TwinCAT 3開発環境（XAE）からアクティベートできます。

TwinCAT 3ファンクション（フルバージョン）のライセンス認証

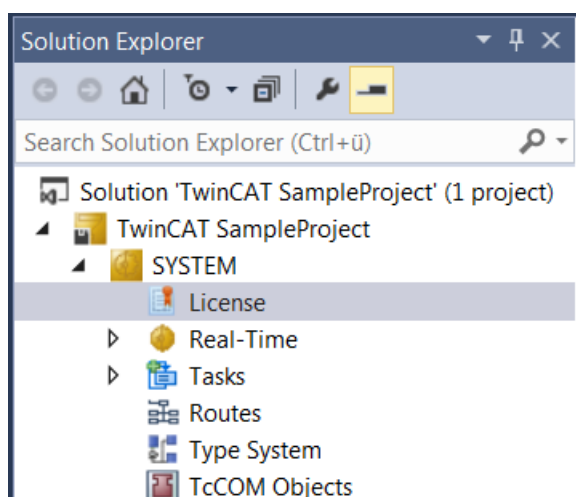
フルバージョンのライセンス取得手順については、Beckhoff Information Systemのドキュメント [TwinCAT 3 ライセンス認証](#)を参照してください。

TwinCAT 3ファンクション（7日間テストバージョン）のライセンス認証



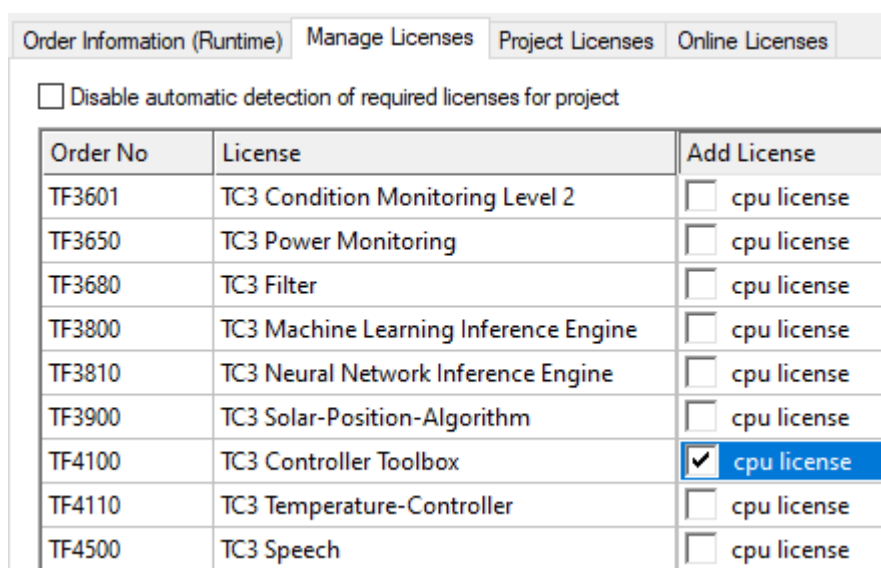
TwinCAT 3 ライセンスドングルでは、7日間テストバージョンを有効化できません。

1. TwinCAT開発環境（XAE）を起動します。
2. 既存のTwinCATプロジェクトを開くか、新規プロジェクトを作成します。
3. リモートデバイスのライセンスをアクティベートする場合は、目的のターゲットシステムを設定します。これを行うには、ツールバーの**Choose Target System**ドロップダウンリストからターゲットシステムを選択します。
 - ⇒ ライセンス設定は、常に選択したターゲットシステムを参照します。ターゲットシステムでプロジェクトが有効になると、設定されたTwinCAT 3 ライセンスがこのシステムに自動的にコピーされます。
4. **Solution Explorer**で、**SYSTEM**サブツリーの**License**をダブルクリックします。



⇒ TwinCAT 3ライセンスマネージャが開きます。

5. **Manage Licenses**タブを開きます。**Add License**の列で、プロジェクトに追加するライセンスのチェックボックスをオンにします（例：TF4100 TC3 Controller Toolbox）。



6. Order Information (Runtime)タブを開きます。

⇒ ライセンス一覧では、先に選択したライセンスが「missing（見つかりません）」というステータスで表示されます。

7. 7-Day Trial License...をクリックして、7日間のトライアルライセンスをアクティベートします。

Order Information (Runtime) | Manage Licenses | Project Licenses | Online Licenses

License Device: Target (Hardware Id) [Add...]

System Id: 2DB25408-B4CD-81DF-5488-6A3D9B49EF19 | Platform: other (91)

License Request

Provider: Beckhoff Automation [Generate File...]

License Id: | Customer Id: |

Comment: |

License Activation

7 Days Trial License... | License Response File...

⇒ ダイアログが開きます。このダイアログに表示されたセキュリティコードを入力することが求められます。

Enter Security Code [X]

Please type the following 5 characters: [Kg8T4]

[] [OK] [Cancel]

8. 表示されたコードを正確に入力し、入力を確定します。

9. アクティベーションが成功したことを示すダイアログが表示されます。

⇒ ライセンス一覧で、ステータスにライセンスの有効期限が表示されるようになりました。

10. TwinCATシステムを再起動します。

⇒ 7日間の試用版が有効になりました。

4 技術概要

4.1 メモリ管理

Condition Monitoringライブラリは、インストールされたドライバが提供するTcCOMオブジェクトを内部使用します。これらは、TwinCAT AMSルータメモリを使用して動的に作成されます。

動的メモリ管理の必要性

全てのメモリ要求と初期化は、初期化フェーズで完了します。入力データと内部構造の要素数は、各ファンクションブロックの構成に依存するため、メモリ領域は原則として動的に割り当てられます。これは、PLC Condition Monitoringライブラリを使用することで自動的に行われます。

初期化時に全てのメモリ割り当てが行われ、ファンクションブロックの初期化で比較的大量のメモリが必要になるため、メモリ不足が原因でこの時点では失敗する可能性があります。ただし、それ以降は発生しません。

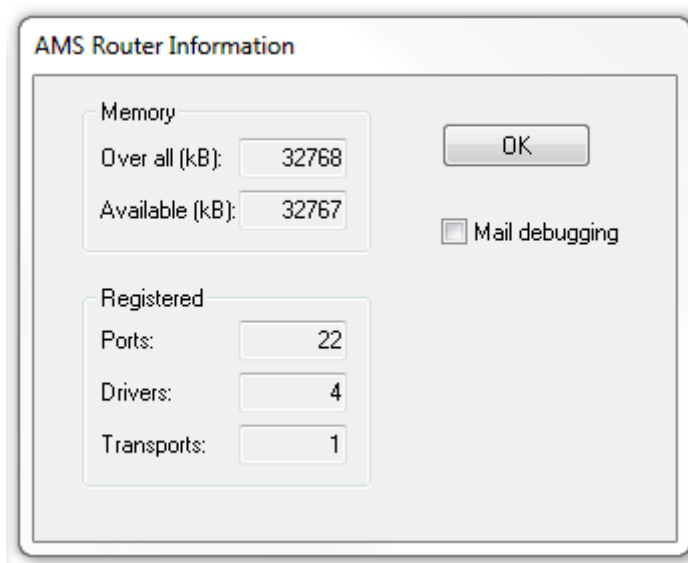
割り当てられたメモリは、オブジェクトが削除されると再び解放されます。

動的に作成されるオブジェクト用のTwinCATルータメモリ

TwinCAT 3 Condition Monitoringライブラリで予約されたバッファは、TwinCAT AMSルータメモリ内のファンクションブロックの初期化時に作成され、リアルタイム環境下で実行できるようになります。高解像度のヒストグラムや分位数、非常に高い解像度のスペクトル計算など、特定の機能は、従来の制御プログラムよりもかなり多くのルータメモリを必要とします。そのため、ルータのメモリサイズを大きくする必要があります。

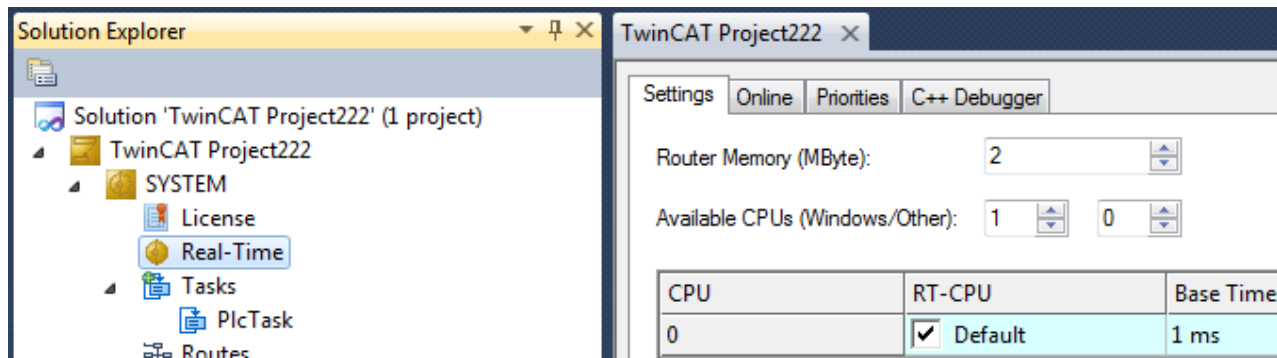
ルータメモリの調整

標準サイズは32 MB（TwinCAT 3.1.4016までは2 MB）です。現在の設定は AMS Router Information ダイアログボックスで表示できます。



ルータのメモリ容量を増やすには、TwinCATの設定（System¥ Real-Time¥ Settings）からMB単位の値を入力し、アクティベートします。

TwinCAT 3.1.4022.4までは、ルータメモリの適合にターゲットデバイスの再起動が必要です。



4.2 タスク設定

複数のリアルタイムタスクがあるアプリケーション

Condition Monitoringの解析チェーンは、データ収集、複数のアルゴリズム、結果の提供による構成されます。結果のさらなる処理とそれに対するプログラムの反応は、アプリケーションにより異なります。

入力データの範囲、例えば入力ベクトルの長さは、それぞれのアプリケーションに大きく依存するため、信号処理ソフトウェアには、長さや要素の型が異なる配列が必要です。そのため、TwinCAT 3 Condition Monitoringライブラリは、数値配列に対して柔軟なデータ構造を採用しています。これにより、数値データをブロックごとに保存、転送、評価できます。多次元データも1次元データも表現できます。

Condition Monitoringのアルゴリズムは、設定によってはCPU負荷が非常に高くなります。そのため、アルゴリズムは優先的に別のタスクに委託されます。この場合、解析チェーンは複数のタスクにまたがります。同期データ交換やスレッドセキュリティに関連する課題は、柔軟に操作可能な解析チェーンを実現するために、ライブラリのファンクションブロックによって内部的にカプセル化されています。

データ交換の詳細については、[並列処理 \[▶ 72\]](#)の項を参照してください。

ヒント：もちろん、プログラムは単一タスクのアプリケーションとして実装することもできます。CPUやタスクのサイクルタイムにもよって、必要なアルゴリズムが十分に高速に処理できる場合は、この方法を推奨します。

タスクのサイクルタイム

解析ステップとバッファサイズは、タスクのサイクルタイムに対する条件を示します。計算は、全ての入力データを処理できるように、十分な頻度で実行しなければなりません。

例：収集データはバッファに格納され、そのサイズは1600エレメントと宣言されています。オーバーサンプリングレートが10倍の場合、バッファが満たされるのに160サイクルかかります。信号収集が1msタスクによってトリガされる場合、タスク計算は160ms未満のサイクルタイムでトリガされなければなりません。

より速い応答（少なくとも0.5倍）を実現するために、計算サイクルタイムをより低い値に設定することが推奨されます。一方、最小限の計算サイクルタイムは、計算するアルゴリズムの複雑度と使用するCPUの性能に依存します。

● 計算サイクルタイムの上限ガイド値

i 計算サイクル時間 < 0.5 * 信号収集サイクル時間 * バッファサイズ / オーバーサンプリングレート

ほとんどのアルゴリズム（スペクトル、ケプストラムなど）には、計算負荷の高い数学的演算が含まれています。これらは、十分なサイクルタイムを確保したタスクのコンテキストで呼び出す必要があります。必要な実行ポイントもハードウェアのプラットフォームに依存します。上記の式は、計算サイクルタイムの上限目安値を示しています。例えば、各ファンクションブロックには、下限ガイド値を推定するためのプロファイルが用意されており、オンラインモニタリング中に起動できます。このプロファイルは、ファンクション

ブロックのインスタンスfbImplementation → fbExecutionTimeMonitoring にあります。**手動で** bMeasureMeasureMaxExecTime を設定することで、プロファイラーが起動します。通常どおり、ファンクションブロックの内部変数にプログラムでアクセスすることは避けるべきです。

RealFFT_Sample.RealFFT.MAIN_CM		
Expression	Type	Value
nCntResults	ULINT	278
stInitParsResultBuffer	ST_MA_MultiArr...	
fbImplementation	FB_CMA_Imple...	
nDiscardFirstResults	USINT	0
fbExecTimeMonitoring	FB_CMA_ExecTi...	
bMeasureMaxExecTime	BOOL	TRUE
fbProfiler	Profiler	
tMaxExecTime	LTIME	LTIME#656us600ns
tMaxElapsedTimeout	LTIME	LTIME#15us800ns
bValueInitialized	BOOL	TRUE
bObjectsInitialized	BOOL	TRUE
bStreamInitialized	BOOL	TRUE
bStreamsAllocated	BOOL	TRUE

表示される値は最大実行時間です。実行ポイントが長くなる可能性のあるパラメータと入力値の組み合わせに対して、タスク設定では、少しの余裕を確保すべきです。

上記の考慮すべき事項の例外は、いくつかの統計ファンクションブロック（分位数、ヒストグラムなど）です。原則として、これらのファンクションブロックは、最初は内部メモリに数タスクサイクル分のデータしか追加しません。時間がかかるのは、その後の計算（Nサイクル後のデータ収集）だけです。対応するタスクのサイクルタイムは、計算することなく単純な呼び出しに合わせることができます。このため、計算を伴う呼び出しの場合、サイクルタイムを超過しますが、迅速な応答が可能になります。これはPLCプログラミングの特殊ケースです。通常、タスクのサイクルタイムは超過してはいけません。

● サイクルタイムに注意

i Condition Monitoringアルゴリズムを呼び出すだけのタスクの場合、サイクルタイムを超えることがほとんどないよう調整可能です。このタスクによって呼び出されるプログラムブロックには、他のプログラムコードを含めないようにしてください。そして、このような遅いタスクの優先順位は、もちろん他のタスクよりも低くすべきです。

浮動小数点の例外処理

これらの例外は、タスクごとに個別に無効化できます。デフォルトでは有効になっています。

The screenshot shows the 'Parameter (Online)' tab for a task named 'PlcTask'. In the 'Options' section, the 'Floating point exceptions' checkbox is checked and highlighted with a red rectangular box. Other visible settings include 'Port' set to 350, 'Object Id' set to 0x02010020, 'Priority' set to 20, 'Cycle ticks' set to 1.000 ms, and 'Watchdog Cycles' set to 0.

アルゴリズム呼び出しの中には、NaN（非数）結果を導くものがあります。アプリケーションで NaN を処理する場合、このタスクでは FP 例外を無効にしてください。その上で、プログラムコード全体と全ての関数がNaNを処理できることを検証してください。

NaN値の取り扱いに関する詳細は、別項NaN値 [▶ 71]に記載されています。

⚠ 注意

実行停止

浮動小数点の例外処理はデフォルトで有効です。NaN（非数）との比較は、実行停止を引き起こす例外を発生させ、機械に損傷を与える可能性があります。NaNの処理を行う前に、早急に結果を確認することを強く推奨します。（「NaN値」の項を参照）

4.3 NaN値

エラーコード [▶ 367]によるエラー処理は、最良の選択ではない場合があります。特に、通常とは異なる、しかし原理的には可能な入力データのためにオペレーションが未定義の値を返す場合や、処理から値を除外する場合などです。

IEEE 754規格では、このような目的のためにNaN（非数）というカテゴリの記号値を定義しています。TwinCAT 3 Condition Monitoringライブラリでは、次のような場合にこれらが生成または考慮されます：

- 統計的評価のために有効な値が不十分である場合、結果は NaN で出力されます。
- 統計的評価から特定の値を除外する場合、入力の NaN 値を無視するファンクションブロックによって除外されます。
- 時系列の周波数解析の入力ベクトルに1つ以上のNaN値が含まれる場合、出力ベクトル全体がNaNで満たされます。この特性は、入力信号のギャップが時系列のジャンプにつながる場合に、結果を未定義としてマークするために使用できます。時系列データが断片化しているため、正しいスペクトルを計算することはできません。

NaN値を生成するファンクションブロックの機能は、その説明に記載されています。

以下の点は、NaN値の主な特徴です。

- NaNを入力データとして使用する全ての算術演算は、演算結果としてNaNを返します。
- 少なくともオペランドの1つがNaNの場合、全ての関係演算子=、≠、>、<、>=、<=は、必ず値Falseを返す。
- 引数が値NaNの場合、スタンダードCファンクションisnan()、_isnan()、またはPLCファンクションLrealIsNaN()とRealIsNaN (Tc2_Uilitiesライブラリ)は、値TRUEを返します。

- 式 $\text{isnan}(a)$ は、式 $!(a == a)$ または $\text{NOT}(a = a)$ に相当します。

さらに計算で使用する場合、NaN値が自身を再生成するという事実は、無効な値が見過ごされることができないという点で 有利です。

⚠ 注意

ソフトウェアの誤作動

NaN値は明示的に承認される場合、NaN値をPLCライブラリでのみ使用できます。特に、モーション制御とドライブ制御のファンクションで制御値として使用できます。そうでない場合、NaN値は該当するソフトウェアに潜在的に危険な誤作動を引き起こすことがあります。

⚠ 注意

浮動小数点の例外処理

NaNが使用され、アプリケーションで処理される場合、FPの例外処理はオフにしなければいけません。オフでない場合、NaNとの比較が例外処理を引き起こし、ランタイムの停止と機械の損傷を引き起こします。

FP例外のオフ/オン切り替えオプションの詳細には、[タスク設定 \[▶ 69\]](#)の章を参照してください。

4.4 転送トレイを使った並列処理

次の項では、TwinCAT 3 Condition Monitoringライブラリが提供するスレッドセーフおよびマルチコア対応のデータ転送について説明します。

計算負荷の高い処理における非同期通信と並列実行

Condition Monitoringのアプリケーションは、数メガバイトのデータセットを必要とすることが多く、演算時間と性能要件が高まります。許容される最大演算時間はサイクルタイムに基づきます。例えばドライブコントローラではこれを絶対に超過してはいけません。このため、TwinCAT 3 Condition Monitoringアプリケーションのマルチタスクソフトウェアアーキテクチャは、計算量の多いアルゴリズムの場合に推奨されます。[タスク設定 \[▶ 69\]](#)の章を参照。

転送トレイのアイデア

そのためには、アルゴリズムのスレッドセーフな実装が必要です。TwinCAT 3 Condition Monitoringライブラリは、非常に効率的で使いやすい通信メカニズムを提供し、データのロックとロック解除の典型的な問題を可能な限り排除します。このライブラリは、例えば異なる速度でデータを並列処理するために非常に効率的なメカニズムを提供します。これにより、排他的な同期アクセスを行う複数のタスク間で、転送トレイに基づくキューを使用して、配列データをエラーなく転送することができます。また、同期の問題なしにマルチコアCPUの使用が可能になり、数値データの同期されていないオーバーライドに起因するブロッキングや不整合などの診断が困難なエラーを防止します。

マルチアレイバッファへの並列書き込みアクセス（マルチアレイの取り扱い方法 [\[▶ 74\]](#)の項を参照）と、同じファンクションブロックの並列実行は明示的に禁止されているため、ライブラリのファンクションブロックはグローバル変数のリストでグローバルインスタンスとして宣言することはできません。

例：サイクルタイム遷移の必要性

状況によっては、連続的な概念では不十分な場合もあります。これは、あるデータセットの処理に、制御タスクのサイクルタイムが許容する以上の時間がかかる場合に常に起こります。

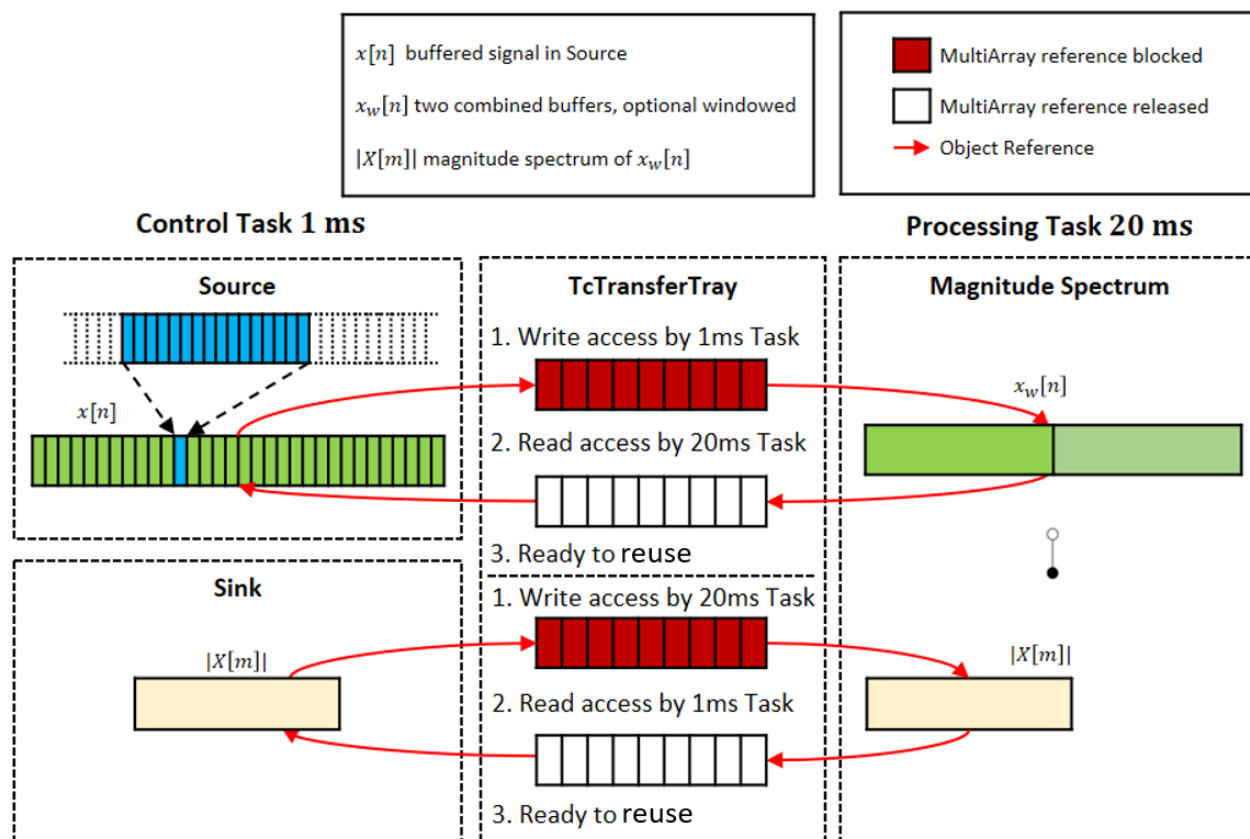
例えば、制御タスクのサイクルタイムは1ms、データのオーバーサンプリングは20サンプル/サイクル（サンプリングレート20kHzに相当）の場合です。信号処理には0.16Hzの周波数分解能が必要です。これは、例えば大型のローラーベアリングの解析において、内輪と外輪の軌道面がわずかに異なる速度で回転している場合に、それぞれの欠陥を区別するために必要です。

FFT長 N 、周波数分解能 Δf 、サンプリングレート f_s の関係は次の通りです： $N = f_s / \Delta f$ （簡略化のため、ここでは長方形のウィンドウを想定）。その結果、FFTの長さは $N = 125000$ となります。さらに、FFT長 N' は2の累乗でなければなりません。したがって、 $\log_2(125000) = 16.93$ となるため、長さ N の信号は、長さ $N' = 2^{17} = 131072$ までゼロ埋められます。

必要な計算時間はCPUの性能によりますが、制御タスクの中で計算することは間違いなく不可能です。必要な入力データ量は数秒の信号セグメントに相当するため、計算はほとんど必要ありません。

転送トレイによるソリューションの概念

Condition Monitoringライブラリが提供する高性能なソリューションを下図に示します。制御タスクは、オーバーサンプリングターミナル（図では青色）を介して、20サンプルの「パケット」でデータを収集します。これらは、振幅スペクトルファンクションブロックの入力バッファの長さに対応するサイズのバッファに格納されます（ $125000 / 20 = 6250$ 、図では緑色）。バッファが満杯になると、つまりControl Taskが3125サイクル終了すると、そのオブジェクト参照は非同期通信メカニズム（FIFO）により第2のタスク（Processing Task）に転送されます。これはサイクルタイム20msよりはるかに長くなります。タスク設定 [▶ 69] に記載されている経験則によれば、計算タスクの最大サイクルタイムは1562.5msです。この要件は、明らかに20msという値で満たされます。



この通信メカニズムは、ハードウェアで保護されたいわゆるアトミック操作と呼ばれる手法を使用し、タスクのうち1つだけが同時に対応するバッファ（以下、マルチアレイとも呼ばれる）にアクセスできることを保証します。これは、銀行ウィンドウ口転送トレイに似ており、顧客か受付係のどちらか（ただし、両方が同時ではない）が、その中身にアクセスできるようになっています。

● 応答待ち時間

i

キューにはFIFO（先入れ先出し）の原則が適用されます。そのため、加えて非同期通信のため、結果はすぐには得られません。応答待ち時間が変化することもあります。

計算結果（マグニチュードスペクトル）は、同じ通信メカニズムを使用した別のキューを介して制御タスクに返され、制御タスクはそれをさらに評価することができます。もちろん、第3の別のタスクへの通信や、計算タスク自体で結果を出すことも可能です。

一般的に、演算タスクはモーションアプリケーションと比較して、厳しいリアルタイム条件が求められないため、制御タスクよりも低い優先度で実行できます。TwinCAT 3システムのタスク管理により、優先順位の最も高いタスクが常に最初に実行されるため、複雑な計算でもこれらのリアルタイム条件を満たすことができます。

この概念は、シングルコアCPUでもマルチコアCPUでも使用できます。中央のロックがボトルネックを引き起こすことなく、多くのコアへの分配が可能です。

● タイムアウト

1 転送トレイの内部通信コマンドは、ハードウェアの特性などにより、まれに失敗することがあります。例えば、キューにある空のバッファに他のタスクがアクセスしているため、それを取り除くことができない場合です。同期タイムアウトとして認定され、結果としてタイムアウトエラーが発生する可能性があります。したがって、プログラムは常に、信号データの継続に必要なバッファが利用できないエラー状態に備えておく必要があります。データのオーバーフローや解析結果の不連続な時系列など、結果として生じるエラーは一貫した方法で処理すべきです。解析チェーンの入力信号データがエラーなく収集できる限り、不連続は発生しません。下流のアルゴリズムのファンクションブロックでタイムアウトが一度発生した場合、または使用可能な結果のマルチアレイバッファがなかった場合、入力データも結果データも失われません。次の呼び出し時に転送されます。

転送トレイの仕組み

転送トレイ自体は、Tc3_CMライブラリが提供する内部ファンクションブロックを使用して表示されます。このファンクションブロックは、グローバル構造インスタンスで定義された初期パラメータで初期化されます。

キューの典型的な使用法は、あるタスクからのバッファが、固定されたデータストリームの識別子とともにキューに追加されます。これらのバッファは、処理のために他の特定のタスクから取り除かれます。これらのバッファは、その後、異なるバインディング識別子を持つ別のキューを通じて送り返され、再利用されます。しかし、統計データを解析する場合など、複数のタスクが同じキューに読み書きアクセスしても問題はありせん。

マルチアレイ・バッファ

いわゆるマルチアレイバッファは、あるタスクから次のタスクへの転送トレイを介したデータ通信に使用します。これについては[マルチアレイ機能の使用 \[▶ 74\]](#)の章で説明しています。

4.5 マルチアレイの取り扱い方法

マルチアレイは**多次元データバッファ**で、Condition Monitoringライブラリにおいて、転送トレイと組み合わせて使用します。これにより、アプリケーションは複数のPLCタスク間で多次元データを簡単に交換できるようになります。タスク間の通信中、メモリはコピーされず、データバッファへの参照のみが転送されるため、通信は非常に効率的です。通信に必要なオーバーヘッドはごくわずかで、実行時間はマイクロ秒の範囲です。

マルチアレイ通信リング

入力または結果のデータストリームに対するマルチアレイの充填（コンテンツの書き込み）と送信（アクセス権の転送）は、結果として、「空の」マルチアレイが常に必要です。このため、評価されたマルチアレイは、それを充填したタスクに「空の」データコンテナとして返されます。これはマルチアレイの継続したサイクルを生成します（転送トレイを使った並列処理 [\[▶ 72\]](#) の図を参照）。

通常、1つの回路につき少なくとも3つのマルチアレイが必要です：最初のマルチアレイは制御タスクに「所属」し、新しいデータで満たされようとしています。プロセスタスクは2番目のマルチアレイにアクセスし、それを処理します。制御タスクが現在のマルチアレイを満たしているものの、残りのオーバーサンプリングデータを正確にこのサイクルで次のマルチアレイに書き込む必要がある場合に備え、3番目のマルチアレイを予備として保持しておく必要があります。したがって、最小数は3です。

● マルチアレイの数

1 安全のため、ワーストケースを考慮して回路あたり4つのマルチアレイを推奨します。複数のアルゴリズムがマルチアレイのデータにアクセスする場合、アルゴリズムごとにさらに追加のマルチアレイを提供することが推奨されます。

提供されるマルチアレイの数は、Condition Monitoringライブラリのファンクションブロックの入力パラメータ `nResultBuffers` で設定します。デフォルトの値は4です。

● 通信リング内のマルチアレイ数

i

4つ以上のマルチアレイが必要なのは、結果のバッファ（＝マルチアレイ）を複数のアルゴリズムで直接処理する場合のみです。言い換えれば、通信リング内の2つ以上の解析モジュールがこれらの結果に関与する場合です。解析モジュールを追加するごとに、結果のバッファ数を1つ増やすことを推奨します。非同期通信リングで使用するマルチアレイバッファの数は、各解析ファンクションブロックで設定できます。

これらの追加バッファは内部で作成、管理されます。これらはAMSルータに一定量の追加メモリを必要とします。

基本的に、マルチアレイの次元は、長さ、サイズ、データ型まで個別に設定できます。これらのパラメータは、マルチアレイのライフサイクル全体の形状を決定します。

マルチアレイの内部構造は自動的に管理され、プログラミングの必要はありません。マルチアレイの疲労寿命は、アプリケーションの疲労寿命と同じであり、PLCの起動から停止までの間に適用されます。マルチアレイは、いわゆる転送トレイを使用して、あるタスクから別のタスクへ転送されます。

概念は非常に柔軟です。他のタスクやCPUに計算を変更したり再配分したりするのは簡単で単純です。

マルチアレイの設定

マルチアレイはST_MA_MultiArray_InitPars [▶ 303]構造体で設定します。これはTc3_MultiArrayライブラリの一部であり、Condition Monitoring 設定ファイルとともにインストールされる。

マルチアレイ設定の例：

```
cInitSource : ST_MA_MultiArray_InitPars:= ( eTypeCode := eMA_TypeCode_LREAL,
                                             nDims := 2,
                                             aDimSizes := [cChannels, cBufferLength]);
```

マルチアレイをファンクションブロックFB_CMA_Sourceと併用する場合、ソースインスタンスfbSourceには、1つまたは複数の設定済みのマルチアレイインスタンスが必要です。上記のマルチアレイは2次元（nDims = 2, nDims = 1 も可）です。次元のサイズはaDimSizes で記述されます。したがって、記述されたマルチアレイは、各要素のデータ型がLREALで、次元はcChannels x cBufferLengthです。

FB_CMA_Sourceとマルチアレイの使用例：

```
fbSource : FB_CMA_Source := ( stInitPars := cInitSource,
                              nOwnId := eID_Source,
                              aDestIDs := [eID_Rms],
                              nResultBuffers := 4);
```

マルチアレイは、データストレージの管理という点で柔軟性があります。例えば、上記の場合、行と列は完全に交換可能です。（以下の例に示すように）次元が正しく割り当て/識別されている場合、これは結果に影響しません。

高度な設定オプション

以下の例でわかるように、FB_CMA_Source [▶ 230]（またはFB_CMA_Sink [▶ 223]、FB_CMA_BufferConverting [▶ 90]）は、nWorkDim、pStartIndex またはnElementsDim のようなパラメータを提供します。これらのパラメータは以下のように使用できます：

- マルチアレイの特定セグメントの記述／読み出し
- 特定の場所からの書き込み/読み取り/コピー
- 特定のポイント以降の要素を一定数コピー

これらのパラメータの組み合わせは、メモリの最適化を保証するだけでなく、マルチチャンネル、マルチタスクアプリケーションにおける選択性も保証します。下の例を参照してください。

アプリケーションシナリオ

このアプリケーションシナリオは、TwinCAT Condition Monitoringアプリケーションエリア内でのみ有効です。前述のように、マルチアレイは自動的に管理されますが、最初に初期化する必要があります。これは、ST_MA_MultiArray_InitPars を使用してPLC宣言で行われ、FB_CMA_Source インスタンスに渡されます。

各アルゴリズムのファンクションブロックは、stInitParsで設定されたマルチアレイを使用して結果を転送します。これらの形状は、FB_CMA_Sinkを除き、初期化パラメータで定義されます（各ファンクションブロックの説明を参照）。マルチアレイの一部をPLC配列にコピーして、追加の処理や評価を行うことも可能です。これはFB_CMA_BufferConvertingを使用して行われます。

ファンクションブロックには、PLC変数マルチアレイに書き込んだり読み込んだりするメソッドがあります。メソッドとそのパラメータの詳細については、ファンクションブロックの説明を参照してください。



ファンクションブロックFB_CMA_Sinkは、マルチアレイの初期化を必要としません。FB_CMA_Sinkで使用するマルチアレイの形状は、内部で指定されます。

WorkDimと呼ばれるマルチアレイの各次元には、0から始まるインデックスがあります。

2次元マルチアレイの場合、作業次元0は通常Condition Monitoringライブラリのチャンネル数とリンクしています（上記本文中「マルチアレイ設定の例」を参照）。

マルチアレイの処理例

状態監視アプリケーションにおけるマルチアレイの使用方法をよりよく理解するために、以下のケーススタディを検証します。

例では、2つのEL3632を用いて加速度センサからの3つの信号が、オーバーサンプリング係数10で記録されます。入力データは長さ1000のマルチアレイに収集され、ファンクションブロックに転送されます。この場合、FB_CMA_MomentCoefficients [▶ 172] を計算するファンクションブロックです。

FB_CMA_MomentCoefficients は、設定に応じて、各チャンネルの入力データに対する様々な統計パラメータを計算します。ここでの目的は、FB_CMA_MomentCoefficient の出力でマルチアレイを設定し、例えば平均値や標準偏差など、結果の特定の部分のみ出力するようにすることです。

入出力変数は以下のように宣言され、初期化されます：

```
cInitSource : ST_CM_MultiArray_InitPars := (eTypeCode := eMA_TypeCode_LREAL,
      nDims := 2,
      aDimSizes := [3,1000]);

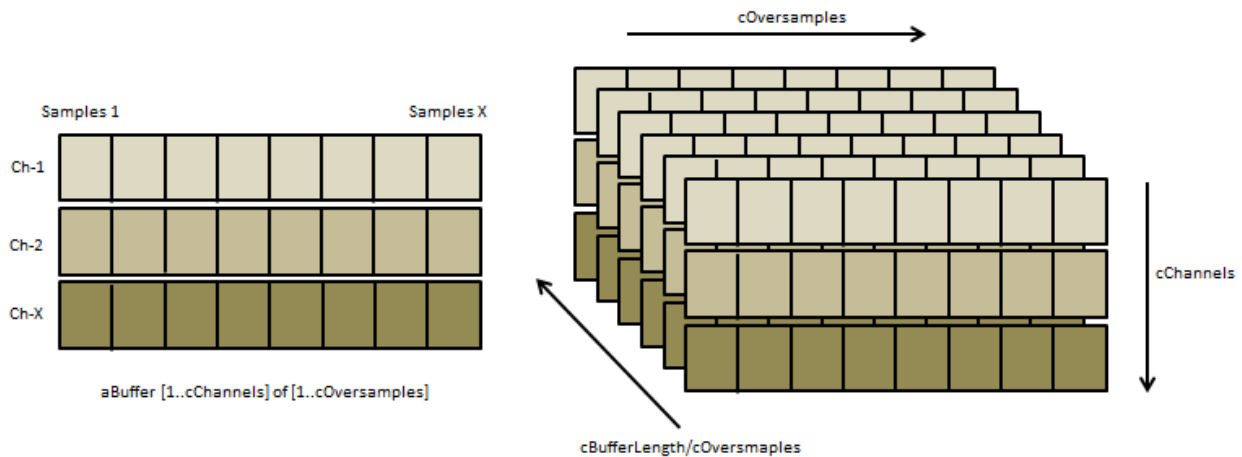
aBuffer : ARRAY [1..3] OF ARRAY [1..cOverSamples] OF LREAL;
fbSource : FB_CMA_Source := (stInitPars := cInitSource,
      nOwnID := eID_Source,
      aDestIDs := [eID_MomentCoeffs]);

// MultiArray indices begin with 0, not 1!
// aStartIndex := [0,0],[0,1],[0,2],[1,0],[1,1],[1,2],[2,0],...
aStartIndex : ARRAY [1..2] OF UDINT := [0, 1];

// Select channels := 1: one, 2: one and two, 3: one, two and three and so on
// Select moments := 0: count, 1: mean, 2: standard deviation, 3: skew, 4: kurtosis
aMomentCoef : ARRAY [1..3, 1..2] OF LREAL;
```

上に示したように、fbSource は2次元のマルチアレイを取得し、適切なバッファリングの後、aBuffer からFB_CMA_MomentCoefficients にデータを渡す必要があります。初期化パラメータの関数としてデータを保存できます：

- 行をチャンネルとして、列をサンプリング結果として保存するか、
- または、行をサンプリング結果として、列をチャンネルとして保存します。



マルチアレイは2次元なので、**Input2D()**メソッドを呼び出してこれを実施します。

```
fbSource.Input2D(pDataIn := ADR(aBuffer),
  nDataInSize := SIZEOF(aBuffer),
  eElementType := eMA_TypeCode_LREAL,
  nWorkDim0 := 0, (* aBuffer stores channels across first dim*)
  nWorkDim1 := 1, (* aBuffer stores samples across second dim*)
  pStartIndex := 0,
  nOptionPars := 0 );
```

このメソッドの呼び出しを順を追って説明します：

- ローカルPLC変数aBuffer が参照として渡されます。
- 転送するデータ型を指定します。
- このメソッドは、マルチアレイの第1の作業次元をaBuffer の第1の次元に割り当て(cChannels)、第2の作業次元をサンプリング値(cOversamples)に割り当てます。あるいは、例えば変数aBuffer : ARRAY [1.. cOversamples] OF ARRAY [1.. 3] OF LREAL を宣言し、必要な転置をnWorkDim0=1 とnWorkDim1=0 で実現することもできます。
- pStartIndex=0は、aBuffer 全体をマルチアレイにコピーします。これはデフォルト設定です。配列の一部だけをコピーする方法を、FB_CMA_Sink を使って以下に示します。

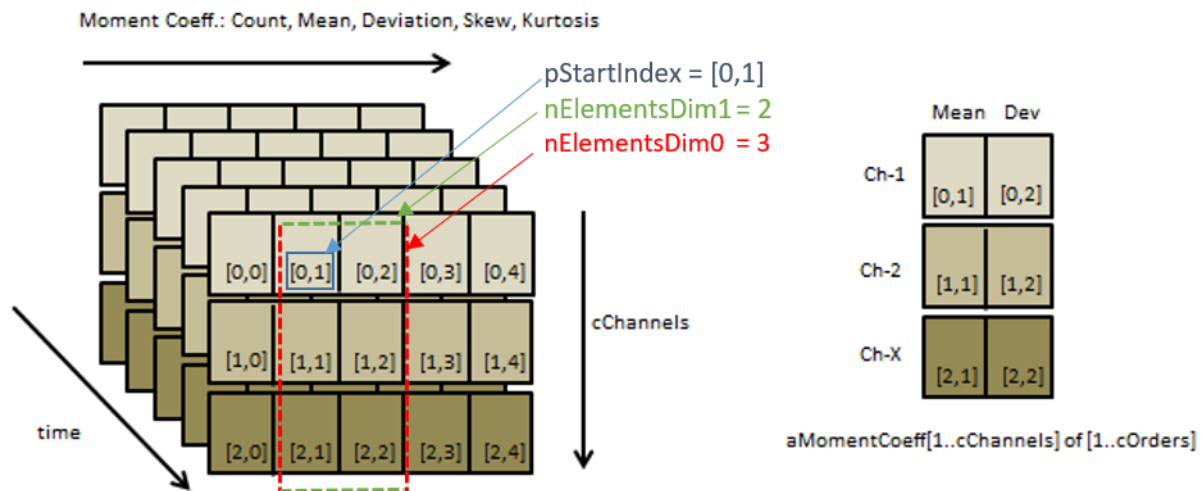
上記の全ての設定によって、マルチアレイはその第1次元（行）にチャンネルを、そして第2次元（列）に長さcBufferLengthまでのサンプリング値を格納するように完全に構成されます。

同様に、FB_CMA_Sink インスタンスは、マルチアレイ の内容をローカルPLC変数aMomentCoef に書き込むことができます。

```
fbSink.Output2D(pDataOut := ADR(aMomentCoef),
  nDataOutSize := SIZEOF(aMomentCoef),
  eElementType := E_MA_ElementTypeCode.eMA_TypeCode_LREAL,
  nWorkDim0 := 0, (* aMomentCoef stores channels across first dim *)
  nWorkDim1 := 1, (* aMomentCoef stores moments across second dim *)
  nElementsDim0 := 3, (* aMomentCoef stores all 3 channels *)
  nElementsDim1 := 2, (* aMomentCoef stores mean and deviation*)
  pStartIndex := ADR(aStartIndex),
  nOptionPars := 0);
```

もう一度、このメソッド呼び出しを順を追って説明します：

- ローカルPLC変数aMomentCoef （書き込みアクセスが必要）が参照として渡されます。
- データ型を指定します。
- マルチアレイの最初の作業次元は、変数aMomentCoef の最初の作業次元、すなわちチャンネルに割り当てられます。2つ目の次元は、同様に転送され、統計パラメータ数、平均、偏差、歪度、尖度に対応します。
- パラメータnElementsDim0 とnElementsDim1 は、WorkDim0 方向と WorkDim1 方向にコピーするマルチアレイの要素数を指定します。この場合、WorkDim0方向に3要素（3チャンネル全て）、WorkDim1方向に2要素となります。
- パラメータpStartIndex は、コピーする2x3の矩形の最初の要素を定義します。パラメータは2次元配列（ここではaStartIndex ）へのポインタです。



図示の構成では、Output2D() メソッドは、マルチアレイの1セグメントのみをPLC 変数aMomentCoeff にコピーします。上で説明したように、コピーするセグメントは、パラメータnWorkDim0、nWorkDim1、nElementsDim0、nElementsDim1、pStartIndex で設定します。

5 PLC API

TwinCAT3 Condition Monitoringライブラリは、TwinCAT PLCアプリケーションに解析オプションを提供します。製品の概要と重要な詳細情報については、製品説明 [▶ 10]と技術紹介を参照ください。

PLC APIセットは、3つの**PLC ライブラリ**で構成されます。これらのライブラリは、Condition Monitoring PLCプロジェクトに統合する必要があります：

- Tc3_CM
- Tc3_CM_Base
- Tc3_MultiArray

状態監視解析

計測データのロギング、異なるアルゴリズムに基づく処理、結果の評価などのプログラミングに加えて、各信号処理は適切な解析チェーンに依存します。そのため、TwinCAT 3 Condition Monitoring・ライブラリは、計画した解析チェーンの実装を実質的には純粋なパラメータ設定作業に変えるためのファンクションブロックをサポートしています。

解析チェーンの図式

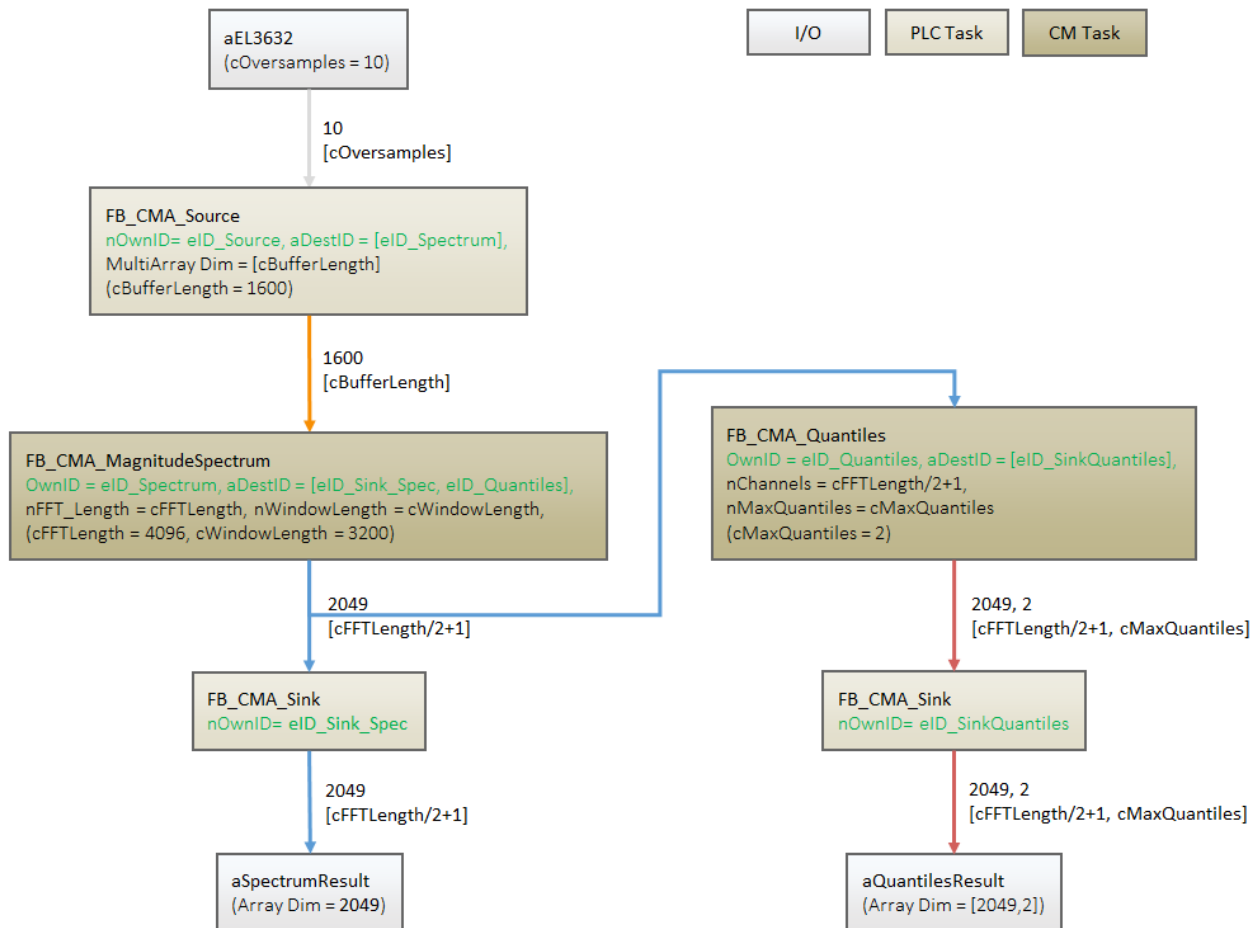
状態監視アプリケーションをプログラミングする前に、解析ステップを図式化することは合理的です（下図の例）。

これには、各PLCファンクションブロックの表現が含まれます。通常、少なくとも2つのタスクが使用されます。1つは通常の制御プログラム用のタスクで、もう1つはCondition Monitoringの計算負荷の高い操作用の（より低速で優先順位の低い）タスクです。

各解析ファンクションブロックは、相互に特殊な通信方法を使用します。この内部実装によって、複数のタスクにまたがる相互通信も可能になります。内部的には、1つの転送トレイオブジェクトと複数のマルチアレイを使用します（[並列処理 \[▶ 72\]](#)の章を参照）。ただし、ファンクションブロックやそのメソッドは、アプリケーションのタスクコンテキスト内からのみ呼び出すことができます。

解析ファンクションブロックは、異なるタスクコンテキストに配置することが可能です。解析ステップの順序は、転送ID（下図の緑色の値）を使って割り当てられます。各ファンクションブロックは、任意のIDと、結果の送信先であるターゲットIDを受け取ります。転送IDは列挙として定義するのが最適です。

下図は、グレー、オレンジ、青、赤の4種類でデータバッファを示しています。対応する全てのバッファ（PLC配列、マルチアレイ）およびアルゴリズムパラメータの形状は、これらのバッファサイズに一致する必要があります。



● 周期的な呼び出し

i `FB_CMA_Source`の機能が呼び出され、信号データが対象のファンクションブロックに転送される限り、解析チェーンの他の全てのモジュールは周期的に呼び出される必要があります。並列処理の章の内部通信原理の説明を参照してください。特定のフェーズで全てのターゲットブロックが処理されない場合、これらの呼び出しはまだ必要ですが、`PassInputs()`メソッドを使用すると、結果を生成せずに入力バッファのみを渡すことができます。

● 連鎖的なエラーに注意

i 解析ファンクションブロックの周期的なエラーは、解析チェーンに連鎖的なエラーを引き起こす可能性があります。

オンライン表示:

Condition Monitoringライブラリのファンクションブロックは、ファンクションブロックの現在の状態に関する重要な情報をオンライン表示に示します。これらは、アプリケーションのプログラミング（ファンクションブロックの初期化と設定、解析チェーンのバッファサイズの調整など）だけでなく、解析と評価（エラー解析、グラフィック表示）にも使用できます。

下の図は、ファンクションブロック `FB_CMA_MagnitudeSpectrum` のオンライン表示を示しています。ノードとその用途については、以下で詳しく説明します。

Expression	Type	Value
fbAlgorithm	FB_CMA_MagnitudeSpectrum	
bError	BOOL	FALSE
hrErrorCode	HRESULT	16#00000000
ipErrorMessage	I_TcMessage	16#FFFFB8691299868
nCntResults	ULINT	0
nOwnID	UDINT	2
aDestIDs	ARRAY [1..cCMA_MaxDest] OF UDINT	
nResultBuffers	UDINT	4
tTransferTimeout	LTIME	LTIME#500us
stInitPars	ST_CM_MagnitudeSpectrum_InitPars	
stInitParsResultBuffer	ST_MA_MultiArray_InitPars	
fbImplementation	FB_CMA_Implementation	
fbBaseAlgorithm	FB_CM_MagnitudeSpectrum	
eTraceLevel	TCEVENTSEVERITY	Critical

戻り値

エラーの場合、bError := TRUE が設定され、戻り値hrErrorCode にエラーコードが格納されます（エラーコードの概要 [▶ 367] を参照）。

複数のチャンネル（nChannels > 1）を処理する場合、各チャンネルが異なる戻り値になる可能性があります。この場合、戻り値はファンクションブロックで個別に問い合わせできます。全てのチャンネルではなく、1つ以上のチャンネルの結果が許容されない場合、ファンクションブロックの値は eCM_InfRTime_AmbiguousChannelResults に対応します。全てのチャンネルの結果が許容されない場合、ファンクションブロック上の値は eCM_ErrRTime_ErrorousChannelResults に対応します。

エラーリストの読み方と処理の例は、マルチチャンネルのマグニチュードスペクトル [▶ 314] を参照してください。

エラーの説明と情報

現在の戻り値に関する詳細情報は、ipErrorMessage を参照してください。sEventText は、発生したエラーの説明をテキストで表示します。の設定（初期パラメータeTraceLevel, についてはParam [▶ 305] の設定も参照）に応じて、このメッセージはTC3 EventLogger 経由で出力されます。

ipErrorMessage	I_TcMessage	16#FFFFB8691299868
eSeverity	TCEVENTSEVERITY	Error
ipSourceInfo	I_TcSourceInfo	16#FFFFB86912996B0
nEventId	UDINT	8230
sEventClassName	STRING(255)	'TcMultiArrayEventClass'
sEventText	STRING(255)	'Input data has illegal shape.'

初期パラメータ

PLCの初期化フェーズで採用されるファンクションブロックの初期パラメータは、ノードstInitPars の下に格納されます。

調整可能なオーバーラップ（nOverlap パラメータ）を持つ全てのファンクションブロックは、計算された推奨値（nOverlap := cCM_OverlapRecommended の場合）は、PLCのログイン後に指定されたノードで読み取ることができます。これはその後、解析チェーンのデータバッファ（入出力形状）の調整に使用できます。このためにアプリケーションを起動する必要はありません。

結果データへのADSアクセス

各解析ファンクションブロックはサブノードpResultData を持ち、これは TwinCAT Target Browserでのみ表示されます。解析チェーンの様々なファンクションブロック間で送信された結果（マルチアレイ）は、ADS経由で読み取りできます。これにより、最初に結果をデータの受け口となるファンクションブロックを介してマルチアレイからPLC アレイに変換することなく、結果を簡単にグラフィック表示できます。



この機能を使用するには、バージョン1.4.3141以降のTC3 Scope Viewが必要です（バージョン3.1.4024.7以降のTwinCATで使用可能）。

Name	Type	Size	Subitems
[-] pResultData	POINTER TO ARRAY [1..<aResultDataLen>] OF ARRAY [1..0] OF LREAL	8	0
[-] pResultData^	ARRAY [1..<aResultDataLen>] OF ARRAY [1..0] OF LREAL	41000	5
[-] pResultData^[1]	ARRAY [1..0] OF LREAL	8200	1025
[-] pResultData^[2]	ARRAY [1..0] OF LREAL	8200	1025
[-] pResultData^[3]	ARRAY [1..0] OF LREAL	8200	1025
[-] pResultData^[4]	ARRAY [1..0] OF LREAL	8200	1025
[-] pResultData^[5]	ARRAY [1..0] OF LREAL	8200	1025
[-] aResultDataLen	ARRAY [0..1] OF UDINT	8	2
[-] aResultDataLen[0]	UDINT	4	0
[-] aResultDataLen[1]	UDINT	4	0
[-] hrResultDataSymbol	HRESULT	4	0

5.1 ファンクションブロック

以下のリストは、利用可能なファンクションブロックを見つけやすくするために、様々な基準に基づいて分類されています。

全てのCondition Monitoringライブラリ

信号を処理するためのアルゴリズムとツール：

信号処理	統計	分類	バッファ処理
FB_CMA_AnalyticSignal [▶ 83]	FB_CMA_HistArray [▶ 145]	FB_CMA_DiscreteClassification [▶ 113]	FB_CMA_BufferConverting [▶ 90]
FB_CMA_ArgSort [▶ 87]	FB_CMA_MinersRule [▶ 168]	FB_CMA_RainflowCounting [▶ 198]	FB_CMA_ComplexDataHandling [▶ 101]
FB_CMA_ComplexFFT [▶ 104]	FB_CMA_MomentCoefficients [▶ 172]	FB_CMA_VibrationAssessment [▶ 242]	FB_CMA_Sink [▶ 223]
FB_CMA_Correlation [▶ 108]	FB_CMA_Quantiles [▶ 193]	FB_CMA_WatchUpperThresholds [▶ 246]	FB_CMA_Source [▶ 230]
FB_CMA_CrestFactor [▶ 93]	FB_CMA_EmpiricalMean [▶ 123]		
FB_CMA_CrestFactorPlus [▶ 97]	FB_CMA_EmpiricalStandardDeviation [▶ 133]		
FB_CMA_Downsampling [▶ 117]	FB_CMA_EmpiricalSkew [▶ 128]		
FB_CMA_Envelope [▶ 137]	FB_CMA_EmpiricalExcess [▶ 118]		
FB_CMA_EnvelopeSpectrum [▶ 141]	FB_CMA_MovingMean [▶ 254]		
FB_CMA_InstantaneousFrequency [▶ 150]	FB_CMA_MovingStandardDeviation [▶ 258]		
FB_CMA_InstantaneousPhase [▶ 153]	FB_CMA_MovingSkew [▶ 262]		
FB_CMA_IntegratedRMS [▶ 157]	FB_CMA_MovingExcess [▶ 266]		
FB_CMA_MagnitudeSpectrum [▶ 161]			
FB_CMA_MultiBandRMS [▶ 177]			
FB_CMA_PowerCepstrum [▶ 185]			
FB_CMA_PowerSpectrum [▶ 189]			
FB_CMA_OrderPowerSpectrum [▶ 181]			

[FB_CMA_RealFFT](#)

[\[▶ 203\]](#)

[FB_CMA_RMS \[▶ 207\]](#)

[FB_CMA_SlidingDFT](#)

[\[▶ 210\]](#)

[FB_CMA_SparseSpectrum](#)

[\[▶ 214\]](#)

[FB_CMA_SpikeEnergySpectrum](#)

[\[▶ 219\]](#)

[FB_CMA_ZoomFFT](#)

[\[▶ 250\]](#)

信号処理をテーマ別に構造化

信号解析するためのアルゴリズム：

時間領域	周波数領域	時間/周波数領域	その他
FB_CMA_AnalyticSignal [▶ 83]	FB_CMA_ComplexFFT [▶ 104]	FB_CMA_InstantaneousFrequency [▶ 150]	FB_CMA_ArgSort [▶ 87]
FB_CMA_CrestFactor [▶ 93]	FB_CMA_EnvelopeSpectrum [▶ 141]	FB_CMA_SpikeEnergySpectrum [▶ 219]	FB_CMA_ComplexDataHandling [▶ 101]
FB_CMA_CrestFactorPlus [▶ 97]	FB_CMA_IntegratedRMS [▶ 157]		FB_CMA_Downsampling [▶ 117]
FB_CMA_Envelope [▶ 137]	FB_CMA_MagnitudeSpectrum [▶ 161]		FB_CMA_OrderPowerSpectrum [▶ 181]
FB_CMA_InstantaneousPhase [▶ 153]	FB_CMA_MultiBandRMS [▶ 177]		
FB_CMA_RMS [▶ 207]	FB_CMA_PowerSpectrum [▶ 189]		
FB_CMA_PowerCepstrum [▶ 185]	FB_CMA_RealFFT [▶ 203]		
FB_CMA_Correlation [▶ 108]	FB_CMA_SlidingDFT [▶ 210]		
FB_CMA_MovingMean [▶ 254]	FB_CMA_SparseSpectrum [▶ 214]		
FB_CMA_MovingStandardDeviation [▶ 258]	FB_CMA_ZoomFFT [▶ 250]		
FB_CMA_MovingSkew [▶ 262]			
FB_CMA_MovingExcess [▶ 266]			
FB_CMA_MomentCoefficients [▶ 172]			

5.1.1 FB_CMA_AnalyticSignal

時系列の解析信号を計算します。

解析信号は、入力された実信号に対応する複素数表現の信号であり、虚数部は実信号に対して90度位相シフトされます。虚数部は、入力された実数信号のヒルベルト変換により形成されます。時間連続表現では、実信号 $x(t)$ の解析信号 $x_{\text{analytic}}(t)$ は次式で記述されます。

$$x_{\text{analytic}}(t) = x(t) + i\mathcal{H}[x(t)]$$

このファンクションブロックは、周波数領域の離散ヒルベルト変換によって解析信号を計算します。結果は長さ $nWindowLength/2$ の複素数ベクトルです。

入力ベクトルは、ウェルチ法に基づき、50%オーバーラップした先行入力ベクトルと組み合わせられます。その後、ウィンドウ関数（使用するウィンドウタイプ：eCM_HannWindow）が乗算されます。その後、実入力値に対するFFTが適用されます。周波数領域では、信号にヒルベルト変換が適用されます。そこから複素数値の結果が得られます。その後、結果は逆FFTにより時間領域に変換されます。このようにして得られた時間信号は、オーバーラップ加算法により重畳して加算されます。信号合成中、再びウィンドウ処理が行われます（使用されるウィンドウタイプ：eCM_HannWindow）。その結果、ジャンプのない連続的な出力信号が得られます。

● ウィンドウの長さに注意

i $nWindowLength$ の値は $nFFT\_Length$ の値以下になるようにしてください。FFTの長さは、必要な周波数分解能に合わせるすることができます。一般的には、FFT長の約4/5の値がウィンドウ長としてよく使用されます。

$nFFT\_Length$ が $nWindowLength$ よりも大きい場合、FFTの周波数分解能が（したがって、戻り値ベクトルの長さも）増加します。長さの差は、フーリエ変換の前にゼロで満たされます。これは、より高い周波数分解能を得るためや、時間領域での逆変換による計算で円形エイリアシングを避けるために有用です。ただし、より高い周波数分解能にもかかわらず、結果にはそれ以上の情報は含まれていません。

メモリ特性

オーバーラップ加算法が使用されるため、計算には現在の入力バッファと直近に転送された2つのバッファが使用されます。

NaN（非数）の発生

入力ベクトルに1つ以上のNaN（非数）値が含まれる場合、出力セクタ全体がNaNで満たされます。詳細については別項NaN値 [▶ 71]を参照してください。

● NaN値の取り扱い

i NaN値につながる上記の状況を除外できない、または安全に無視できない場合、アプリケーションプログラムはこれらのエラー値を適切に処理できるようにしてください。

マルチチャンネル入力データ処理時の動作

複数のチャンネル（ $nChannels > 1$ ）を処理する場合、各チャンネルが異なる戻り値になる可能性があります。この場合、戻り値はファンクションブロックで個別に問い合わせできます。全てのチャンネルではなく、1つ以上のチャンネルの結果が許容されない場合、ファンクションブロックの値はeCM_InfRTime_AmbiguousChannelResultsに対応します。全てのチャンネルの結果が許容されない場合、ファンクションブロック上の値はeCM_ErrRTime_ErrornousChannelResultsに対応します。

全チャンネルの戻り値のリストは、GetChannelErrors()のメソッドで照会できます。

入力と出力

入力バッファと出力バッファは、以下の定義（入力/出力形状）のいずれかに該当します。変数パラメータは、ファンクションブロック入力stInitParsの一部です。

バージョン	入力バッファ（マルチアレイ入力ストリーム） 要素タイプ、次元数、次元サイズ	出力バッファ（マルチアレイ出力ストリーム） 要素タイプ、次元数、次元サイズ
標準バージョン ($nChannels = 1$)	LREAL, 1, $nWindowLength/2$	LCOMPLEX, 1, $nWindowLength/2$
マルチチャンネルバージョン ($nChannels > 1$)	LREAL, 2, $nChannels \times nWindowLength/2$	LCOMPLEX, 2, $nChannels \times nWindowLength/2$

入力パラメータ

このファンクションブロックの入力パラメータは初期化パラメータを表し、FBインスタンスの宣言で割り当て済みである必要があります。(または、Init()メソッド) 一度しか割り当てできません。実行中の変更はできません。

```
VAR_INPUT
    stInitPars      : ST_CM_AnalyticSignal_InitPars; // init parameter
    nOwnID          : UDINT; // ID for this FB instance
    aDestIDs        : ARRAY[1..cCMA_MaxDest] OF UDINT; // IDs of destinations for output
    nResultBuffers  : UDINT := 4; // number of MultiArrays which should be initialized for results (0 for no initialization)
    tTransferTimeout : LTIME := LTIME#500US; // timeout checking off during access to inter-task FIFOs
END_VAR
```

- **stInitPars:** ST_CM_AnalyticSignal_InitPars [▶ 280]型の初期化パラメータを持つファンクションブロック固有の構造体。パラメータは、上記の入出力バッファの定義と紐づいていなければなりません。
- **nOwnID:**一意のIDでファンクションブロックインスタンスを識別します。これは常に0より大きくなければなりません。この目的のためには、列挙を定義するアプローチが確立されています。
- **aDestIDs:**転送先のIDを指定することで、結果の転送先を定義します。出力バッファの定義（上記を参照）は、選択された各転送先の入力バッファの定義と紐づいていなければなりません。
- **nResultBuffers:**このファンクションブロックは、指定された数のマルチアレイバッファにより転送トレイストリームを初期化します。デフォルト値は4です。
- **tTransferTimeout:**内部マルチアレイ転送の同期タイムアウトの設定です。並列処理 [▶ 72]の項を参照してください。

出力パラメータ

```
VAR_OUTPUT
    bError          : BOOL; // TRUE if an error occurs. Reset by next method call.
    hrErrorCode      : HRESULT; // '< 0' = error; '> 0' = info; '0' = no error/info
    ipErrorMessage : I_TcMessage := fbErrorMessage; // Shows detailed information about occurred errors, warnings and more.
    nCntResults     : ULINT; // Counts outgoing results (MultiArrays were calculated and sent to transfer tray).
END_VAR
```

- **bError:**エラーが発生した場合の出力は、TRUEです。
- **hrErrorCode:**エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、エラーコード一覧 [▶ 367]に記載されています。
- **ipErrorMessage:**現在の戻り値に関するより詳細な情報を含みます。エラーの説明と情報 [▶ 81]の項を参照してください。この特別なインターフェースポイントは、常に有効かつ割り当て済みであるよう、内部的に保護されています。

メソッド

Call():

このメソッドは、現在の入力データにアルゴリズムを適用するために、サイクルごとに呼び出されます。メソッドが新しい結果もエラーも示さない場合、ファンクションブロックは入力データを待機します。これは、解析チェーンのプロセスにおける規則的な動作です。

- **戻り値:** エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、エラーコード一覧 [▶ 367]に記載されています。

```
METHOD Call : HRESULT
VAR_OUTPUT
    bNewResult      : BOOL; // TRUE every time when outgoing MultiArray was calculated and sent to transfer tray.
    bError          : BOOL; // TRUE if an error occurs.
    hrErrorCode      : HRESULT; // '< 0' = error; '> 0' = info; '0' = no error/info
END_VAR
```

- **bError:**エラーが発生した場合の出力は、TRUEです。
- **hrErrorCode:**エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、エラーコード一覧 [▶ 367]に記載されています。この出力はメソッドの戻り値と同じです。



タイムアウトが発生した場合、または結果を格納するマルチアレイバッファがない場合、入力データも結果データも失われません。次の呼び出しに転送されます。

Init():

このメソッドは通常、状態監視アプリケーションでは必要ありません。これはファンクションブロック初期化の代替手段を提供するものです。Init() メソッドは、PLCの初期化フェーズ中にのみ呼び出すことができます。実行時には使用できません。メソッドFB_init または、属性'call_after_init'の使用を参照してください（TwinCAT PLCリファレンスを参照）。さらに、これはファンクションブロックのカプセル化を容易にします。

Init()メソッドを使用して初期化を行う場合、ファンクションブロックインスタンスの入力パラメータを宣言で代入することはできません。

- **戻り値：**エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD Init : HRESULT
VAR_INPUT
    stInitPars      : ST_CM_AnalyticSignal_InitPars;    // init parameter
    nOwnID           : UDINT;                          // ID for this FB instance
    aDestIDs         : ARRAY[1..cCMA_MaxDest] OF UDINT; // IDs of destinations for output
    nResultBuffers   : UDINT := 4;                    // number of MultiArrays which should be initialized for results (0 for no initialization)
END_VAR
```

- **stInitPars:** [ST_CM_AnalyticSignal_InitPars \[▶ 280\]](#)型の初期化パラメータを持つファンクションブロック固有の構造体。パラメータは、上記の入出力バッファの定義と紐づいていなければなりません。
- **nOwnID:** 一意のIDでファンクションブロックインスタンスを識別します。これは常に0より大きくなければなりません。この目的のためには、列挙を定義するアプローチが確立されています。
- **aDestIDs:** 転送先のIDを指定することで、結果の転送先を定義します。出力バッファの定義（上記を参照）は、選択された各転送先の入力バッファの定義と紐づいていなければなりません。
- **nResultBuffers:** このファンクションブロックは、指定された数のマルチアレイバッファにより転送トレイストリームを初期化します。デフォルト値は4です。

ResetData() :

このメソッドは、既に追加された全てのデータレコードを削除します。ファンクションブロックのメモリ特性を参照してください。ResetData() の後、Call() メソッドが再度呼び出された場合、有効な結果を計算するために内部メモリを補充する必要があります。

- **戻り値：**エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD ResetData : HRESULT
VAR_INPUT
END_VAR
```

PassInputs() :

API PLC リファレンス [\[▶ 79\]](#)で説明されているとおり、FB_CMA_Source インスタンスが呼び出され、信号データがターゲットブロックに転送される限り、解析チェーンの以降の全てのファンクションブロックを周期的に呼び出す必要があります。

アルゴリズムを一定時間実行しない方が有用な場合があります。例えば、いくつかのアルゴリズムは、事前に教示や設定をした後でのみ実行すべきです。ファンクションブロックは周期的に呼び出す必要がありますが、ファンクションブロックに到着したデータが通信リング [\[▶ 72\]](#)内で転送されれば十分です。これは、Call() メソッドの代わりにPassInputs() メソッドを使用して行われます。アルゴリズム自体はここでは呼び出されず、したがって結果も計算されず、出力バッファも生成されません。

- **戻り値：**エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD PassInputs : HRESULT
VAR_INPUT
END_VAR
```

GetChannelErrors() :

このメソッドは、複数のチャンネル（ $nChannels > 1$ ）を処理する際に、チャンネル固有の戻り値のリストを照会できるようにします。呼び出しは、ファンクションブロックの戻り値が `eCM_InfRTTime_AmbiguousChannelResults` または `eCM_ErrRTTime_ErrornousChannelResults` のいずれかに該当する場合に有用です。

- **戻り値:** エラーコード一覧の読み込み処理に関する情報です。この値は、クエリが成功した場合は `TRUE` に、そうでない場合は `FALSE` に設定されます。

```
METHOD GetChannelErrors : BOOL
VAR_IN_OUT
    aChannelErrors : ARRAY[*] OF HRESULT;
END_VAR
```

- `aChannelErrors:HRESULT` 型の長さ `nChannels` のエラーリストです。

類似したファンクションブロック

ファンクションブロック `FB_CMA_Envelope` [▶ 137] は、時系列の包絡線を計算します。

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM, Tc3_CM_Base



CM 3.1 では一部の機能がすでに使用可能です。[互換性](#) [▶ 61] の項を参照してください。

5.1.2 FB_CMA_ArgSort

入力された引数をソートします。

入力された引数は、昇順または降順にソートされます。その際、入力データで発生したNaN値は出力データの末尾に移動されます。パワースペクトルの出力のような1次元配列が入力データストリームとして供給されます。出力データとして2次元配列が得られます。最初の次元は振幅、2番目の次元はこの振幅が入力配列内で見つかるインデックスを表します。スケーリング係数によって、例えばインデックスの代わりに周波数を直接表示できます。このファンクションブロックは、0ベースの配列で内部計算を行います。これは評価において考慮されなければなりません。

NaN（非数）の発生

常に有効なソート結果を得るために、入力データで発生し得るNaN値は、出力データの最後でソートされます。



NaN値の取り扱い

NaN値につながる上記の状況を除外できない、または安全に無視できない場合、アプリケーションプログラムはこれらのエラー値を適切に処理できるようにしてください。

マルチチャンネル入力データ処理時の動作

複数のチャンネル（ $nChannels > 1$ ）を処理する場合、各チャンネルが異なる戻り値になる可能性があります。この場合、戻り値はファンクションブロックで個別に問い合わせできます。全てのチャンネルではなく、1つ以上のチャンネルの結果が許容されない場合、ファンクションブロックの値は `eCM_InfRTTime_AmbiguousChannelResults` に対応します。全てのチャンネルの結果が許容されない場合、ファンクションブロック上の値は `eCM_ErrRTTime_ErrornousChannelResults` に対応します。

全チャンネルの戻り値のリストは、`GetChannelErrors()` のメソッドで照会できます。

入力と出力

入力バッファと出力バッファは、以下の定義（入力/出力形状）のいずれかに該当します。変数パラメータは、ファンクションブロック入力 `stInitPars` の一部です。

バージョン	入力バッファ（マルチアレイ入力ストリーム） 要素タイプ、次元数、次元サイズ	出力バッファ（マルチアレイ出力ストリーム） 要素タイプ、次元数、次元サイズ
標準バージョン (nChannels = 1)	LREAL, 1, nInLength	LREAL, 2, nInLength x 2
マルチチャンネルバージョン (nChannels > 1)	LREAL, 2, nChannels x nInLength	LREAL, 3, nChannels x nInLength x 2

```

VAR_INPUT
    stInitPars      : ST_CM_ArgSort_InitPars;           // init parameter
    nOwnID          : UDINT;                             // ID for this FB instance
    aDestIDs        : ARRAY[1..cCMA_MaxDest] OF UDINT;  // IDs of destinations for output
    nResultBuffers  : UDINT := 4;                       // number of MultiArrays which should be in
    itialized for results (0 for no initialization)
    tTransferTimeout : LTIME := LTIME#500US;           // timeout checking off during access to in
    ter-task FIFOs
END_VAR

```

入力パラメータ

このファンクションブロックの入力パラメータは初期化パラメータを表し、FBインスタンスの宣言で割り当て済みである必要があります。(または、Init()メソッド) 一度しか割り当てできません。実行中の変更はできません。

- **stInitPars:** ST_CM_ArgSort_InitPars [▶ 280]型の初期化パラメータを持つファンクションブロック固有の構造体。パラメータは、上記の入出力バッファの定義と紐づいていなければなりません。
- **nOwnID:**一意のIDでファンクションブロックインスタンスを識別します。これは常に0より大きくなければなりません。この目的のためには、列挙を定義するアプローチが確立されています。
- **aDestIDs:**転送先のIDを指定することで、結果の転送先を定義します。出力バッファの定義（上記を参照）は、選択された各転送先の入力バッファの定義と紐づいていなければなりません。
- **nResultBuffers:**このファンクションブロックは、指定された数のマルチアレイバッファにより転送トレイストリームを初期化します。デフォルト値は4です。
- **tTransferTimeout:**内部マルチアレイ転送の同期タイムアウトの設定です。[並列処理 \[▶ 72\]](#)の項を参照してください。

出力パラメータ

```

VAR_OUTPUT
    bError          : BOOL;                             // TRUE if an error occurs. Reset by next metho
    d call.
    hrErrorcode     : HRESULT;                          // '< 0' = error; '> 0' = info; '0' = no error/
    info
    ipErrorMessage : I_TcMessage := fbErrorMessage;    // Shows detailed information about occurred er
    rors, warnings and more.
    nCntResults     : ULINT;                             // Counts outgoing results (MultiArrays were ca
    lculated and sent to transfer tray).
END_VAR

```

- **bError:**エラーが発生した場合の出力は、TRUEです。
- **hrErrorcode:**エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。
- **ipErrorMessage:**現在の戻り値に関するより詳細な情報を含みます。[エラーの説明と情報 \[▶ 81\]](#)の項を参照してください。この特別なインターフェースポインタは、常に有効かつ割り当て済みであるよう、内部的に保護されています。

メソッド

METHOD Init : HRESULT

```

VAR_INPUT
    stInitPars      : ST_CM_ArgSort_InitPars;           // init parameter
    nOwnID          : UDINT;                             // ID for this FB instance
    aDestIDs        : ARRAY[1..cCMA_MaxDest] OF UDINT;  // IDs of destinations for output
    nResultBuffers  : UDINT := 4;                       // number of MultiArrays which should be ini
    tialized for results (0 for no initialization)
END_VAR

```

Call():

このメソッドは、現在の入力データにアルゴリズムを適用するために、サイクルごとに呼び出されます。メソッドが新しい結果もエラーも示さない場合、ファンクションブロックは入力データを待機します。これは、解析チェーンのプロセスにおける規則的な動作です。

- **戻り値**：エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

METHOD Call : HRESULT

```
VAR_OUTPUT
    bNewResult      : BOOL;          // TRUE every time when outgoing MultiArray was calculated and sent to transfer tray.
    bError          : BOOL;          // TRUE if an error occurs.
    hrErrorCode     : HRESULT;       // '< 0' = error; '> 0' = info; '0' = no error/info
END_VAR
```

- **bError**:エラーが発生した場合の出力は、TRUEです。
- **hrErrorCode**:エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。この出力はメソッドの戻り値と同じです。



タイムアウトが発生した場合、または結果を格納するマルチアレイバッファがない場合、入力データも結果データも失われません。次の呼び出しに転送されます。

Init():

このメソッドは通常、状態監視アプリケーションでは必要ありません。これはファンクションブロック初期化の代替手段を提供するものです。Init() メソッドは、PLCの初期化フェーズ中にのみ呼び出すことができます。実行時には使用できません。メソッドFB_init または、属性'call_after_init'の使用を参照してください（TwinCAT PLCリファレンスを参照）。さらに、これはファンクションブロックのカプセル化を容易にします。

Init()メソッドを使用して初期化を行う場合、ファンクションブロックインスタンスの入力パラメータを宣言で代入することはできません。

- **戻り値**：エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。
- **stInitPars**:ST_CM_ArgSort_InitPars [▶ 280]型の初期化パラメータを持つファンクションブロック固有の構造体。パラメータは、上記の入出力バッファの定義と紐づいていなければなりません。
- **nOwnID**:一意のIDでファンクションブロックインスタンスを識別します。これは常に0より大きくなければなりません。この目的のためには、列挙を定義するアプローチが確立されています。
- **aDestIDs**:転送先のIDを指定することで、結果の転送先を定義します。出力バッファの定義（上記を参照）は、選択された各転送先の入力バッファの定義と紐づいていなければなりません。
- **nResultBuffers**:このファンクションブロックは、指定された数のマルチアレイバッファにより転送ストリームを初期化します。デフォルト値は4です。
- **tTransferTimeout**:内部マルチアレイ転送の同期タイムアウトの設定です。[並列処理 \[▶ 72\]](#)の項を参照してください。

PassInputs() :

API PLC リファレンス [▶ 79]で説明されているとおり、FB_CMA_Source インスタンスが呼び出され、信号データがターゲットブロックに転送される限り、解析チェーンの以降の全てのファンクションブロックを周期的に呼び出す必要があります。

アルゴリズムを一定時間実行しない方が有用な場合があります。例えば、いくつかのアルゴリズムは、事前に教示や設定をした後でのみ実行すべきです。ファンクションブロックは周期的に呼び出す必要がありますが、ファンクションブロックに到着したデータが[通信リング \[▶ 72\]](#)内で転送されれば十分です。これは、Call() メソッドの代わりにPassInputs() メソッドを使用して行われます。アルゴリズム自体はここでは呼び出されず、したがって結果も計算されず、出力バッファも生成されません。

- **戻り値**：エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

METHOD PassInputs : HRESULT

```
VAR_INPUT
END_VAR
```

GetChannelErrors() :

このメソッドは、複数のチャンネル（ $nChannels > 1$ ）を処理する際に、チャンネル固有の戻り値のリストを照会できるようにします。呼び出しは、ファンクションブロックの戻り値が `eCM_InfRTTime_AmbiguousChannelResults` または `eCM_ErrRTTime_ErrornousChannelResults` のいずれかに該当する場合に有効です。

- **戻り値:** エラーコード一覧の読み込み処理に関する情報です。この値は、クエリが成功した場合は `TRUE` に、そうでない場合は `FALSE` に設定されます。

```
METHOD GetChannelErrors : BOOL
VAR_IN_OUT
    aChannelErrors : ARRAY[*] OF HRESULT;
END_VAR
```

- `aChannelErrors:HRESULT` 型の長さ `nChannels` のエラーリストです。

類似したファンクションブロック

なし

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM, Tc3_CM_Base



CM 3.1 では一部の機能がすでに使用可能です。[互換性 \[▶ 61\]](#)の項を参照してください。

5.1.3 FB_CMA_BufferConverting

マルチアレイから別のマルチアレイにデータをコピーします。

あるアルゴリズムのファンクションブロックの定義された入力バッファが、解析チェーンで先行するファンクションブロックの出力バッファと一致しない場合、この機能を使用して転送を行うことができます。それに応じて異なる次元数に変換できます。

また別の用途として、例えばスペクトルの関連する周波数領域のみを考慮するために、データのサブセットのみを追加の処理に使用することです。

入力と出力

入力バッファと出力バッファは、以下の定義（入力/出力形状）のいずれかに該当します。変数パラメータは、ファンクションブロック入力 `stInitPars` の一部です。

マルチアレイの所在	要素の型	次元	次元変数
出カストリーム	<code>eTypeCode</code>	<code>nDims</code>	<code>aDimSizes</code>

```
VAR_INPUT
    stInitPars : ST_MA_MultiArray_InitPars; // init parameter
    nOwnID : UDINT; // ID for this FB instance
    aDestIDs : ARRAY[1..cCMA_MaxDest] OF UDINT; // IDs of destinations for output
    nResultBuffers : UDINT := 4; // number of MultiArrays which should be initialized for results (0 for no initialization)
    tTransferTimeout : LTIME := LTIME#500US; // timeout checking off during access to inter-task FIFOs
END_VAR
```

入力パラメータ

このファンクションブロックの入力パラメータは初期化パラメータを表し、FBインスタンスの宣言で割り当て済みである必要があります。（または、`Init()`メソッド）一度しか割り当てできません。実行中の変更はできません。

- `stInitPars:ST_MA_MultiArray_InitPars [▶ 303]`型の初期化パラメータを持つファンクションブロック固有の構造体。パラメータは、上記の入出力バッファの定義と紐づいていなければなりません。

- **nOwnID**:一意のIDでファンクションブロックインスタンスを識別します。これは常に0より大きくなければなりません。この目的のためには、列挙を定義するアプローチが確立されています。
- **aDestIDs**:転送先のIDを指定することで、結果の転送先を定義します。出力バッファの定義（上記を参照）は、選択された各転送先の入力バッファの定義と紐づいていなければなりません。
- **nResultBuffers**:このファンクションブロックは、指定された数のマルチアレイバッファにより転送トレイストリームを初期化します。デフォルト値は4です。
- **tTransferTimeout**:内部マルチアレイ転送の同期タイムアウトの設定です。[並列処理 \[▶ 72\]](#)の項を参照してください。

出力パラメータ

```
VAR_OUTPUT
  bError      : BOOL;                // TRUE if an error occurs. Reset by next method call.
  hrErrorCode  : HRESULT;            // '< 0' = error; '> 0' = info; '0' = no error/info
  ipErrorMessage : I_TcMessage := fbErrorMessage; // Shows detailed information about occurred errors, warnings and more.
  nCntResults   : ULINT;             // Counts outgoing results (MultiArrays were calculated and sent to transfer tray).
END_VAR
```

- **bError**:エラーが発生した場合の出力は、TRUEです。
- **hrErrorCode**:エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。
- **ipErrorMessage**:現在の戻り値に関するより詳細な情報を含みます。[エラーの説明と情報 \[▶ 81\]](#)の項を参照してください。この特別なインターフェースポイントは、常に有効かつ割り当て済みであるよう、内部的に保護されています。

メソッド

Copy1D():

マルチアレイから別のマルチアレイに1次元データをコピーします。

```
METHOD Copy1D : HRESULT
VAR_INPUT
  nWorkDimIn      : UDINT;           // It designates the dimension in the input MultiArray being processed.
  nWorkDimOut     : UDINT;           // It designates the dimension in the output MultiArray being processed.
  nElements       : UDINT;           // optional: default:0->full copy; It designates the number of elements to be copied out of the MultiArray.
  pStartIndexIn   : POINTER TO UDINT; (* optional: default:0->internally handled as [0,0,..]; It designates the index of the first element to be copied out of the MultiArray.
                                     If allocated it must point to a onedimensional array of UDINT with so many elements as dimensions of the MultiArray. *)
  pStartIndexOut  : POINTER TO UDINT; (* optional: default:0->internally handled as [0,0,..]; It designates the index of the first MultiArray element to be copied.
                                     If allocated it must point to a onedimensional array of UDINT with so many elements as dimensions of the MultiArray. *)
  nOptionPars     : DWORD;           // option mask
END_VAR
VAR_OUTPUT
  bNewResult      : BOOL;            // TRUE every time when outgoing MultiArray was calculated and sent to transfer tray.
  bError          : BOOL;            // TRUE if an error occurs.
  hrErrorCode     : HRESULT;         // '< 0' = error; '> 0' = info; '0' = no error/info
END_VAR
```

- **nWorkDimIn**:入力マルチアレイが多次元の場合、コピーしたいデータの次元を選択できます。最初の次元は0（0ベース）になります。
- **nWorkDimOut**:出力マルチアレイが多次元の場合、データをコピーしたい次元を選択できます。最初の次元は0（0ベース）になります。
- **nElements**:マルチアレイ次元の全データをコピーするには、このパラメータを0に設定します。この場合、合計の数は内部で決定されます。あるいは、コピーする要素の数を指定することもできます。
- **bError**:エラーが発生した場合の出力は、TRUEです。

- **hrErrorCode**:エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。この出力はメソッドの戻り値と同じです。



タイムアウトが発生した場合、または結果を格納するマルチアレイバッファがない場合、入力データも結果データも失われません。次の呼び出しに転送されます。

Transpose() :

出力バッファが入力バッファの転置になるように、2つの次元を転置します。この操作は、1次元と2次元の入力バッファに対してのみ可能です。

```
METHOD Transpose : HRESULT
VAR_INPUT
    nOptionPars : DWORD;    // option mask
END_VAR
VAR_OUTPUT
    bNewResult : BOOL;      // TRUE every time when outgoing MultiArray was calculated and sent to
    transfer tray.
    bError : BOOL;          // TRUE if an error occurs.
    hrErrorCode : HRESULT;  // '< 0' = error; '> 0' = info; '0' = no error/info
END_VAR
```

- **bNewResult**:出力マルチアレイが計算され転送トレイに送信されるたびにTRUE が出力されます。
- **bError**:エラーが発生した場合の出力は、TRUEです。
- **hrErrorCode**:エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。この出力はメソッドの戻り値と同じです。



タイムアウトが発生した場合、または結果を格納するマルチアレイバッファがない場合、入力データも結果データも失われません。次の呼び出しに転送されます。

```
METHOD Init : HRESULT
VAR_INPUT
    stInitPars : ST_MA_MultiArray_InitPars;    // init parameter
    nOwnID : UDINT;                            // ID for this FB instance
    aDestIDs : ARRAY[1..CMA_MaxDest] OF UDINT; // IDs of destinations for output
    nResultBuffers : UDINT := 4;               // number of MultiArrays which should be initialized for results (0 for no initialization)
END_VAR
```

Init():

このメソッドは通常、状態監視アプリケーションでは必要ありません。これはファンクションブロック初期化の代替手段を提供するものです。Init() メソッドは、PLCの初期化フェーズ中にのみ呼び出すことができます。実行時には使用できません。メソッドFB_init または、属性'call_after_init' の使用を参照してください（TwinCAT PLCリファレンスを参照）。さらに、これはファンクションブロックのカプセル化を容易にします。

Init()メソッドを使用して初期化を行う場合、ファンクションブロックインスタンスの入力パラメータを宣言で代入することはできません。

- **戻り値**: エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。
- **stInitPars**:ST_MA_MultiArray_InitPars [▶ 303]型の初期化パラメータを持つファンクションブロック固有の構造体。パラメータは、上記の入出力バッファの定義と紐づいていなければなりません。
- **nOwnID**:一意のIDでファンクションブロックインスタンスを識別します。これは常に0より大きくなければなりません。この目的のためには、列挙を定義するアプローチが確立されています。
- **aDestIDs**:転送先のIDを指定することで、結果の転送先を定義します。出力バッファの定義（上記を参照）は、選択された各転送先の入力バッファの定義と紐づいていなければなりません。
- **nResultBuffers**:このファンクションブロックは、指定された数のマルチアレイバッファにより転送トレイストリームを初期化します。デフォルト値は4です。

PassInputs() :

API PLC リファレンス [▶ 79]で説明されているとおり、FB_CMA_Source インスタンスが呼び出され、信号データがターゲットブロックに転送される限り、解析チェーンの以降の全てのファンクションブロックを周期的に呼び出す必要があります。

アルゴリズムを一定時間実行しない方が有用な場合があります。例えば、いくつかのアルゴリズムは、事前に教示や設定をした後でのみ実行すべきです。ファンクションブロックは周期的に呼び出す必要がありますが、ファンクションブロックに到着したデータが通信リング [▶ 72]内で転送されれば十分です。これは、Call() メソッドの代わりにPassInputs() メソッドを使用して行われます。アルゴリズム自体はここでは呼び出されず、したがって結果も計算されず、出力バッファも生成されません。

- **戻り値**：エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD PassInputs : HRESULT
VAR_INPUT
END_VAR
```

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM, Tc3_CM_Base, Tc3_MultiArray

5.1.4 FB_CMA_CrestFactor

各チャンネルの時系列データに対して、シングルチャンネルおよびマルチチャンネルのクレストファクタを計算します。

これは、信号のピーク値とRMS値との比として定義されます。

$$C_{dB} = 20 \log_{10} \left(\frac{\max\{|x(t)|\}}{\text{rms}\{x(t)\}} \right)$$

クレストファクタは対数単位のデシベルで計算されます。例えば、正弦波のクレストファクタは3.01dB (=1,414) です。これにより、ローラーベアリングの状態などについて結論を出すことができます。一般的に、クレストファクタはローラーベアリングの損傷開始時に増加し、損傷が進行するにつれて再び減少します。

ログ演算による数値範囲エラーを避けるため、計算された各値は、対数を適用する前に最小値と比較され、それよりも小さい場合は、その最小値で置き換えられます。

クレストファクタはピーク値とRMS値の比率で定義されるため、この結果は個々の最大値に強く影響されやすく、予期せぬ結果を招く可能性があります。

メモリ特性

RMS値の計算のために、nBufferLength の時系列の値がチャンネル/サブチャンネルごとに内部に保存されます。入力バッファサイズが小さい場合の呼び出しでは、転送できる値の数は少なくなります。この場合、バッファの内容はシフトされ、信号長は新たに転送された値の対応する数で満たされます。入力バッファが内部バッファより大きい場合、内部バッファは計算のために最新の値で満たされます。

NaN（非数）の発生

入力ベクトルに1つ以上のNaN（非数）値が含まれる場合、出力セクタ全体がNaNで満たされます。詳細については別項NaN値 [▶ 71]を参照してください。

内部バッファが新しい有効値で完全に満たされるまで、出力ベクトルはNaNで満たされます。

● NaN値の取り扱い



NaN値につながる上記の状況を除外できない、または安全に無視できない場合、アプリケーションプログラムはこれらのエラー値を適切に処理できるようにしてください。

マルチチャンネル入力データ処理時の動作

複数のチャンネル ($nChannels > 1$) を処理する場合、各チャンネルが異なる戻り値になる可能性があります。この場合、戻り値はファンクションブロックで個別に問い合わせできます。全てのチャンネルではなく、1つ以上のチャンネルの結果が許容されない場合、ファンクションブロックの値は `eCM_InfRTime_AmbiguousChannelResults` に対応します。全てのチャンネルの結果が許容されない場合、ファンクションブロック上の値は `eCM_ErrRTime_ErrornousChannelResults` に対応します。

全チャンネルの戻り値のリストは、`GetChannelErrors()` のメソッドで照会できます。

複数のサブチャンネル ($nSubChannels > 0$) を処理する場合、入出力データのフォーマットには特に注意が必要です。入力データが上流のファンクションブロックによるマルチチャンネルの結果で構成されている場合、 $nChannels$ の値を採用する必要があります。この場合、パラメータ $nSubChannels$ を使用して追加の設定を行います。

例：マルチチャンネルスペクトル (`FB_CMA_MagnitudeSpectrum` など) の周波数チャンネルの統計的考察 (`FB_CMA_Quantiles` など) では、 $nChannels$ の値は、入力信号の数と同じである必要があります。サブチャンネルの数 $nSubChannels$ は、スペクトルの長さに対応します。

サンプルの実装

実装用のサンプルは以下のリンクから入手できます：[クレストファクタ \[▶ 341\]](#)。

入力と出力

入力バッファと出力バッファは、以下の定義（入力/出力形状）のいずれかに該当します。変数パラメータは、ファンクションブロック入力 `stInitPars` の一部です。

バージョン	入力バッファ（マルチアレイ入力ストリーム） 要素タイプ、次元数、次元サイズ	出力バッファ（マルチアレイ出力ストリーム） 要素タイプ、次元数、次元サイズ
複数データセット用標準バージョン ($nSubChannels = 0$)	LREAL, 2, $nChannels \times$ 指定なし*	LREAL, 1, $nChannels$
複数データセット用マルチチャンネルバージョン ($nSubChannels > 0$)	LREAL, 3, $nChannels \times nSubChannels \times$ 指定なし*	LREAL, 2, $nChannels \times nSubChannels$

*: この次元の長さは任意に選択できるため、アプリケーションや先行のアルゴリズムの出力バッファに合わせるができます。

```

VAR_INPUT
    stInitPars      : ST_CM_CrestFactor_InitPars;           // init parameter
    nOwnID          : UDINT;                               // ID for this FB instance
    aDestIDs        : ARRAY[1..cCMA_MaxDest] OF UDINT;     // IDs of destinations for output
    nResultBuffers  : UDINT := 4;                           // number of MultiArrays which should be initialized for results (0 for no initialization)
    tTransferTimeout : LTIME := LTIME#500US;               // timeout checking off during access to inter-task FIFOs
END_VAR

```

入力パラメータ

このファンクションブロックの入力パラメータは初期化パラメータを表し、FBインスタンスの宣言で割り当て済みである必要があります。（または、`Init()` メソッド）一度しか割り当てできません。実行中の変更はできません。

- `stInitPars`: `ST_CM_CrestFactor_InitPars` [▶ 281] 型の初期化パラメータを持つファンクションブロック固有の構造体。パラメータは、上記の入出力バッファの定義と紐づいていなければなりません。
- `nOwnID`: 一意のIDでファンクションブロックインスタンスを識別します。これは常に0より大きくなければなりません。この目的のためには、列挙を定義するアプローチが確立されています。
- `aDestIDs`: 転送先のIDを指定することで、結果の転送先を定義します。出力バッファの定義（上記を参照）は、選択された各転送先の入力バッファの定義と紐づいていなければなりません。
- `nResultBuffers`: このファンクションブロックは、指定された数のマルチアレイバッファにより転送トレイストリームを初期化します。デフォルト値は4です。

- **tTransferTimeout**:内部マルチアレイ転送の同期タイムアウトの設定です。[並列処理 \[▶ 72\]](#)の項を参照してください。

出力パラメータ

```
VAR_OUTPUT
    bError      : BOOL;                // TRUE if an error occurs. Reset by next method call.
    hrErrorCode  : HRESULT;            // '< 0' = error; '> 0' = info; '0' = no error/info
    ipErrorMessage : I_TcMessage := fbErrorMessage; // Shows detailed information about occurred errors, warnings and more.
    nCntResults   : ULINT;             // Counts outgoing results (MultiArrays were calculated and sent to transfer tray).
END_VAR
```

- **bError**:エラーが発生した場合の出力は、TRUEです。
- **hrErrorCode**:エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。
- **ipErrorMessage**:現在の戻り値に関するより詳細な情報を含みます。[エラーの説明と情報 \[▶ 81\]](#)の項を参照してください。この特別なインターフェースポイントは、常に有効かつ割り当て済みであるよう、内部的に保護されています。

メソッド

```
METHOD Init : HRESULT
VAR_INPUT
    stInitPars      : ST_CM_CrestFactor_InitPars;    // init parameter
    nOwnID           : UDINT;                        // ID for this FB instance
    aDestIDs         : ARRAY[1..cCMA_MaxDest] OF UDINT; // IDs of destinations for output
    nResultBuffers   : UDINT := 4;                  // number of MultiArrays which should be initialized for results (0 for no initialization)
END_VAR
```

Call():

このメソッドは、現在の入力データにアルゴリズムを適用するために、サイクルごとに呼び出されます。メソッドが新しい結果もエラーも示さない場合、ファンクションブロックは入力データを待機します。これは、解析チェーンのプロセスにおける規則的な動作です。

- **戻り値**: エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD Call : HRESULT
VAR_OUTPUT
    bNewResult      : BOOL;                // TRUE every time when outgoing MultiArray was calculated and sent to transfer tray.
    bError           : BOOL;                // TRUE if an error occurs.
    hrErrorCode      : HRESULT;            // '< 0' = error; '> 0' = info; '0' = no error/info
END_VAR
```

- **bError**:エラーが発生した場合の出力は、TRUEです。
- **hrErrorCode**:エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。この出力はメソッドの戻り値と同じです。



タイムアウトが発生した場合、または結果を格納するマルチアレイバッファがない場合、入力データも結果データも失われません。次の呼び出しに転送されます。

Init():

このメソッドは通常、状態監視アプリケーションでは必要ありません。これはファンクションブロック初期化の代替手段を提供するものです。Init() メソッドは、PLCの初期化フェーズ中にのみ呼び出すことができます。実行時には使用できません。メソッドFB_init または、属性'call_after_init'の使用を参照してください（TwinCAT PLCリファレンスを参照）。さらに、これはファンクションブロックのカプセル化を容易にします。

Init()メソッドを使用して初期化を行う場合、ファンクションブロックインスタンスの入力パラメータを宣言で代入することはできません。

- **戻り値**：エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。
- **stInitPars**: ST_CM_CrestFactor_InitPars [▶ 281]型の初期化パラメータを持つファンクションブロック固有の構造体。パラメータは、上記の入出力バッファの定義と紐づいていなければなりません。
- **nOwnID**: 一意のIDでファンクションブロックインスタンスを識別します。これは常に0より大きくなければなりません。この目的のためには、列挙を定義するアプローチが確立されています。
- **aDestIDs**: 転送先のIDを指定することで、結果の転送先を定義します。出力バッファの定義（上記を参照）は、選択された各転送先の入力バッファの定義と紐づいていなければなりません。
- **nResultBuffers**: このファンクションブロックは、指定された数のマルチアレイバッファにより転送ストリームを初期化します。デフォルト値は4です。

ResetData() :

このメソッドは、既に追加された全てのデータレコードを削除します。ファンクションブロックのメモリ特性を参照してください。ResetData() の後、Call() メソッドが再度呼び出された場合、有効な結果を計算するために内部メモリを補充する必要があります。

- **戻り値**：エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD ResetData : HRESULT
VAR_INPUT
END_VAR
```

PassInputs() :

API PLC リファレンス [▶ 79]で説明されているとおり、FB_CMA_Source インスタンスが呼び出され、信号データがターゲットブロックに転送される限り、解析チェーンの以降の全てのファンクションブロックを周期的に呼び出す必要があります。

アルゴリズムを一定時間実行しない方が有用な場合があります。例えば、いくつかのアルゴリズムは、事前に教示や設定をした後でのみ実行すべきです。ファンクションブロックは周期的に呼び出す必要がありますが、ファンクションブロックに到着したデータが[通信リング \[▶ 72\]](#)内で転送されれば十分です。これは、Call() メソッドの代わりにPassInputs() メソッドを使用して行われます。アルゴリズム自体はここでは呼び出されず、したがって結果も計算されず、出力バッファも生成されません。

- **戻り値**：エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD PassInputs : HRESULT
VAR_INPUT
END_VAR
```

GetChannelErrors() :

このメソッドは、複数のチャンネル（nChannels > 1）を処理する際に、チャンネル固有の戻り値のリストを照会できるようにします。呼び出しは、ファンクションブロックの戻り値が eCM_InfRTTime_AmbiguousChannelResults または eCM_ErrRTTime_ErrornousChannelResults のいずれかに該当する場合に有効です。

- **戻り値**: エラーコード一覧の読み込み処理に関する情報です。この値は、クエリが成功した場合は TRUE に、そうでない場合は FALSE に設定されます。

```
METHOD GetChannelErrors : BOOL
VAR_IN_OUT
aChannelErrors : ARRAY[*] OF HRESULT;
END_VAR
```

- **aChannelErrors**: HRESULT 型の長さnChannelsのエラーリストです。

類似したファンクションブロック

[FB_CMA_HistArray \[▶ 145\]](#) ファンクションブロックは、入力値分布のヒストグラムを計算します。

[FB_CMA_Quantiles \[▶ 193\]](#) ブロックは、経験分布の分位を計算し、これにより外れ値の周波数を評価します。

[FB_CMA_MomentCoefficients \[▶ 172\]](#) ブロックは、外れ値の影響を受けにくい指標として、信号のピーク性を計測する尖度を提供します。

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM, Tc3_CM_Base



CM 3.1 では一部の機能がすでに使用可能です。[互換性 \[▶ 61\]](#)の項を参照してください。

5.1.5 FB_CMA_CrestFactorPlus

各チャンネルの時系列データに対して、シングルチャンネルおよびマルチチャンネルのクレストファクタプラスを計算します。

これは、信号のピーク値の大きさ、RMS値、および2つの変数の比の線形結合として定義されます：

$$C^+ = c_1 \max\{|x(t)|\} + c_2 \text{rms}\{x(t)\} + c_3 \frac{\max\{|x(t)|\}}{\text{rms}\{x(t)\}}, \quad C_{\text{dB}}^+ = 20 \log_{10}(C^+), \quad c_1, c_2, c_3 \in \mathbb{R}$$

定数のデフォルト値は、クレストファクタプラスが、パラメータを個別に設定しない場合のクレストファクタに対応するように選択されます。

この値は、クレストファクタとは対照的に、適切に設定することでローラーベアリングの損傷が進行する過程で増加します。したがって、ベアリングの状態に関する事前の知識は必要ありません。比較対象：[FB_CMA_CrestFactor \[▶ 93\]](#)

ログ演算による数値範囲エラーを避けるため、計算された各値は、対数を適用する前に最小値と比較され、それよりも小さい場合は、その最小値で置き換えられます。

クレストファクタは、RMS平均値に対するピーク値の比率に依存するため、個々の最大値によって結果が大きく左右される可能性があります。これは予期せぬ結果を招く可能性があります。

メモリ特性

RMS値の計算のために、nBufferLength の時系列の値がチャンネル/サブチャンネルごとに内部に保存されます。入力バッファサイズが小さい場合の呼び出しでは、転送できる値の数は少なくなります。この場合、バッファの内容はシフトされ、信号長は新たに転送された値の対応する数で満たされます。入力バッファが内部バッファより大きい場合、内部バッファは計算のために最新の値で満たされます。

NaN（非数）の発生

入力ベクトルに1つ以上のNaN（非数）値が含まれる場合、出力セクタ全体がNaNで満たされます。詳細については別項[NaN値 \[▶ 71\]](#)を参照してください。

内部バッファが新しい有効値で完全に満たされるまで、出力ベクトルはNaNで満たされます。



NaN値の取り扱い

NaN値につながる上記の状況を除外できない、または安全に無視できない場合、アプリケーションプログラムはこれらのエラー値を適切に処理できるようにしてください。

マルチチャンネル入力データ処理時の動作

複数のチャンネル（nChannels > 1）を処理する場合、各チャンネルが異なる戻り値になる可能性があります。この場合、戻り値はファンクションブロックで個別に問い合わせできます。全てのチャンネルではなく、1つ以上のチャンネルの結果が許容されない場合、ファンクションブロックの値はeCM_InfRTime_AmbiguousChannelResultsに対応します。全てのチャンネルの結果が許容されない場合、ファンクションブロック上の値はeCM_ErrRTime_ErrornousChannelResultsに対応します。

全チャンネルの戻り値のリストは、GetChannelErrors()のメソッドで照会できます。

複数のサブチャンネル($nSubChannels > 0$)を処理する場合、入出力データのフォーマットには特に注意が必要です。入力データが上流のファンクションブロックによるマルチチャンネルの結果で構成されている場合、 $nChannels$ の値を採用する必要があります。この場合、パラメータ $nSubChannels$ を使用して追加の設定を行います。

例：マルチチャンネルスペクトル（FB_CMA_MagnitudeSpectrumなど）の周波数チャンネルの統計的考察（FB_CMA_Quantilesなど）では、 $nChannels$ の値は、入力信号の数と同じである必要があります。サブチャンネルの数 $nSubChannels$ は、スペクトルの長さに対応します。

入力と出力

入力バッファと出力バッファは、以下の定義（入力/出力形状）のいずれかに該当します。変数パラメータは、ファンクションブロック入力 $stInitPars$ の一部です。

バージョン	入力バッファ（マルチアレイ入力ストリーム） 要素タイプ、次元数、次元サイズ	出力バッファ（マルチアレイ出力ストリーム） 要素タイプ、次元数、次元サイズ
複数データセット用標準バージョン ($nSubChannels = 0$)	LREAL, 2, $nChannels \times$ 指定なし*	LREAL, 1, $nChannels$
複数データセット用マルチチャンネルバージョン ($nSubChannels > 0$)	LREAL, 3, $nChannels \times nSubChannels \times$ 指定なし*	LREAL, 2, $nChannels \times nSubChannels$

*:この次元の長さは任意に選択できるため、アプリケーションや先行のアルゴリズムの出力バッファに合わせる必要があります。

入力パラメータ

このファンクションブロックの入力パラメータは初期化パラメータを表し、FBインスタンスの宣言で割り当て済みである必要があります。(または、Init()メソッド) 一度しか割り当てできません。実行中の変更はできません。

```
VAR_INPUT
    stInitPars      : ST_CM_CrestFactorPlus_InitPars;      // init parameter
    nOwnID          : UDINT;                               // ID for this FB instance
    aDestIDs        : ARRAY[1..cCMA_MaxDest] OF UDINT;     // IDs of destinations for output
    nResultBuffers  : UDINT := 4;                          // number of MultiArrays which should be initialized for results (0 for no initialization)
    tTransferTimeout : LTIME := LTIME#500US;              // timeout checking off during access to inter-task FIFOs
END_VAR
```

- **stInitPars:** ST_CM_CrestFactorPlus_InitPars [▶ 281] 型の初期化パラメータを持つファンクションブロック固有の構造体。パラメータは、上記の入出力バッファの定義と紐づいていなければなりません。
- **nOwnID:**一意のIDでファンクションブロックインスタンスを識別します。これは常に0より大きくなければなりません。この目的のためには、列挙を定義するアプローチが確立されています。
- **aDestIDs:**転送先のIDを指定することで、結果の転送先を定義します。出力バッファの定義（上記を参照）は、選択された各転送先の入力バッファの定義と紐づいていなければなりません。
- **nResultBuffers:**このファンクションブロックは、指定された数のマルチアレイバッファにより転送トレーストリームを初期化します。デフォルト値は4です。
- **tTransferTimeout:**内部マルチアレイ転送の同期タイムアウトの設定です。[並列処理 \[▶ 72\]](#)の項を参照してください。

出力パラメータ

```
VAR_OUTPUT
    bError          : BOOL;                               // TRUE if an error occurs. Reset by next method call.
    hrErrorCode      : HRESULT;                           // '< 0' = error; '> 0' = info; '0' = no error/info
    ipErrorMessage  : I_TcMessage := fbErrorMessage;     // Shows detailed information about occurred errors, warnings and more.
```

```

    nCntResults      : ULINT;                // Counts outgoing results (MultiArrays were ca
lculated and sent to transfer tray).
END_VAR

```

- **bError:**エラーが発生した場合の出力は、TRUEです。
- **hrErrorCode:**エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。
- **ipErrorMessage:**現在の戻り値に関するより詳細な情報を含みます。[エラーの説明と情報 \[▶ 81\]](#)の項を参照してください。この特別なインターフェイスポインタは、常に有効かつ割り当て済みであるよう、内部的に保護されています。

メソッド

Call():

このメソッドは、現在の入力データにアルゴリズムを適用するために、サイクルごとに呼び出されます。メソッドが新しい結果もエラーも示さない場合、ファンクションブロックは入力データを待機します。これは、解析チェーンのプロセスにおける規則的な動作です。

- **戻り値:**エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```

METHOD Call : HRESULT
VAR_OUTPUT
    bNewResult      : BOOL;                // TRUE every time when outgoing MultiArray was calculated and sent t
o transfer tray.
    bError          : BOOL;                // TRUE if an error occurs.
    hrErrorCode     : HRESULT;            // '< 0' = error; '> 0' = info; '0' = no error/info
END_VAR

```

- **bError:**エラーが発生した場合の出力は、TRUEです。
- **hrErrorCode:**エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。この出力はメソッドの戻り値と同じです。



タイムアウトが発生した場合、または結果を格納するマルチアレイバッファがない場合、入力データも結果データも失われません。次の呼び出しに転送されます。

Configure() :

このメソッドを呼び出すことで、ピーク値の大きさ、RMS値、および2つの変数の比の重みを再設定できます。対応するPLC配列は以下のように定義してください。

- **戻り値:**エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧](#)に記載されています。

```

METHOD Configure : HRESULT
VAR_INPUT
    pArg          : POINTER TO LREAL;    // pointer to 2-dimensional array (LREAL) of arguments
    nArgSize      : UDINT;                // size of arguments buffer in bytes
END_VAR

```

入力バッファは、以下の定義（入力形状）のいずれかに紐づいています。変数パラメータは、ファンクションブロック入力stInitParsの一部です。チャンネルおよびサブチャンネルごとに設定される3つのパラメータは、c1;c2;c3です。

バージョン	入力バッファ（マルチアレイ入カストリーム）要素タイプ、次元数、次元サイズ
全てのチャンネルおよびサブチャンネルで同一設定	LREAL, 1, 3
チャンネル別の設定 (nSubChannels = 0)	LREAL, 2, nChannels x 3
サブチャンネル別の設定 (nSubChannels > 0)	LREAL, 2, nSubChannels x 3
チャンネルおよびサブチャンネル別の設定 (nSubChannels > 0)	LREAL, 3, nChannels x nSubChannels x 3

Init():

このメソッドは通常、状態監視アプリケーションでは必要ありません。これはファンクションブロック初期化の代替手段を提供するものです。Init() メソッドは、PLCの初期化フェーズ中にのみ呼び出すことができます。実行時には使用できません。メソッドFB_init または、属性'call_after_init' の使用を参照してください（TwinCAT PLCリファレンスを参照）。さらに、これはファンクションブロックのカプセル化を容易にします。

Init()メソッドを使用して初期化を行う場合、ファンクションブロックインスタンスの入力パラメータを宣言で代入することはできません。

- **戻り値**：エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD Init : HRESULT
VAR_INPUT
    stInitPars      : ST_CM_CrestFactorPlus_InitPars; // init parameter
    nOwnID           : UDINT; // ID for this FB instance
    aDestIDs         : ARRAY[1..cCMA_MaxDest] OF UDINT; // IDs of destinations for output
    nResultBuffers   : UDINT := 4; // number of MultiArrays which should be initialized for results (0 for no initialization)
END_VAR
```

- **stInitPars:ST_CM_CrestFactorPlus_InitPars [▶ 281]** 型の初期化パラメータを持つファンクションブロック固有の構造体。パラメータは、上記の入出力バッファの定義と紐づいていなければなりません。
- **nOwnID**:一意のIDでファンクションブロックインスタンスを識別します。これは常に0より大きくなければなりません。この目的のためには、列挙を定義するアプローチが確立されています。
- **aDestIDs**:転送先のIDを指定することで、結果の転送先を定義します。出力バッファの定義（上記を参照）は、選択された各転送先の入力バッファの定義と紐づいていなければなりません。
- **nResultBuffers**:このファンクションブロックは、指定された数のマルチアレイバッファにより転送トレイストリームを初期化します。デフォルト値は4です。

ResetData() :

このメソッドは、既に追加された全てのデータレコードを削除します。ファンクションブロックのメモリ特性を参照してください。ResetData() の後、Call() メソッドが再度呼び出された場合、有効な結果を計算するために内部メモリを補充する必要があります。

- **戻り値**：エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD ResetData : HRESULT
VAR_INPUT
END_VAR
```

PassInputs() :

API PLC リファレンス [▶ 79]で説明されているとおり、FB_CMA_Source インスタンスが呼び出され、信号データがターゲットブロックに転送される限り、解析チェーンの以降の全てのファンクションブロックを周期的に呼び出す必要があります。

アルゴリズムを一定時間実行しない方が有用な場合があります。例えば、いくつかのアルゴリズムは、事前に教示や設定をした後でのみ実行すべきです。ファンクションブロックは周期的に呼び出す必要がありますが、ファンクションブロックに到着したデータが通信リング [▶ 72]内で転送されれば十分です。これは、Call() メソッドの代わりにPassInputs() メソッドを使用して行われます。アルゴリズム自体はここでは呼び出されず、したがって結果も計算されず、出力バッファも生成されません。

- **戻り値**：エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD PassInputs : HRESULT
VAR_INPUT
END_VAR
```

GetChannelErrors() :

このメソッドは、複数のチャンネル（ $nChannels > 1$ ）を処理する際に、チャンネル固有の戻り値のリストを照会できるようにします。呼び出しは、ファンクションブロックの戻り値が `eCM_InfRTTime_AmbiguousChannelResults` または `eCM_ErrRTTime_ErrornousChannelResults` のいずれかに該当する場合に有用です。

- **戻り値:** エラーコード一覧の読み込み処理に関する情報です。この値は、クエリが成功した場合は `TRUE` に、そうでない場合は `FALSE` に設定されます。

```
METHOD GetChannelErrors : BOOL
VAR_IN_OUT
    aChannelErrors : ARRAY[*] OF HRESULT;
END_VAR
```

- `aChannelErrors:HRESULT` 型の長さ `nChannels` のエラーリストです。

類似したファンクションブロック

ファンクションブロック `FB_CMA_CrestFactor` [▶ 93] は、クレストファクタの標準的な形式を計算します。

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM, Tc3_CM_Base



CM 3.1 では一部の機能がすでに使用可能です。[互換性](#) [▶ 61] の項を参照してください。

5.1.6 FB_CMA_ComplexDataHandling

複雑な入力データを実際の出カデータに変換します。

ファンクションブロック `FB_CMA_ComplexDataHandling` は、複素数入力データから実数部または虚数部を抽出したり、絶対値や位相を計算します。

NaN（非数）の発生

入力ベクトルに1つ以上のNaN（非数）値が含まれる場合、出力ベクトルの関連する要素はNaNで満たされます。[NaN値](#) [▶ 71] の詳細については別項を参照してください。



NaN値の取り扱い

NaN値につながる上記の状況を除外できない、または安全に無視できない場合、アプリケーションプログラムはこれらのエラー値を適切に処理できるようにしてください。

マルチチャンネル入力データ処理時の動作

複数のチャンネル（ $nChannels > 1$ ）を処理する場合、各チャンネルが異なる戻り値になる可能性があります。この場合、戻り値はファンクションブロックで個別に問い合わせできます。全てのチャンネルではなく、1つ以上のチャンネルの結果が許容されない場合、ファンクションブロックの値は `eCM_InfRTTime_AmbiguousChannelResults` に対応します。全てのチャンネルの結果が許容されない場合、ファンクションブロック上の値は `eCM_ErrRTTime_ErrornousChannelResults` に対応します。

全チャンネルの戻り値のリストは、`GetChannelErrors()` のメソッドで照会できます。

入力と出力

入力バッファと出力バッファは、以下の定義（入力/出力形状）のいずれかに該当します。変数パラメータは、ファンクションブロック入力 `stInitPars` の一部です。

バージョン	入力バッファ（マルチアレイ 入力ストリーム） 要素タイプ、次元数、次元サイズ	出力バッファ（マルチアレイ 出力ストリーム） 要素タイプ、次元数、次元サイズ
標準バージョン (nChannels = 1)	LCOMPLEX, 1, nInLength	LREAL, 1, nInLength
マルチチャンネルバージョンのスペクトル (nChannels > 1)	LCOMPLEX, 2, nChannels x nInLength	LREAL, 2, nChannels x nInLength

入力パラメータ

このファンクションブロックの入力パラメータは初期化パラメータを表し、FBインスタンスの宣言で割り当て済みである必要があります。(または、Init()メソッド) 一度しか割り当てできません。実行中の変更はできません。

```
VAR_INPUT
    stInitPars      : ST_CM_ComplexDataHandling_InitPars;    // init parameter
    nOwnID           : UDINT;                                  // ID for this FB instance
    aDestIDs         : ARRAY[1..cCMA_MaxDest] OF UDINT;       // IDs of destinations for output
    nResultBuffers   : UDINT := 4;                             // number of MultiArrays which should
// be initialized for results (0 for no initialization)
    tTransferTimeout : LTIME := LTIME#500US;                  // timeout checking off during access
// to inter-task FIFOs
END_VAR
```

- **stInitPars:** [ST_CM_ComplexDataHandling_InitPars \[▶ 282\]](#) 型の初期化パラメータを持つファンクションブロック固有の構造体。パラメータは、上記の入出力バッファの定義と紐づいていなければなりません。
- **nOwnID:** 一意のIDでファンクションブロックインスタンスを識別します。これは常に0より大きくなければなりません。この目的のためには、列挙を定義するアプローチが確立されています。
- **aDestIDs:** 転送先のIDを指定することで、結果の転送先を定義します。出力バッファの定義（上記を参照）は、選択された各転送先の入力バッファの定義と紐づいていなければなりません。
- **nResultBuffers:** このファンクションブロックは、指定された数のマルチアレイバッファにより転送トレイストリームを初期化します。デフォルト値は4です。
- **tTransferTimeout:** 内部マルチアレイ転送の同期タイムアウトの設定です。[並列処理 \[▶ 72\]](#)の項を参照してください。

出力パラメータ

```
VAR_OUTPUT
    bError           : BOOL;                                  // TRUE if an error occurs. Reset by next method call.
    hrErrorCode       : HRESULT;                               // '< 0' = error; '> 0' = info; '0' = no error/info
    ipErrorMessage   : I_TcMessage := fbErrorMessage;       // Shows detailed information about occurred errors, warnings and more.
    nCntResults       : ULINT;                                 // Counts outgoing results (MultiArrays were calculated and sent to transfer tray).
END_VAR
```

- **bError:** エラーが発生した場合の出力は、TRUEです。
- **hrErrorCode:** エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。
- **ipErrorMessage:** 現在の戻り値に関するより詳細な情報を含みます。[エラーの説明と情報 \[▶ 81\]](#)の項を参照してください。この特別なインターフェイスポイントは、常に有効かつ割り当て済みであるよう、内部的に保護されています。

メソッド

Call():

このメソッドは、現在の入力データにアルゴリズムを適用するために、サイクルごとに呼び出されます。メソッドが新しい結果もエラーも示さない場合、ファンクションブロックは入力データを待機します。これは、解析チェーンのプロセスにおける規則的な動作です。

- **戻り値** : エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD Call : HRESULT
VAR_OUTPUT
  bNewResult : BOOL;          // TRUE every time when outgoing MultiArray was calculated and sent to transfer tray.
  bError      : BOOL;          // TRUE if an error occurs.
  hrErrorCode : HRESULT;       // '< 0' = error; '> 0' = info; '0' = no error/info
END_VAR
```

- **bError**:エラーが発生した場合の出力は、TRUEです。
- **hrErrorCode**:エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。この出力はメソッドの戻り値と同じです。



タイムアウトが発生した場合、または結果を格納するマルチアレイバッファがない場合、入力データも結果データも失われません。次の呼び出しに転送されます。

Configure() :

このメソッドを呼び出すことで、複雑な入力データを処理するメソッドを実行時に変更できます。データ型 [E_CM_ComplexDataHandling \[▶ 272\]](#)を使用して、メソッドを選択できます。

対応するPLC配列は以下のように定義してください。Configure()メソッドは、異なる引数セットを持つ新しい設定にも使用できます。

- **戻り値** : エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は[エラーコードの概要 \[▶ 367\]](#)に記載されています。

```
METHOD Configure : HRESULT
VAR_INPUT
  pArg : PVOID;          // pointer to array of arguments
  nArgSize : UDINT;       // size of arguments buffer in bytes
END_VAR
```

入力バッファは、以下の定義（入力形状）のいずれかに紐づいています。変数パラメータは、ファンクションブロック入力 [stInitPars](#) の一部です。

バージョン	入力バッファ（マルチアレイ入力ストリーム）要素タイプ、次元数、次元サイズ
全てのチャンネルで同一設定	UDINT, 1, 1
チャンネル別の設定 (nChannels > 1)	UDINT, 2, nChannels x 1

Init():

このメソッドは通常、状態監視アプリケーションでは必要ありません。これはファンクションブロック初期化の代替手段を提供するものです。Init() メソッドは、PLCの初期化フェーズ中にのみ呼び出すことができます。実行時には使用できません。メソッド [FB_init](#) または、属性 ['call_after_init'](#) の使用を参照してください（[TwinCAT PLCリファレンス](#)を参照）。さらに、これはファンクションブロックのカプセル化を容易にします。

Init()メソッドを使用して初期化を行う場合、ファンクションブロックインスタンスの入力パラメータを宣言で代入することはできません。

- **戻り値** : エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD Init : HRESULT
VAR_INPUT
  stInitPars : ST_CM_ComplexDataHandling_InitPars; // init parameter
  nOwnID     : UDINT;                               // ID for this FB instance
  aDestIDs   : ARRAY[1..CMA_MaxDest] OF UDINT;     // IDs of destinations for output
  nResultBuffers : UDINT := 4;                     // number of MultiArrays which should be initialized for results (0 for no initialization)
END_VAR
```

- **stInitPars:** ST_CM_ComplexDataHandling_InitPars [▶ 282] 型の初期化パラメータを持つファンクションブロック固有の構造体。パラメータは、上記の入出力バッファの定義と紐づいていなければなりません。
- **nOwnID:** 一意のIDでファンクションブロックインスタンスを識別します。これは常に0より大きくなければなりません。この目的のためには、列挙を定義するアプローチが確立されています。
- **aDestIDs:** 転送先のIDを指定することで、結果の転送先を定義します。出力バッファの定義（上記を参照）は、選択された各転送先の入力バッファの定義と紐づいていなければなりません。
- **nResultBuffers:** このファンクションブロックは、指定された数のマルチアレイバッファにより転送トレイストリームを初期化します。デフォルト値は4です。

PassInputs() :

API PLC リファレンス [▶ 79]で説明されているとおり、FB_CMA_Source インスタンスが呼び出され、信号データがターゲットブロックに転送される限り、解析チェーンの以降の全てのファンクションブロックを周期的に呼び出す必要があります。

アルゴリズムを一定時間実行しない方が有用な場合があります。例えば、いくつかのアルゴリズムは、事前に教示や設定をした後でのみ実行すべきです。ファンクションブロックは周期的に呼び出す必要がありますが、ファンクションブロックに到着したデータが通信リング [▶ 72]内で転送されれば十分です。これは、Call() メソッドの代わりにPassInputs() メソッドを使用して行われます。アルゴリズム自体はここでは呼び出されず、したがって結果も計算されず、出力バッファも生成されません。

- **戻り値:** エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD PassInputs : HRESULT
VAR_INPUT
END_VAR
```

GetChannelErrors() :

このメソッドは、複数のチャンネル（nChannels > 1）を処理する際に、チャンネル固有の戻り値のリストを照会できるようにします。呼び出しは、ファンクションブロックの戻り値が eCM_InfRTTime_AmbiguousChannelResults または eCM_ErrRTTime_ErrornousChannelResults のいずれかに該当する場合に有用です。

- **戻り値:** エラーコード一覧の読み込み処理に関する情報です。この値は、クエリが成功した場合は TRUE に、そうでない場合は FALSE に設定されます。

```
METHOD GetChannelErrors : BOOL
VAR_IN_OUT
aChannelErrors : ARRAY[*] OF HRESULT;
END_VAR
```

- **aChannelErrors:** HRESULT 型の長さnChannelsのエラーリストです。

類似したファンクションブロック

ファンクションブロックFB_CMA_BufferConverting [▶ 90]は、多次元の入力データから低次元のデータを転置または抽出することができます。

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM, Tc3_CM_Base

5.1.7 FB_CMA_ComplexFFT

複素数値の入力信号に対する高速フーリエ変換（FFT）を計算します。

ファンクションブロックFB_CMA_ComplexFFTは、ファンクションブロックに存在する複素数値の入力信号x[n]のフーリエ変換を計算します。高性能なFFTアルゴリズムが使用されています。順変換と逆変換の両方を計算できます。設定は、入力stInitPars で設定できます（入力と出力を参照）。

DFT表記による順フーリエ変換の定義：

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-i2\pi nk/N}$$

DFT表記による逆フーリエ変換の定義：

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] e^{i2\pi nk/N}$$

入力信号成分の最高周波数は、エイリアシング効果为了避免のために、入力信号のサンプリングレートの最大でも半分程度にするべきです。

FFTは、周期的に継続する信号の変換として定義されます。これは、周期的に継続する信号が正確に連続していない、つまり最初と最後が一致しない場合、即座にジャンプと判定される可能性があります。ファンクションブロック [FB_CMA_PowerSpectrum \[▶ 189\]](#) および [FB_CMA_MagnitudeSpectrum \[▶ 161\]](#) は、ウィンドウ関数を乗算したオーバーラップセクションの解析により、これらの困難を回避します。

スケーリング

計測によるスペクトルの定量的な評価のために、計算されたスペクトルは、直流成分、すなわち出力の最初の配列要素については $1/nFFT\_Length$ 、他の全ての要素については $2/nFFT\_Length$ 、重み付けされます。ファンクションブロックは、順変換と逆変換の間でスケーリングを行い、連続した順変換と逆変換の場合に、元の入力信号が出力で再び直接得られるようにします。

NaN（非数）の発生

入力ベクトルに1つ以上のNaN（非数）値が含まれる場合、出力セクタ全体がNaNで満たされます。詳細については別項 [NaN値 \[▶ 71\]](#) を参照してください。

● NaN値の取り扱い



NaN値につながる上記の状況を除外できない、または安全に無視できない場合、アプリケーションプログラムはこれらのエラー値を適切に処理できるようにしてください。

マルチチャンネル入力データ処理時の動作

複数のチャンネル（ $nChannels > 1$ ）を処理する場合、各チャンネルが異なる戻り値になる可能性があります。この場合、戻り値はファンクションブロックで個別に問い合わせできます。全てのチャンネルではなく、1つ以上のチャンネルの結果が許容されない場合、ファンクションブロックの値は `eCM_InfRTime_AmbiguousChannelResults` に対応します。全てのチャンネルの結果が許容されない場合、ファンクションブロック上の値は `eCM_ErrRTime_ErrornousChannelResults` に対応します。

全チャンネルの戻り値のリストは、`GetChannelErrors()` のメソッドで照会できます。

サンプルの実装

実装用のサンプルは以下のリンクから入手できます：[複素数値入力信号によるFFT \[▶ 309\]](#)。

入力と出力

入力バッファと出力バッファは、以下の定義（入力/出力形状）のいずれかに該当します。変数パラメータは、ファンクションブロック入力 `stInitPars` の一部です。

バージョン	入力バッファ（マルチアレイ入力ストリーム） 要素タイプ、次元数、次元サイズ	出力バッファ（マルチアレイ出力ストリーム） 要素タイプ、次元数、次元サイズ
標準バージョン ($nChannels = 1$)	LCOMPLEX, 1, <code>nFFT_Length</code>	LCOMPLEX, 1, <code>nFFT_Length</code>
マルチチャンネルバージョン ($nChannels > 1$)	LCOMPLEX, 2, <code>nChannels x nFFT_Length</code>	LCOMPLEX, 2, <code>nChannels x nFFT_Length</code>

```

VAR_INPUT
    stInitPars      : ST_CM_ComplexFFT_InitPars;          // init parameter
    nOwnID          : UDINT;                              // ID for this FB instance
    aDestIDs        : ARRAY[1..cCMA_MaxDest] OF UDINT;    // IDs of destinations for output
    nResultBuffers  : UDINT := 4;                        // number of MultiArrays which should be ini
tialized for results (0 for no initialization)
    tTransferTimeout : LTIME := LTIME#500US;              // timeout checking off during access to int
er-task FIFOs
END_VAR

```

入力パラメータ

このファンクションブロックの入力パラメータは初期化パラメータを表し、FBインスタンスの宣言で割り当て済みである必要があります。(または、Init()メソッド) 一度しか割り当てできません。実行中の変更はできません。

- **stInitPars:** ST_CM_ComplexFFT_InitPars [▶ 282]型の初期化パラメータを持つファンクションブロック固有の構造体。パラメータは、上記の入出力バッファの定義と紐づいていなければなりません。
- **nOwnID:**一意のIDでファンクションブロックインスタンスを識別します。これは常に0より大きくなければなりません。この目的のためには、列挙を定義するアプローチが確立されています。
- **aDestIDs:**転送先のIDを指定することで、結果の転送先を定義します。出力バッファの定義（上記を参照）は、選択された各転送先の入力バッファの定義と紐づいていなければなりません。
- **nResultBuffers:**このファンクションブロックは、指定された数のマルチアレイバッファにより転送トレイストリームを初期化します。デフォルト値は4です。
- **tTransferTimeout:**内部マルチアレイ転送の同期タイムアウトの設定です。並列処理 [▶ 72]の項を参照してください。

出力パラメータ

```

VAR_OUTPUT
    bError          : BOOL;                              // TRUE if an error occurs. Reset by next metho
d call.
    hrErrorCode      : HRESULT;                          // '< 0' = error; '> 0' = info; '0' = no error/
info
    ipErrorMessage  : I_TcMessage := fbErrorMessage;    // Shows detailed information about occurred er
rors, warnings and more.
    nCntResults      : ULINT;                            // Counts outgoing results (MultiArrays were ca
lculated and sent to transfer tray).
END_VAR

```

- **bError:**エラーが発生した場合の出力は、TRUEです。
- **hrErrorCode:**エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、エラーコード一覧 [▶ 367]に記載されています。
- **ipErrorMessage:**現在の戻り値に関するより詳細な情報を含みます。エラーの説明と情報 [▶ 81]の項を参照してください。この特別なインターフェースポイントは、常に有効かつ割り当て済みであるよう、内部的に保護されています。

メソッド

```

METHOD Init : HRESULT
VAR_INPUT
    stInitPars      : ST_CM_ComplexFFT_InitPars;          // init parameter
    nOwnID          : UDINT;                              // ID for this FB instance
    aDestIDs        : ARRAY[1..cCMA_MaxDest] OF UDINT;    // IDs of destinations for output
    nResultBuffers  : UDINT := 4;                        // number of MultiArrays which should be initi
alized for results (0 for no initialization)
END_VAR

```

Call():

このメソッドは、現在の入力データにアルゴリズムを適用するために、サイクルごとに呼び出されます。メソッドが新しい結果もエラーも示さない場合、ファンクションブロックは入力データを待機します。これは、解析チェーンのプロセスにおける規則的な動作です。

- **戻り値:** エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、エラーコード一覧 [▶ 367]に記載されています。

```

METHOD Call : HRESULT
VAR_OUTPUT
    bNewResult      : BOOL;                              // TRUE every time when outgoing MultiArray was calculated and sent t

```



```
o transfer tray.
  bError      : BOOL;          // TRUE if an error occurs.
  hrErrorCode  : HRESULT;      // '< 0' = error; '> 0' = info; '0' = no error/info
END_VAR
```

- **bError**:エラーが発生した場合の出力は、TRUEです。
- **hrErrorCode**:エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。この出力はメソッドの戻り値と同じです。



タイムアウトが発生した場合、または結果を格納するマルチアレイバッファがない場合、入力データも結果データも失われません。次の呼び出しに転送されます。

Init():

このメソッドは通常、状態監視アプリケーションでは必要ありません。これはファンクションブロック初期化の代替手段を提供するものです。Init() メソッドは、PLCの初期化フェーズ中にのみ呼び出すことができます。実行時には使用できません。メソッドFB_init または、属性'call_after_init' の使用を参照してください (TwinCAT PLCリファレンスを参照)。さらに、これはファンクションブロックのカプセル化を容易にします。

Init()メソッドを使用して初期化を行う場合、ファンクションブロックインスタンスの入力パラメータを宣言で代入することはできません。

- **戻り値**: エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。
- **stInitPars**: [ST_CM_ComplexFFT_InitPars \[▶ 282\]](#)型の初期化パラメータを持つファンクションブロック固有の構造体。パラメータは、上記の入出力バッファの定義と紐づいていなければなりません。
- **nOwnID**:一意のIDでファンクションブロックインスタンスを識別します。これは常に0より大きくなければなりません。この目的のためには、列挙を定義するアプローチが確立されています。
- **aDestIDs**:転送先のIDを指定することで、結果の転送先を定義します。出力バッファの定義（上記を参照）は、選択された各転送先の入力バッファの定義と紐づいていなければなりません。
- **nResultBuffers**:このファンクションブロックは、指定された数のマルチアレイバッファにより転送トレイストリームを初期化します。デフォルト値は4です。

PassInputs() :

[API PLC リファレンス \[▶ 79\]](#)で説明されているとおり、FB_CMA_Source インスタンスが呼び出され、信号データがターゲットブロックに転送される限り、解析チェーンの以降の全てのファンクションブロックを周期的に呼び出す必要があります。

アルゴリズムを一定時間実行しない方が有用な場合があります。例えば、いくつかのアルゴリズムは、事前に教示や設定をした後でのみ実行すべきです。ファンクションブロックは周期的に呼び出す必要がありますが、ファンクションブロックに到着したデータが[通信リング \[▶ 72\]](#)内で転送されれば十分です。これは、Call() メソッドの代わりにPassInputs() メソッドを使用して行われます。アルゴリズム自体はここでは呼び出されず、したがって結果も計算されず、出力バッファも生成されません。

- **戻り値**: エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD PassInputs : HRESULT
VAR_INPUT
END_VAR
```

GetChannelErrors() :

このメソッドは、複数のチャンネル (nChannels > 1) を処理する際に、チャンネル固有の戻り値のリストを照会できるようにします。呼び出しは、ファンクションブロックの戻り値が `eCM_InfRTTime_AmbiguousChannelResults` または `eCM_ErrRTTime_ErrornousChannelResults` のいずれかに該当する場合に有効です。

- **戻り値**: エラーコード一覧の読み込み処理に関する情報です。この値は、クエリが成功した場合は TRUE に、そうでない場合は FALSE に設定されます。

```

METHOD GetChannelErrors : BOOL
VAR_IN_OUT
    aChannelErrors : ARRAY[*] OF HRESULT;
END_VAR

```

- aChannelErrors:HRESULT 型の長さnChannelsのエラーリストです。

類似したファンクションブロック

ファンクションブロックFB_CMA_RealFFT [▶ 203]は、実数値信号のフーリエ変換を計算します。

ファンクションブロックFB_CMA_PowerSpectrum [▶ 189]は、連続時間信号のパワースペクトルを計算します。

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM, Tc3_CM_Base



CM 3.1 では一部の機能がすでに使用可能です。[互換性 \[▶ 61\]](#)の項を参照してください。

5.1.8 FB_CMA_Correlation

シングルチャンネルおよびマルチチャンネルの実数値時系列データの相関関数を計算します。

FB_CMA_Correlationファンクションブロックは、実数値時系列データの相関関数を計算します。

2つの信号の相関関数は、異なる時間シフトにおける両者の類似性を表します。以下では、時間離散変位を m とします。

入力データは、1つ（または複数）のテスト信号 $x[n]$ と1つ（または複数）の参照信号 $y[n]$ のタプル形式で構成されます。入力データは、ファンクションブロックFB_CMA_SourcePaired [▶ 236] を介して解析チェーンに導入されます。単一の基準信号を指定した場合、テスト信号の全チャンネルが同じ基準信号と相関することに注意してください。逆に、複数の基準信号をペアで指定する場合、チャンネル i の基準信号はチャンネル i のテスト信号と相関します。nChannels のチャンネル数もそれに合わせてください。

相関関数には多くの応用が考えられます。例えば、あるプロセスの特定の段階における基準信号を記録し、それをその後の同じプロセス段階の繰り返しと継続的に比較することが可能です。類似性の尺度（相関関数の最大値）は、プロセス段階を監視するために使用できます。同様に、相関関数は時間シフトを決定するための良いツールです。ここで探しているのは最大値の値ではなく、変位軸 m 上の位置です。これは、例えば、複数のマイクを使って漏水箇所を特定する際に、信号の実行時間差を判定するために使用できます。一般に、自己相関、すなわち信号とそれ自身との相関は、ノイズの多い信号の認識に利用できます。ノイズがランダムで、目的の信号が決定論的な信号である場合、自己相関の結果には $m=0$ においてのみノイズ成分が含まれ、決定論的な信号が明確に現れます。

サンプリング定理：離散フーリエ変換と同様に、サンプリング定理も同じように適用されます。自己相関関数の時間的に連続した曲線は、サンプリングレートが信号内の最高周波数の少なくとも2倍であれば、離散的な時系列から再構成することができます。

相関関数の計算は、設定可能な以下の定義のいずれかに基づいて行われます：

基本

$$C_{xy}[m] = \sum_{n=0}^{N-1} x[n-m] y[n]$$

規格化

$$\hat{C}_{xy}[m] = \frac{1}{N} \sum_{n=0}^{N-1} x[n-m] y[n]$$

共分散

$$cov_{xy}[m] = \left(\frac{1}{N} \sum_{n=0}^{N-1} x[n-m] y[n] \right) - \left(\frac{1}{N} \sum_{n=0}^{N-1} x[n-m] \right) \left(\frac{1}{N} \sum_{n=0}^{N-1} y[n] \right)$$

共分散（ベッセル補正）

$$\widetilde{cov}_{xy}[m] = \frac{N}{N-1} cov_{xy}[m]$$

ピアソン

$$\rho_{xy}[m] = \frac{cov_{xy}[m]}{\sigma_x[m] \sigma_y[m]}, \quad \sigma_x^2[m] = cov_{xx}[m]$$

上記の定義は、E_CM_CorrelationMode [▶ 272] 型の初期パラメータeCorrelationMode を介して選択します。信号の類似性を定量的に評価する場合は、ピアソンを推奨します。逆に言えば、シフト時間を求める際には、標準化は関係しません。

計算は、E_CM_WindowMode [▶ 274] 型のeWindowMode で設定した時間枠を介して行われます。3つの選択肢があります：スライディングウィンドウ、固定ウィンドウ、および前回ResetData()以降に収集された全データに基づく計算です。

設定

初期化パラメータは、計算バリエーションとウィンドウを定義するために使用されます。Configure() メソッドを使用して、各チャンネルごとに調整できます。

NaN（非数）の発生

入力ベクトルに1つ以上のNaN（非数）値が含まれる場合、出力セクタ全体がNaNで満たされます。詳細については別項NaN値 [▶ 71]を参照してください。

● NaN値の取り扱い

NaN値につながる上記の状況を除外できない、または安全に無視できない場合、アプリケーションプログラムはこれらのエラー値を適切に処理できるようにしてください。

マルチチャンネル入力データ処理時の動作

複数のチャンネル（nChannels > 1）を処理する場合、各チャンネルが異なる戻り値になる可能性があります。この場合、戻り値はファンクションブロックで個別に問い合わせできます。全てのチャンネルではなく、1つ以上のチャンネルの結果が許容されない場合、ファンクションブロックの値はeCM_InfRTTime_AmbiguousChannelResultsに対応します。全てのチャンネルの結果が許容されない場合、ファンクションブロック上の値はeCM_ErrRTTime_ErrornousChannelResults に対応します。

全チャンネルの戻り値のリストは、GetChannelErrors()のメソッドで照会できます。

サンプルの実装

実装用のサンプルは以下のリンクから入手できます：[相関関数 \[▶ 362\]](#)

入力と出力

入力バッファと出力バッファは、以下の定義（入力/出力形状）のいずれかに該当します。変数パラメータは、ファンクションブロック入力stInitPars の一部です。シングルチャンネル（nChannels = 1）の処理ルールは以下の通り：

マルチアレイの所在	要素の型	次元	次元変数
入カストリームA	LREAL	1	指定なし*
入カストリームB	LREAL	1	指定なし*
出カストリーム	LREAL	1	1+ (nPositiveShift+nNegativeShift)/nStepsize

マルチチャンネル（nChannels > 1）の処理ルールは以下の通り：

マルチアレイの所在	要素の型	次元	次元変数
入カストリームA	LREAL	2	nChannels x 指定なし*
入カストリームB	LREAL	1	指定なし*
出カストリーム	LREAL	2	nChannels x (1+ (nPositiveShift+nNegativeShift)/nStepsize)

または相関係数のペアワイズ計算のために使用されます：

マルチアレイの所在	要素の型	次元	次元変数
入カストリームA	LREAL	2	nChannels x 指定なし*
入カストリームB	LREAL	2	nChannels x 指定なし*
出カストリーム	LREAL	2	nChannels x (1+ (nPositiveShift+nNegativeShift)/nStepsize)

```

VAR_INPUT
    stInitPars      : ST_CM_Correlation_InitPars;           // init parameter
    nOwnID           : UDINT;                               // ID for this FB instance
    aDestIDs         : ARRAY[1..cCMA_MaxDest] OF UDINT;     // IDs of destinations for output
    nResultBuffers   : UDINT := 4;                          // number of MultiArrays which should be
// initialized for results (0 for no initialization)
    tTransferTimeout : LTIME := LTIME#500US;               // timeout checking off during access to
// inter-task FIFOs
END_VAR

```

入カパラメータ

このファンクションブロックの入カパラメータは初期化パラメータを表し、FBインスタンスの宣言で割り当て済みである必要があります。(または、Init()メソッド) 一度しか割り当てできません。実行中の変更はできません。

- **stInitPars:** ST_CM_Correlation_InitPars [▶ 282] 型の初期化パラメータを持つファンクションブロック固有の構造体。パラメータは、上記の入出力バッファの定義と紐づいていなければなりません。
- **nOwnID:** 一意のIDでファンクションブロックインスタンスを識別します。これは常に0より大きくなければなりません。この目的のためには、列挙を定義するアプローチが確立されています。
- **aDestIDs:** 転送先のIDを指定することで、結果の転送先を定義します。出力バッファの定義（上記を参照）は、選択された各転送先の入力バッファの定義と紐づいていなければなりません。
- **nResultBuffers:** このファンクションブロックは、指定された数のマルチアレイバッファにより転送トレイストリームを初期化します。デフォルト値は4です。
- **tTransferTimeout:** 内部マルチアレイ転送の同期タイムアウトの設定です。[並列処理 \[▶ 72\]](#)の項を参照してください。

出カパラメータ

```

VAR_OUTPUT
    bError           : BOOL;                               // TRUE if an error occurs. Reset by next method call.
    hrErrorCode       : HRESULT;                           // '< 0' = error; '> 0' = info; '0' = no error/info
    ipErrorMessage   : I_TcMessage := fbErrorMessage;     // Shows detailed information about occurred errors, warnings and more.
    nCntResults       : ULINT;                             // Counts outgoing results (MultiArrays were calculated and sent to transfer tray).
END_VAR

```

- **bError:** エラーが発生した場合の出力は、TRUEです。
- **hrErrorCode:** エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。
- **ipErrorMessage:** 現在の戻り値に関するより詳細な情報を含みます。[エラーの説明と情報 \[▶ 81\]](#)の項を参照してください。この特別なインターフェースポインタは、常に有効かつ割り当て済みであるよう、内部的に保護されています。

メソッド

Call():

このメソッドは、現在の入力データにアルゴリズムを適用するために、サイクルごとに呼び出されます。メソッドが新しい結果もエラーも示さない場合、ファンクションブロックは入力データを待機します。これは、解析チェーンのプロセスにおける規則的な動作です。

- **戻り値**：エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD Call : HRESULT
VAR_OUTPUT
    bNewResult    : BOOL;          // TRUE every time when outgoing MultiArray was calculated and sent to transfer tray.
    bError        : BOOL;          // TRUE if an error occurs.
    hrErrorCode    : HRESULT;      // '< 0' = error; '> 0' = info; '0' = no error/info
END_VAR
```

- **bError**:エラーが発生した場合の出力は、TRUEです。
- **hrErrorCode**:エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。この出力はメソッドの戻り値と同じです。



タイムアウトが発生した場合、または結果を格納するマルチアレイバッファがない場合、入力データも結果データも失われません。次の呼び出しに転送されます。

Configure() :

このメソッドを呼び出すことで、相関係数（初期パラメータeCorrelationMode）のスケーリングや、基礎となる計算ウィンドウ（初期パラメータeWindowMode）を実行時に変更できます。

対応するPLC配列は以下のように定義してください。Configure()メソッドは、異なる引数セットを持つ新しい設定にも使用できます。

- **戻り値**：エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は[エラーコードの概要 \[▶ 367\]](#)に記載されています。

```
METHOD Configure : HRESULT
VAR_INPUT
    pArg          : PVOID;        // pointer to array of arguments
    nArgSize       : UDINT;        // size of arguments buffer in bytes
END_VAR
```

入力バッファは、以下の定義（入力形状）のいずれかに紐づいています。変数パラメータは、ファンクションブロック入力stInitPars の一部です。

バージョン	入力バッファ（マルチアレイ 入カストリーム） 要素タイプ、次元数、次元サイズ
全てのチャンネルで同一設定	UDINT, 1, 2
チャンネル別の設定 (nChannels > 1)	UDINT, 2, nChannels x 2

```
METHOD Init : HRESULT
VAR_INPUT
    stInitPars      : ST_CM_Correlation_InitPars;          // init parameter
    nOwnID          : UDINT;                                // ID for this FB instance
    aDestIDs        : ARRAY[1..cCMA_MaxDest] OF UDINT;     // IDs of destinations for output
    nResultBuffers  : UDINT := 4;                          // number of MultiArrays which should be initialized for results (0 for no initialization)
END_VAR
```

Init():

このメソッドは通常、状態監視アプリケーションでは必要ありません。これはファンクションブロック初期化の代替手段を提供するものです。Init() メソッドは、PLCの初期化フェーズ中にのみ呼び出すことができます。実行時には使用できません。メソッドFB_init または、属性'call_after_init' の使用を参照してください（TwinCAT PLCリファレンスを参照）。さらに、これはファンクションブロックのカプセル化を容易にします。

Init()メソッドを使用して初期化を行う場合、ファンクションブロックインスタンスの入力パラメータを宣言で代入することはできません。

- **戻り値**：エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。
- **stInitPars:ST CM Correlation InitPars [▶ 282]** 型の初期化パラメータを持つファンクションブロック固有の構造体。パラメータは、上記の入出力バッファの定義と紐づいていなければなりません。
- **nOwnID**:一意のIDでファンクションブロックインスタンスを識別します。これは常に0より大きくなければなりません。この目的のためには、列挙を定義するアプローチが確立されています。
- **aDestIDs**:転送先のIDを指定することで、結果の転送先を定義します。出力バッファの定義（上記を参照）は、選択された各転送先の入力バッファの定義と紐づいていなければなりません。
- **nResultBuffers**:このファンクションブロックは、指定された数のマルチアレイバッファにより転送トレイストリームを初期化します。デフォルト値は4です。

ResetData() :

このメソッドは、既に追加された全てのデータレコードを削除します。ファンクションブロックのメモリ特性を参照してください。ResetData() の後、Call() メソッドが再度呼び出された場合、有効な結果を計算するために内部メモリを補充する必要があります。

- **戻り値**：エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD ResetData : HRESULT
VAR_INPUT
END_VAR
```

PassInputs() :

API PLC リファレンス [▶ 79]で説明されているとおり、FB_CMA_Source インスタンスが呼び出され、信号データがターゲットブロックに転送される限り、解析チェーンの以降の全てのファンクションブロックを周期的に呼び出す必要があります。

アルゴリズムを一定時間実行しない方が有用な場合があります。例えば、いくつかのアルゴリズムは、事前に教示や設定をした後でのみ実行すべきです。ファンクションブロックは周期的に呼び出す必要がありますが、ファンクションブロックに到着したデータが[通信リング \[▶ 72\]](#)内で転送されれば十分です。これは、Call() メソッドの代わりにPassInputs() メソッドを使用して行われます。アルゴリズム自体はここでは呼び出されず、したがって結果も計算されず、出力バッファも生成されません。

- **戻り値**：エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD PassInputs : HRESULT
VAR_INPUT
END_VAR
```

GetChannelErrors() :

このメソッドは、複数のチャンネル（nChannels > 1）を処理する際に、チャンネル固有の戻り値のリストを照会できるようにします。呼び出しは、ファンクションブロックの戻り値がeCM_InfRTime_AmbiguousChannelResults またはeCM_ErrRTime_ErrornousChannelResults のいずれかに該当する場合に有効です。

- **戻り値**: エラーコード一覧の読み込み処理に関する情報です。この値は、クエリが成功した場合はTRUE に、そうでない場合はFALSE に設定されます。

```
METHOD GetChannelErrors : BOOL
VAR_IN_OUT
    aChannelErrors : ARRAY[*] OF HRESULT;
END_VAR
```

- **aChannelErrors**:HRESULT 型の長さnChannelsのエラーリストです。

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM, Tc3_CM_Base

5.1.9 FB_CMA_DiscreteClassification

設定可能な閾値に基づいてマルチチャンネルデータを分類します。

このファンクションブロックは、設定可能な閾値に基づいて、マルチチャンネル信号の各チャンネルを固定数の離散カテゴリーに割り当てます。チャンネル数とカテゴリー数は、インスタンス作成時に指定します。このファンクションブロックは、実行時に各チャンネルと各カテゴリーの閾値を指定することで設定できます。

演算段階では、各時間ステップで入力ベクトルが適用され、各チャンネルごとに該当するカテゴリーの数が計算されます。戻り値は1次元の配列で、各入力チャンネルごとに符号付き整数値、つまり割り当てられたカテゴリーのインデックスを含みます。

入力値が最初のカテゴリーの閾値より小さい場合、このチャンネルには値-1が返されます。入力値がカテゴリーの閾値以上の場合、そのカテゴリーのゼロベースのインデックスが返されます。複数の閾値が同じように設定されている場合、最大のインデックスを持つ値が使用されます。

設定

ファンクションブロックは、分類クラスの数などのパラメータに基づいて設定する必要があります。各チャンネルの分類閾値は個別に割り当てることができます。これらの閾値は単調増加である必要があります（ただし、厳密な単調増加である必要はありません）。したがって、どの閾値も前の値より小さくなりません。

メモリ特性

ファンクションブロックは、設定とトレーニング中に保存された値のみを考慮します。分類中に転送された値は、後の呼び出しには影響しません。

NaN（非数）の発生

入力値がNaNの場合、結果は-2です。出力にはNaN値は含まれないことが想定されます。

マルチチャンネル入力データ処理時の動作

複数のチャンネル（ $nChannels > 1$ ）を処理する場合、各チャンネルが異なる戻り値になる可能性があります。この場合、戻り値はファンクションブロックで個別に問い合わせできます。全てのチャンネルではなく、1つ以上のチャンネルの結果が許容されない場合、ファンクションブロックの値は `eCM_InfRTIME_AmbiguousChannelResults` に対応します。全てのチャンネルの結果が許容されない場合、ファンクションブロック上の値は `eCM_ErrRTIME_ErrornousChannelResults` に対応します。

全チャンネルの戻り値のリストは、`GetChannelErrors()` のメソッドで照会できます。

複数のサブチャンネル（ $nSubChannels > 0$ ）を処理する場合、入出力データのフォーマットには特に注意が必要です。入力データが上流のファンクションブロックによるマルチチャンネルの結果で構成されている場合、 $nChannels$ の値を採用する必要があります。この場合、パラメータ $nSubChannels$ を使用して追加の設定を行います。

例：マルチチャンネルスペクトル（`FB_CMA_MagnitudeSpectrum` など）の周波数チャンネルの統計的考察（`FB_CMA_Quantiles` など）では、 $nChannels$ の値は、入力信号の数と同じである必要があります。サブチャンネルの数 $nSubChannels$ は、スペクトルの長さに対応します。

サンプルの実装

実装用の2つのサンプルは以下のリンクから入手できます：[平均化したマグニチュードスペクトルに対する閾値の考慮 \[▶ 340\]](#)、[周波数解析による状態監視 \[▶ 335\]](#)

入力と出力

入力バッファと出力バッファは、以下の定義（入力/出力形状）のいずれかに該当します。変数パラメータは、ファンクションブロック入力 `stInitPars` の一部です。

バージョン	入力バッファ（マルチアレイ入力ストリーム） 要素タイプ、次元数、次元サイズ	出力バッファ（マルチアレイ出力ストリーム） 要素タイプ、次元数、次元サイズ
標準バージョン (nSubChannels = 0)	LREAL, 1, nChannels	DINT (32bit), 1, nChannels
マルチチャンネルバージョン (nSubChannels > 0)	LREAL, 2, nChannels x nSubChannels	DINT (32bit), 2, nChannels x nSubChannels

```

VAR_INPUT
    stInitPars      : ST_CM_DiscreteClassification_InitPars; // init parameter
    nOwnID           : UDINT; // ID for this FB instance
    aDestIDs         : ARRAY[1..cCMA_MaxDest] OF UDINT; // IDs of destinations for output
    nResultBuffers   : UDINT := 4; // number of MultiArrays which should
be initialized for results (0 for no initialization)
    tTransferTimeout : LTIME := LTIME#500US; // timeout checking off during access
to inter-task FIFOs
END_VAR

```

入力パラメータ

このファンクションブロックの入力パラメータは初期化パラメータを表し、FBインスタンスの宣言で割り当て済みである必要があります。(または、Init()メソッド) 一度しか割り当てできません。実行中の変更はできません。

- **stInitPars:** ST_CM_DiscreteClassification_InitPars [▶ 283]型の初期化パラメータを持つファンクションブロック固有の構造体。パラメータは、上記の入出力バッファの定義と紐づいていなければなりません。
- **nOwnID:** 一意のIDでファンクションブロックインスタンスを識別します。これは常に0より大きくなければなりません。この目的のためには、列挙を定義するアプローチが確立されています。
- **aDestIDs:** 転送先のIDを指定することで、結果の転送先を定義します。出力バッファの定義（上記を参照）は、選択された各転送先の入力バッファの定義と紐づいていなければなりません。
- **nResultBuffers:** このファンクションブロックは、指定された数のマルチアレイバッファにより転送トレイストリームを初期化します。デフォルト値は4です。
- **tTransferTimeout:** 内部マルチアレイ転送の同期タイムアウトの設定です。並列処理 [▶ 72]の項を参照してください。

出力パラメータ

```

VAR_OUTPUT
    bError          : BOOL; // TRUE if an error occurs. Reset by next method call.
    hrErrorCode      : HRESULT; // '< 0' = error; '> 0' = info; '0' = no error/info
    ipErrorMessage  : I_TcMessage := fbErrorMessage; // Shows detailed information about occurred errors, warnings and more.
    nCntResults      : ULINT; // Counts outgoing results (MultiArrays were calculated and sent to transfer tray).
END_VAR

```

- **bError:** エラーが発生した場合の出力は、TRUEです。
- **hrErrorCode:** エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、エラーコード一覧 [▶ 367]に記載されています。
- **ipErrorMessage:** 現在の戻り値に関するより詳細な情報を含みます。エラーの説明と情報 [▶ 81]の項を参照してください。この特別なインターフェースポイントは、常に有効かつ割り当て済みであるよう、内部的に保護されています。

メソッド

Call():

このメソッドは、現在の入力データにアルゴリズムを適用するために、サイクルごとに呼び出されます。メソッドが新しい結果もエラーも示さない場合、ファンクションブロックは入力データを待機します。これは、解析チェーンのプロセスにおける規則的な動作です。

- **戻り値:** エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、エラーコード一覧 [▶ 367]に記載されています。

```

METHOD Call : HRESULT
VAR_OUTPUT
    bNewResult : BOOL;          // TRUE every time when outgoing MultiArray was calculated and sent to transfer tray.
    bError      : BOOL;          // TRUE if an error occurs.
    hrErrorCode : HRESULT;       // '< 0' = error; '> 0' = info; '0' = no error/info
END_VAR

```

- **bError:**エラーが発生した場合の出力は、TRUEです。
- **hrErrorCode:**エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。この出力はメソッドの戻り値と同じです。



タイムアウトが発生した場合、または結果を格納するマルチアレイバッファがない場合、入力データも結果データも失われません。次の呼び出しに転送されます。

Configure() :

分類の引数は、このメソッドを呼び出す際にあらかじめ設定する必要があります。対応するPLC配列は以下のように定義してください。Configure()メソッドは、異なる引数セットを持つ新しい設定にも使用できます。

- **戻り値 :** エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```

METHOD Configure : HRESULT
VAR_INPUT
    pArg : POINTER TO LREAL; // pointer to array (LREAL) of arguments
    nArgSize : UDINT;        // size of arguments buffer in bytes
END_VAR

```

入力バッファは、以下の定義（入力形状）のいずれかに紐づいています。変数パラメータは、ファンクションブロック入力stInitPars の一部です。

バージョン	入力バッファ（マルチアレイ入力ストリーム）要素タイプ、次元数、次元サイズ
全てのチャンネルおよびサブチャンネルで同一設定	LREAL, 1, nMaxClasses
チャンネル別の設定 (nSubChannels = 0)	LREAL, 2, nChannels x nMaxClasses
サブチャンネル別の設定 (nSubChannels > 0)	LREAL, 2, nSubChannels x nMaxClasses
チャンネルおよびサブチャンネル別の設定 (nSubChannels > 0)	LREAL, 3, nChannels x nSubChannels x nMaxClasses

```

METHOD Init : HRESULT
VAR_INPUT
    stInitPars : ST_CM_DiscreteClassification_InitPars; // init parameter
    nOwnID : UDINT; // ID for this FB instance
    aDestIDs : ARRAY[1..cCMA_MaxDest] OF UDINT; // IDs of destinations for output
    nResultBuffers : UDINT := 4; // number of MultiArrays which should be initialized for results (0 for no initialization)
END_VAR

```

Init():

このメソッドは通常、状態監視アプリケーションでは必要ありません。これはファンクションブロック初期化の代替手段を提供するものです。Init() メソッドは、PLCの初期化フェーズ中にのみ呼び出すことができます。実行時には使用できません。メソッドFB_init または、属性'call_after_init'の使用を参照してください（TwinCAT PLCリファレンスを参照）。さらに、これはファンクションブロックのカプセル化を容易にします。

Init()メソッドを使用して初期化を行う場合、ファンクションブロックインスタンスの入力パラメータを宣言で代入することはできません。

- **戻り値 :** エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

- **stInitPars:** ST_CM_DiscreteClassification_InitPars [▶ 283]型の初期化パラメータを持つファンクションブロック固有の構造体。パラメータは、上記の入出力バッファの定義と紐づいていなければなりません。
- **nOwnID:** 一意のIDでファンクションブロックインスタンスを識別します。これは常に0より大きくなければなりません。この目的のためには、列挙を定義するアプローチが確立されています。
- **aDestIDs:** 転送先のIDを指定することで、結果の転送先を定義します。出力バッファの定義（上記を参照）は、選択された各転送先の入力バッファの定義と紐づいていなければなりません。
- **nResultBuffers:** このファンクションブロックは、指定された数のマルチアレイバッファにより転送ストリームを初期化します。デフォルト値は4です。

PassInputs() :

API PLC リファレンス [▶ 79]で説明されているとおり、FB_CMA_Source インスタンスが呼び出され、信号データがターゲットブロックに転送される限り、解析チェーンの以降の全てのファンクションブロックを周期的に呼び出す必要があります。

アルゴリズムを一定時間実行しない方が有用な場合があります。例えば、いくつかのアルゴリズムは、事前に教示や設定をした後でのみ実行すべきです。ファンクションブロックは周期的に呼び出す必要がありますが、ファンクションブロックに到着したデータが通信リング [▶ 72]内で転送されれば十分です。これは、Call() メソッドの代わりにPassInputs() メソッドを使用して行われます。アルゴリズム自体はここでは呼び出されず、したがって結果も計算されず、出力バッファも生成されません。

- **戻り値:** エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧](#) [▶ 367]に記載されています。

```
METHOD PassInputs : HRESULT
VAR_INPUT
END_VAR
```

GetChannelErrors() :

このメソッドは、複数のチャンネル（nChannels > 1）を処理する際に、チャンネル固有の戻り値のリストを照会できるようにします。呼び出しは、ファンクションブロックの戻り値が eCM_InfRTTime_AmbiguousChannelResults または eCM_ErrRTTime_ErrornousChannelResults のいずれかに該当する場合に有用です。

- **戻り値:** エラーコード一覧の読み込み処理に関する情報です。この値は、クエリが成功した場合は TRUE に、そうでない場合は FALSE に設定されます。

```
METHOD GetChannelErrors : BOOL
VAR_IN_OUT
aChannelErrors : ARRAY[*] OF HRESULT;
END_VAR
```

- **aChannelErrors:** HRESULT 型の長さnChannelsのエラーリストです。

類似したファンクションブロック

ファンクションブロックFB_CMA_WatchUpperThresholds [▶ 246]は、マルチチャンネル入力データの最上位のカテゴリを計算し、対応するチャンネル番号とともに返します。

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM, Tc3_CM_Base



CM 3.1 では一部の機能がすでに使用可能です。[互換性](#) [▶ 61]の項を参照してください。

5.1.10 FB_CMA_Downsampling

PLCバッファから別のPLCバッファ（配列）に信号データをコピーすることでダウンサンプリングします。

バッファ（例えばオーバーサンプリング配列）として存在する信号は、個別に設定されたダウンサンプリング率でスキャン（読み取り）することが可能です。ダウンサンプリングは、高い分解能を維持しながら、FFT長を増やさずに低周波数を解析する方法です。

通常、ダウンサンプリングブロックは、状態監視解析チェーン内のFB_CMA_Source [▶ 230]の前に挿入されます。

入力と出力

入力パラメータ

```
VAR_INPUT
    nDivider    : UDINT := 1;    // given input signal is sampled down by the specified divider. (1 =
    no downsampling)
    nChannels    : UDINT;        // number of channels in data buffer (=1:onedimensional dataset, >1:
    twodimensional dataset)
END_VAR
```

- **nDivider:** ダウンサンプリング係数を整数の除数として指定します。例えば、10kHzのサンプルレートは、nDivider=5 で2kHzのサンプルレートに変換できます。
- **nChannels:** マルチチャンネルデータセットをダウンサンプリングする場合、チャンネル数は入力 nChannels で指定します。

出力パラメータ

```
VAR_OUTPUT
    bError      : BOOL;          // TRUE if an error occurs. Reset by next method call.
    hrErrorCode  : HRESULT;      // '< 0' = error; '> 0' = info; '0' = no error/info
    ipErrorMessage : I_TcMessage := fbErrorMessage; // Shows detailed information about occurred errors, warnings and more.
    nCntResults   : ULINT;        // Counts outgoing results (MultiArrays were calculated and sent to transfer tray).
END_VAR
```

- **bError:** エラーが発生した場合の出力は、TRUEです。
- **hrErrorCode:** エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。
- **ipErrorMessage:** 現在の戻り値に関するより詳細な情報を含みます。[エラーの説明と情報 \[▶ 81\]](#)の項を参照してください。この特別なインターフェースポイントは、常に有効かつ割り当て済みであるよう、内部的に保護されています。

メソッド

Call():

入力データバッファから出力データバッファにデータを書き込みます。bNewResultが設定されると、出力データバッファは満杯になります。

- **戻り値:** エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

METHOD Call : HRESULT

```
VAR_INPUT
    pDataIn      : POINTER TO BYTE; // address of data buffer (e.g. oversampling data)
    nDataInSize   : UDINT;           // size of data buffer in bytes
    pDataOut      : POINTER TO BYTE; // address of result buffer
    nDataOutSize  : UDINT;           // size of data buffer in bytes
    nElementSize  : UDINT;           // Size of element type in bytes
END_VAR
VAR_OUTPUT
    bNewResult    : BOOL;            // TRUE every time when outgoing data buffer was calculated.
```

```

bError      : BOOL;           // TRUE if an error occurs.
hrErrorCode  : HRESULT;       // '< 0' = error; '> 0' = info; '0' = no error/info
END_VAR

```

- pDataIn: 割り当てられた入力バッファ。
- nDataInSize: PLC入力バッファのサイズを指定し、以下の条件を満たす必要があります:
nChannels * nElementSize * „チャンネルあたりの要素数”。
- pDataOut: bNewResult が設定されるまで、割り当てられた出力バッファは変更しないでください。通常、入力バッファと出力バッファは常に維持されます。
- nDataOutSize: PLC出力バッファのサイズを指定し、次の条件を満たす必要があります: nDataOutSize = n * nDataInSize。商が余りなくパラメータnDividerで割り切れる場合、次の条件を代用できます:
nDataOutSize = n * (nDataInSize/nDivider)。
- nElementSize: 要素のサイズをバイト単位で指定します。例えば、LREAL要素の配列の場合、サイズは8です。
- bError: エラーが発生した場合の出力は、TRUEです。
- hrErrorCode: エラーが発生した場合、対応するHRESULT型のエラーコードが出力される。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。この出力はメソッドの戻り値と同じです。

i タイムアウトが発生した場合、または結果を格納するマルチアレイバッファがない場合、入力データも結果データも失われません。次の呼び出しに転送されます。

i エラーが発生した場合、コピーアクションが実行されます。

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM, Tc3_CM_Base

i CM 3.1 では一部の機能がすでに使用可能です。[互換性 \[▶ 61\]](#)の項を参照してください。

5.1.11 FB_CMA_EmpiricalExcess

シングルチャンネルおよびマルチチャンネルの実数値時系列データの過剰度を計算します。

このファンクションブロックは、入力信号を時系列データとして扱い、必要に応じて複数の独立したチャンネルとともに処理します。経験的超過値は、式に従って各チャンネルごとに計算されます。

$$E = \frac{1}{N} \sum_{n=0}^{N-1} \left(\frac{x[n] - \bar{x}}{s} \right)^4 - 3$$

ここでsは経験的標準偏差です。超過値は、経験的尖度から3を引いた値です（ここで、3は正規分布の尖度に該当します）。その結果、超過値の解釈となります：

Exceed > 0: 正規分布よりも尖った分布。サンプル内で頻繁にピークが現れることを示しています。

Exceed < 0: 正規分布に比べて平坦な分布。

超過値は、例えば、ローラーベアリングの局所的な損傷の上を転がる際に発生する振動信号のショックを評価できます。

各サイクルでチャンネルごとに1サンプル（最初の表の入出力を参照）を追加でき、1サイクルでチャンネルごとに複数のサンプルをサンプル数に追加することもできます（2番目の表の入出力を参照）。

追加のコメント

初期結果を計算するためには、4つの値が必要です。さらに、標準偏差はゼロであってはなりません。入力値が桁数で大きく異なる場合、結果が不正確になる可能性があります。

メモリ特性

サンプル数 N は、新しいデータセットが入力されるたびに自動的に増加し、現在の結果を計算します。つまり、ファンクションブロックがインスタンス化されてからの全ての入力値を使用します。サンプル数のゼロリセット（FBの内部メモリの削除）は、ResetData() メソッド、またはCallEx() メソッドが使用される場合、変数bResetData によって提供されます。

NaN（非数）の発生

入力値の数が特定のチャンネルの結果を計算するのに十分でない場合、または分散がゼロの場合、そのチャンネルに対してIEEE 754に従ってNaN（非数）の値が返されます。このエラー値があるかどうかは、ファンクションLrealIsNaN()を使用して確認できます。原因として、これまでのところ十分な入力データが転送されていないか、個々のチャンネルの入力値としてNaN値しか転送されていないことが考えられます。例えば、断線やスイッチングエラーによってセンサデータが転送されなかった場合など、時系列データの値が一定であれば、分散がゼロになることがあります。

入力値セットに特別な定数 NaN が含まれる場合、この時間ステップでは、チャンネルの統計に値は追加されません。つまり、NaN は欠損値の指標として扱われます。



NaN値の取り扱い

NaN値につながる上記の状況を除外できない、または安全に無視できない場合、アプリケーションプログラムはこれらのエラー値を適切に処理できるようにしてください。

マルチチャンネル入力データ処理時の動作

複数のチャンネル（ $nChannels > 1$ ）を処理する場合、各チャンネルが異なる戻り値になる可能性があります。この場合、戻り値はファンクションブロックで個別に問い合わせできます。全てのチャンネルではなく、1つ以上のチャンネルの結果が許容されない場合、ファンクションブロックの値はeCM_InfRTime_AmbiguousChannelResultsに対応します。全てのチャンネルの結果が許容されない場合、ファンクションブロック上の値はeCM_ErrRTime_ErrornousChannelResults に対応します。

全チャンネルの戻り値のリストは、GetChannelErrors()のメソッドで照会できます。

複数のサブチャンネル($nSubChannels > 0$)を処理する場合、入出力データのフォーマットには特に注意が必要です。入力データが上流のファンクションブロックによるマルチチャンネルの結果で構成されている場合、 $nChannels$ の値を採用する必要があります。この場合、パラメータ $nSubChannels$ を使用して追加の設定を行います。

例：マルチチャンネルスペクトル（FB_CMA_MagnitudeSpectrumなど）の周波数チャンネルの統計的考察（FB_CMA_Quantilesなど）では、 $nChannels$ の値は、入力信号の数と同じである必要があります。サブチャンネルの数 $nSubChannels$ は、スペクトルの長さに対応します。

サンプルの実装

実装用のサンプルは以下のリンクから入手できます：[統計的手法 \[▶ 327\]](#)

入力と出力

入力バッファと出力バッファは、以下の定義（形状）に該当します。変数パラメータは、ファンクションブロック入力stInitPars の一部です。

バージョン	入力バッファ（マルチアレイ入力ストリーム） 要素タイプ、次元数、次元サイズ	出力バッファ（マルチアレイ出力ストリーム） 要素タイプ、次元数、次元サイズ
標準バージョン ($nSubChannels = 0$)	LREAL, 1, $nChannels$	LREAL, 1, $nChannels$

バージョン	入力バッファ（マルチアレイ入力ストリーム） 要素タイプ、次元数、次元サイズ	出力バッファ（マルチアレイ出力ストリーム） 要素タイプ、次元数、次元サイズ
複数データセット用標準バージョン (nSubChannels = 0)	LREAL, 2, nChannelsx 指定なし*	LREAL, 1, nChannels
マルチチャンネルバージョン (nSubChannels > 0)	LREAL, 2, nChannels x nSubChannels	LREAL, 2, nChannels x nSubChannels
複数データセット用マルチチャンネルバージョン (nSubChannels > 0)	LREAL, 3, nChannels x nSubChannels x 指定なし*	LREAL, 2, nChannels x nSubChannels

*:この次元の長さは任意に選択できるため、アプリケーションや先行のアルゴリズムの出力バッファに合わせる必要があります。

```
VAR_INPUT
    stInitPars      : ST_CM_EmpiricalMoments_InitPars; // init parameter
    nOwnID          : UDINT;                          // ID for this FB instance
    aDestIDs: ARRAY[1..cMA_MaxDest] OF UDINT;          // IDs of destinations for output
    nResultBuffers  : UDINT := 4;                      // number of MultiArrays which should be initialized for results (0 for no initialization)
    tTransferTimeout : LTIME := LTIME#500US;          // timeout checking off during access to inter-task FIFOs
END_VAR
```

入力パラメータ

このファンクションブロックの入力パラメータは初期化パラメータを表し、FBインスタンスの宣言で割り当て済みである必要があります。(または、Init()メソッド) 一度しか割り当てできません。実行中の変更はできません。

- **stInitPars:** ST_CM_EmpiricalMoments_InitPars [▶ 283] 型の初期化パラメータを持つファンクションブロック固有の構造体。パラメータは、上記の入出力バッファの定義と紐づいていなければなりません。
- **nOwnID:**一意のIDでファンクションブロックインスタンスを識別します。これは常に0より大きくなければなりません。この目的のためには、列挙を定義するアプローチが確立されています。
- **aDestIDs:**転送先のIDを指定することで、結果の転送先を定義します。出力バッファの定義（上記を参照）は、選択された各転送先の入力バッファの定義と紐づいていなければなりません。
- **nResultBuffers:**このファンクションブロックは、指定された数のマルチアレイバッファにより転送トレイストリームを初期化します。デフォルト値は4です。
- **tTransferTimeout:**内部マルチアレイ転送の同期タイムアウトの設定です。並列処理 [▶ 72]の項を参照してください。

出力パラメータ

```
VAR_OUTPUT
    bError          : BOOL;                          // TRUE if an error occurs. Reset by next method call.
    hrErrorCode      : HRESULT;                      // '< 0' = error; '> 0' = info; '0' = no error/info
    ipErrorMessage  : I_TcMessage := fbErrorMessage; // Shows detailed information about occurred errors, warnings and more.
    nCntResults      : ULINT;                        // Counts outgoing results (MultiArrays were calculated and sent to transfer tray).
END_VAR
```

- **bError:**エラーが発生した場合の出力は、TRUEです。
- **hrErrorCode:**エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、エラーコード一覧 [▶ 367]に記載されています。
- **ipErrorMessage:**現在の戻り値に関するより詳細な情報を含みます。エラーの説明と情報 [▶ 81]の項を参照してください。この特別なインターフェースポイントは、常に有効かつ割り当て済みであるよう、内部的に保護されています。

メソッド

Call():

このメソッドは、現在の入力データにアルゴリズムを適用するために、サイクルごとに呼び出されます。メソッドが新しい結果もエラーも示さない場合、ファンクションブロックは入力データを待機します。これは、解析チェーンのプロセスにおける規則的な動作です。

- **戻り値**：エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD Call : HRESULT
VAR_OUTPUT
    bNewResult      : BOOL;           // TRUE every time when outgoing MultiArray was calculated and sent to transfer tray.
    bError           : BOOL;           // TRUE if an error occurs.
    hrErrorCode      : HRESULT;        // '< 0' = error; '> 0' = info; '0' = no error/info
END_VAR
```

- **bError**:エラーが発生した場合の出力は、TRUEです。
- **hrErrorCode**:エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。この出力はメソッドの戻り値と同じです。



タイムアウトが発生した場合、または結果を格納するマルチアレイバッファがない場合、入力データも結果データも失われません。次の呼び出しに転送されます。

CallEx() :

このメソッドは、入力信号から内部メモリを更新するために、各サイクルで呼び出されます。結果は nAppendData サイクルごとに出力されます。別の方法としては、Call() があります。

メソッドが新しい結果もエラーも示さない場合、ファンクションブロックは入力データを待機します。これは、解析チェーンのプロセスにおける規則的な動作です。

- **戻り値**：エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD CallEx : HRESULT
VAR_INPUT
    nAppendData      : UDINT;          // count of data buffers which are appended until calculation (1= calculate always)
    bResetData       : BOOL;           // automatic reset of dataset buffer after each calculation
END_VAR
VAR_OUTPUT
    bNewResult       : BOOL;           // TRUE every time when outgoing MultiArray was calculated and sent to transfer tray.
    bError            : BOOL;           // TRUE if an error occurs.
    hrErrorCode       : HRESULT;        // '< 0' = error; '> 0' = info; '0' = no error/info
END_VAR
```

- **nAppendData**:計算を実行する前に、いくつかの入力データバッファを収集するかを定義します。これは、正確な結果を得るために複数のデータブロックを追加することが望ましいためです。
- **bResetData**:これを設定した場合、内部データバッファは計算後に完全に削除されます。
- **bError**:エラーが発生した場合の出力は、TRUEです。
- **hrErrorCode**:エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。この出力はメソッドの戻り値と同じです。



タイムアウトが発生した場合、または結果を格納するマルチアレイバッファがない場合、入力データも結果データも失われません。次の呼び出しに転送されます。

```
METHOD Init : HRESULT
VAR_INPUT
    stInitPars       : ST_CM_EmpiricalMoments_InitPars; // init parameter
    nOwnID            : UDINT;                          // ID for this FB instance
    aDestIDs          : ARRAY[1..cCMA_MaxDest] OF UDINT; // IDs of destinations for output
    nResultBuffers    : UDINT := 4;                      // number of MultiArrays which should be initialized for results (0 for no initialization)
END_VAR
```


Init():

このメソッドは通常、状態監視アプリケーションでは必要ありません。これはファンクションブロック初期化の代替手段を提供するものです。Init() メソッドは、PLCの初期化フェーズ中にのみ呼び出すことができます。実行時には使用できません。メソッドFB_init または、属性'call_after_init' の使用を参照してください（TwinCAT PLCリファレンスを参照）。さらに、これはファンクションブロックのカプセル化を容易にします。

Init()メソッドを使用して初期化を行う場合、ファンクションブロックインスタンスの入力パラメータを宣言で代入することはできません。

- **戻り値**：エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。
- **stInitPars:ST_CM_EmpiricalMoments_InitPars [▶ 283]** 型の初期化パラメータを持つファンクションブロック固有の構造体。パラメータは、上記の入出力バッファの定義と紐づいていなければなりません。
- **nOwnID**:一意のIDでファンクションブロックインスタンスを識別します。これは常に0より大きくなければなりません。この目的のためには、列挙を定義するアプローチが確立されています。
- **aDestIDs**:転送先のIDを指定することで、結果の転送先を定義します。出力バッファの定義（上記を参照）は、選択された各転送先の入力バッファの定義と紐づいていなければなりません。
- **nResultBuffers**:このファンクションブロックは、指定された数のマルチアレイバッファにより転送ストリームを初期化します。デフォルト値は4です。

ResetData() :

このメソッドは、既に追加された全てのデータレコードを削除します。ファンクションブロックのメモリ特性を参照してください。ResetData() の後、Call() メソッドが再度呼び出された場合、有効な結果を計算するために内部メモリを補充する必要があります。

- **戻り値**：エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD ResetData : HRESULT
VAR_INPUT
END_VAR
```

または、CallEx() の自動リセットを使用することもできます。

PassInputs() :

API PLC リファレンス [▶ 79]で説明されているとおり、FB_CMA_Source インスタンスが呼び出され、信号データがターゲットブロックに転送される限り、解析チェーンの以降の全てのファンクションブロックを周期的に呼び出す必要があります。

アルゴリズムを一定時間実行しない方が有用な場合があります。例えば、いくつかのアルゴリズムは、事前に教示や設定をした後でのみ実行すべきです。ファンクションブロックは周期的に呼び出す必要がありますが、ファンクションブロックに到着したデータが[通信リング \[▶ 72\]](#)内で転送されれば十分です。これは、Call() メソッドの代わりにPassInputs() メソッドを使用して行われます。アルゴリズム自体はここでは呼び出されず、したがって結果も計算されず、出力バッファも生成されません。

- **戻り値**：エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD PassInputs : HRESULT
VAR_INPUT
END_VAR
```

GetChannelErrors() :

このメソッドは、複数のチャンネル（nChannels > 1）を処理する際に、チャンネル固有の戻り値のリストを照会できるようにします。呼び出しは、ファンクションブロックの戻り値がeCM_InfRTTime_AmbiguousChannelResults またはeCM_ErrRTTime_ErrornousChannelResults のいずれかに該当する場合に有効です。

- **戻り値**: エラーコード一覧の読み込み処理に関する情報です。この値は、クエリが成功した場合はTRUE に、そうでない場合はFALSE に設定されます。

```

METHOD GetChannelErrors : BOOL
VAR_IN_OUT
    aChannelErrors : ARRAY[*] OF HRESULT;
END_VAR

```

- aChannelErrors:HRESULT 型の長さnChannelsのエラーリストです。

類似したファンクションブロック

ファンクションブロックFB_CMA_EmpiricalMean [▶ 123] は、入力値の経験平均値を計算します。

ファンクションブロックFB_CMA_EmpiricalStandardDeviation [▶ 133] は、入力値の経験的標準偏差を計算します。

ファンクションブロックFB_CMA_EmpiricalSkew [▶ 128] は、入力値の経験的歪度を計算します。

ファンクションブロックFB_CMA_MomentCoefficients [▶ 172] は、パラメータ設定に応じて、経験平均値、および標準偏差、歪度、過剰度を計算します。

FB_CMA_HistArray [▶ 145]ファンクションブロックは、入力値分布のヒストグラムを計算します。

FB_CMA_Quantiles [▶ 193]ファンクションブロックは、経験分布の分位数を計算し、外れ値の周波数を評価します。

尖度の代替として、FB_CMA_CrestFactor [▶ 93]ファンクションブロックは、信号のピーク性（クレスト係数）の別の尺度を計算しますが、これは異常値に対してより敏感です。

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM, Tc3_CM_Base



CM 3.1 では一部の機能がすでに使用可能です。[互換性 \[▶ 61\]](#)の項を参照してください。

5.1.12 FB_CMA_EmpiricalMean

シングルチャンネルおよびマルチチャンネルの実数値時系列データの平均値を計算します。

このファンクションブロックは、入力信号を時系列データとして扱い、必要に応じて複数の独立したチャンネルとともに処理します。経験的（算術的）平均値は、式にしたがって各チャンネルで計算されます。

$$\bar{x} = \frac{1}{N} \sum_{n=0}^{N-1} x[n]$$

各サイクルでチャンネルごとに1サンプル（最初の表の入出力を参照）を追加でき、1サイクルでチャンネルごとに複数のサンプルをサンプル数に追加することもできます（2番目の表の入出力を参照）。

メモリ特性

サンプル数Nは、新しいデータセットが入力されるたびに自動的に増加し、現在の結果を計算します。つまり、ファンクションブロックがインスタンス化されてからの全ての入力値を使用します。サンプル数のゼロリセット（FBの内部メモリの削除）は、ResetData() メソッド、またはCallEx() メソッドが使用される場合、変数bResetData によって提供されます。

NaN（非数）の発生

入力値の数が特定のチャンネルの結果を計算するのに十分でない場合、または分散がゼロの場合、そのチャンネルに対してIEEE 754に従ってNaN（非数）の値が返されます。このエラー値があるかどうかは、ファンクションLrealIsNaN()を使用して確認できます。原因として、これまでのところ十分な入力データが転

送されていないか、個々のチャンネルの入力値としてNaN値しか転送されていないことが考えられます。例えば、断線やスイッチングエラーによってセンサデータが転送されなかった場合など、時系列データの値が一定であれば、分散がゼロになることがあります。

入力値セットに特別な定数 NaN が含まれる場合、この時間ステップでは、チャンネルの統計に値は追加されません。つまり、NaN は欠損値の指標として扱われます。

● NaN値の取り扱い

i NaN値につながる上記の状況を除外できない、または安全に無視できない場合、アプリケーションプログラムはこれらのエラー値を適切に処理できるようにしてください。

マルチチャンネル入力データ処理時の動作

複数のチャンネル ($nChannels > 1$) を処理する場合、各チャンネルが異なる戻り値になる可能性があります。この場合、戻り値はファンクションブロックで個別に問い合わせできます。全てのチャンネルではなく、1つ以上のチャンネルの結果が許容されない場合、ファンクションブロックの値は `eCM_InfRTime_AmbiguousChannelResults` に対応します。全てのチャンネルの結果が許容されない場合、ファンクションブロック上の値は `eCM_ErrRTime_ErrornousChannelResults` に対応します。

全チャンネルの戻り値のリストは、`GetChannelErrors()` のメソッドで照会できます。

複数のサブチャンネル ($nSubChannels > 0$) を処理する場合、入出力データのフォーマットには特に注意が必要です。入力データが上流のファンクションブロックによるマルチチャンネルの結果で構成されている場合、 $nChannels$ の値を採用する必要があります。この場合、パラメータ $nSubChannels$ を使用して追加の設定を行います。

例：マルチチャンネルスペクトル (`FB_CMA_MagnitudeSpectrum` など) の周波数チャンネルの統計的考察 (`FB_CMA_Quantiles` など) では、 $nChannels$ の値は、入力信号の数と同じである必要があります。サブチャンネルの数 $nSubChannels$ は、スペクトルの長さに対応します。

サンプルの実装

実装用のサンプルは以下のリンクから入手できます：[統計的手法 \[▶ 327\]](#)

入力と出力

入力バッファと出力バッファは、以下の定義（形状）に該当します。変数パラメータは、ファンクションブロック入力 `stInitPars` の一部です。

バージョン	入力バッファ（マルチアレイ入力ストリーム） 要素タイプ、次元数、次元サイズ	出力バッファ（マルチアレイ出力ストリーム） 要素タイプ、次元数、次元サイズ
標準バージョン ($nSubChannels = 0$)	LREAL, 1, $nChannels$	LREAL, 1, $nChannels$
複数データセット用標準バージョン ($nSubChannels = 0$)	LREAL, 2, $nChannels \times$ 指定なし*	LREAL, 1, $nChannels$
マルチチャンネルバージョン ($nSubChannels > 0$)	LREAL, 2, $nChannels \times nSubChannels$	LREAL, 2, $nChannels \times nSubChannels$
複数データセット用マルチチャンネルバージョン ($nSubChannels > 0$)	LREAL, 3, $nChannels \times nSubChannels \times$ 指定なし*	LREAL, 2, $nChannels \times nSubChannels$

*:この次元の長さは任意に選択できるため、アプリケーションや先行のアルゴリズムの出力バッファに合わせるができます。

```
VAR_INPUT
    stInitPars      : ST_CM_EmpiricalMoments_InitPars; // init parameter
    nOwnID          : UDINT;                          // ID for this FB instance
    aDestIDs: ARRAY[1..cCMA_MaxDest] OF UDINT;        // IDs of destinations for output
    nResultBuffers  : UDINT := 4;                      // number of MultiArrays which should be in
initialized for results (0 for no initialization)
```

```

tTransferTimeout : LTIME := LTIME#500US;           // timeout checking off during access to in
ter-task FIFOs
END_VAR

```

入力パラメータ

このファンクションブロックの入力パラメータは初期化パラメータを表し、FBインスタンスの宣言で割り当て済みである必要があります。(または、Init()メソッド) 一度しか割り当てできません。実行中の変更はできません。

- **stInitPars:** ST_CM_EmpiricalMoments_InitPars [▶ 283] 型の初期化パラメータを持つファンクションブロック固有の構造体。パラメータは、上記の入出力バッファの定義と紐づいていなければなりません。
- **nOwnID:** 一意のIDでファンクションブロックインスタンスを識別します。これは常に0より大きくなければなりません。この目的のためには、列挙を定義するアプローチが確立されています。
- **aDestIDs:** 転送先のIDを指定することで、結果の転送先を定義します。出力バッファの定義（上記を参照）は、選択された各転送先の入力バッファの定義と紐づいていなければなりません。
- **nResultBuffers:** このファンクションブロックは、指定された数のマルチアレイバッファにより転送トレイストリームを初期化します。デフォルト値は4です。
- **tTransferTimeout:** 内部マルチアレイ転送の同期タイムアウトの設定です。[並列処理 \[▶ 72\]](#)の項を参照してください。

出力パラメータ

```

VAR_OUTPUT
    bError      : BOOL;           // TRUE if an error occurs. Reset by next metho
d call.
    hrErrorCode  : HRESULT;       // '< 0' = error; '> 0' = info; '0' = no error/
info
    ipErrorMessage : I_TcMessage := fbErrorMessage; // Shows detailed information about occurred er
rors, warnings and more.
    nCntResults   : ULINT;        // Counts outgoing results (MultiArrays were ca
lculated and sent to transfer tray).
END_VAR

```

- **bError:** エラーが発生した場合の出力は、TRUEです。
- **hrErrorCode:** エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。
- **ipErrorMessage:** 現在の戻り値に関するより詳細な情報を含みます。[エラーの説明と情報 \[▶ 81\]](#)の項を参照してください。この特別なインターフェースポインタは、常に有効かつ割り当て済みであるよう、内部的に保護されています。

メソッド

```

METHOD Init : HRESULT
VAR_INPUT
    stInitPars      : ST_CM_EmpiricalMoments_InitPars; // init parameter
    nOwnID          : UDINT;                          // ID for this FB instance
    aDestIDs        : ARRAY[1..cCMA_MaxDest] OF UDINT; // IDs of destinations for output
    nResultBuffers  : UDINT := 4;                     // number of MultiArrays which should be ini
tialized for results (0 for no initialization)
END_VAR

```

Call():

このメソッドは、現在の入力データにアルゴリズムを適用するために、サイクルごとに呼び出されます。メソッドが新しい結果もエラーも示さない場合、ファンクションブロックは入力データを待機します。これは、解析チェーンのプロセスにおける規則的な動作です。

- **戻り値:** エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```

METHOD Call : HRESULT
VAR_OUTPUT
    bNewResult      : BOOL;           // TRUE every time when outgoing MultiArray was calculated and sent t
o transfer tray.
    bError          : BOOL;           // TRUE if an error occurs.
    hrErrorCode      : HRESULT;       // '< 0' = error; '> 0' = info; '0' = no error/info
END_VAR

```

- **bError**:エラーが発生した場合の出力は、TRUEです。
- **hrErrorCode**:エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。この出力はメソッドの戻り値と同じです。



タイムアウトが発生した場合、または結果を格納するマルチアレイバッファがない場合、入力データも結果データも失われません。次の呼び出しに転送されます。

CallEx() :

このメソッドは、入力信号から内部メモリを更新するために、各サイクルで呼び出されます。結果は **nAppendData** サイクルごとに出力されます。別の方法としては、**Call()** があります。

メソッドが新しい結果もエラーも示さない場合、ファンクションブロックは入力データを待機します。これは、解析チェーンのプロセスにおける規則的な動作です。

- **戻り値** : エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD CallEx : HRESULT
VAR_INPUT
    nAppendData : UDINT;          // count of data buffers which are appended until calculation (1= calculate always)
    bResetData  : BOOL;          // automatic reset of dataset buffer after each calculation
END_VAR
VAR_OUTPUT
    bNewResult  : BOOL;          // TRUE every time when outgoing MultiArray was calculated and sent to transfer tray.
    bError      : BOOL;          // TRUE if an error occurs.
    hrErrorCode : HRESULT;       // '< 0' = error; '> 0' = info; '0' = no error/info
END_VAR
```

- **nAppendData**:計算を実行する前に、いくつかの入力データバッファを収集するかを定義します。これは、正確な結果を得るために複数のデータブロックを追加することが望ましいためです。
- **bResetData**:これを設定した場合、内部データバッファは計算後に完全に削除されます。
- **bError**:エラーが発生した場合の出力は、TRUEです。
- **hrErrorCode**:エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。この出力はメソッドの戻り値と同じです。



タイムアウトが発生した場合、または結果を格納するマルチアレイバッファがない場合、入力データも結果データも失われません。次の呼び出しに転送されます。

Init():

このメソッドは通常、状態監視アプリケーションでは必要ありません。これはファンクションブロック初期化の代替手段を提供するものです。Init() メソッドは、PLCの初期化フェーズ中にのみ呼び出すことができます。実行時には使用できません。メソッド**FB_init** または、属性'**call_after_init**'の使用を参照してください（TwinCAT PLCリファレンスを参照）。さらに、これはファンクションブロックのカプセル化を容易にします。

Init()メソッドを使用して初期化を行う場合、ファンクションブロックインスタンスの入力パラメータを宣言で代入することはできません。

- **戻り値** : エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。
- **stInitPars**:**ST_CM_EmpiricalMoments_InitPars [▶ 283]** 型の初期化パラメータを持つファンクションブロック固有の構造体。パラメータは、上記の入出力バッファの定義と紐づいていなければなりません。
- **nOwnID**:一意のIDでファンクションブロックインスタンスを識別します。これは常に0より大きくなければなりません。この目的のためには、列挙を定義するアプローチが確立されています。
- **aDestIDs**:転送先のIDを指定することで、結果の転送先を定義します。出力バッファの定義（上記を参照）は、選択された各転送先の入力バッファの定義と紐づいていなければなりません。

- **nResultBuffers**: このファンクションブロックは、指定された数のマルチアレイバッファにより転送トレイストリームを初期化します。デフォルト値は4です。

ResetData() :

このメソッドは、既に追加された全てのデータレコードを削除します。ファンクションブロックのメモリ特性を参照してください。ResetData() の後、Call() メソッドが再度呼び出された場合、有効な結果を計算するために内部メモリを補充する必要があります。

- **戻り値**: エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD ResetData : HRESULT
VAR_INPUT
END_VAR
```

または、CallEx() の自動リセットを使用することもできます。

PassInputs() :

API PLC リファレンス [▶ 79]で説明されているとおり、FB_CMA_Source インスタンスが呼び出され、信号データがターゲットブロックに転送される限り、解析チェーンの以降の全てのファンクションブロックを周期的に呼び出す必要があります。

アルゴリズムを一定時間実行しない方が有用な場合があります。例えば、いくつかのアルゴリズムは、事前に教示や設定をした後でのみ実行すべきです。ファンクションブロックは周期的に呼び出す必要がありますが、ファンクションブロックに到着したデータが[通信リング \[▶ 72\]](#)内で転送されれば十分です。これは、Call() メソッドの代わりにPassInputs() メソッドを使用して行われます。アルゴリズム自体はここでは呼び出されず、したがって結果も計算されず、出力バッファも生成されません。

- **戻り値**: エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD PassInputs : HRESULT
VAR_INPUT
END_VAR
```

GetChannelErrors() :

このメソッドは、複数のチャンネル (nChannels > 1) を処理する際に、チャンネル固有の戻り値のリストを照会できるようにします。呼び出しは、ファンクションブロックの戻り値が eCM_InfRTTime_AmbiguousChannelResults または eCM_ErrRTTime_ErrornousChannelResults のいずれかに該当する場合に有効です。

- **戻り値**: エラーコード一覧の読み込み処理に関する情報です。この値は、クエリが成功した場合は TRUE に、そうでない場合は FALSE に設定されます。

```
METHOD GetChannelErrors : BOOL
VAR_IN_OUT
aChannelErrors : ARRAY[*] OF HRESULT;
END_VAR
```

- **aChannelErrors**: HRESULT 型の長さ nChannels のエラーリストです。

類似したファンクションブロック

ファンクションブロック [FB_CMA_EmpiricalStandardDeviation \[▶ 133\]](#) は、入力値の経験的標準偏差を計算します。

ファンクションブロック [FB_CMA_EmpiricalSkew \[▶ 128\]](#) は、入力値の経験的歪度を計算します。

ファンクションブロック [FB_CMA_EmpiricalExcess \[▶ 118\]](#) は、入力値の経験的な超過分を計算します。

ファンクションブロック [FB_CMA_MomentCoefficients \[▶ 172\]](#) は、パラメータ設定に応じて、経験平均値、および標準偏差、歪度、過剰度を計算します。

[FB_CMA_HistArray \[▶ 145\]](#) ファンクションブロックは、入力値分布のヒストグラムを計算します。

[FB_CMA_Quantiles \[▶ 193\]](#) ファンクションブロックは、経験分布の分位数を計算し、外れ値の周波数を評価します。

尖度の代替として、FB_CMA_CrestFactor [▶ 93]ファンクションブロックは、信号のピーク性（クレスト係数）の別の尺度を計算しますが、これは異常値に対してより敏感です。

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM, Tc3_CM_Base



CM 3.1 では一部の機能がすでに使用可能です。[互換性 \[▶ 61\]](#)の項を参照してください。

5.1.13 FB_CMA_EmpiricalSkew

シングルチャンネルおよびマルチチャンネルの実数値時系列データの経験的歪度を計算します。

このファンクションブロックは、入力信号を時系列データとして扱い、必要に応じて複数の独立したチャンネルとともに処理します。経験的歪度は、次の式にしたがって各チャンネルで計算されます。

$$v = \frac{1}{N} \sum_{n=0}^{N-1} \left(\frac{x[n] - \bar{x}}{s} \right)^3$$

ここでsは経験的標準偏差です。歪度は、サンプルの非対称性を定量的に表現します。これは、振動信号における衝撃（例えば、ローラーベアリングの局所的な損傷の上を転がるなど）を評価できます。局所的な損傷が必ずしも非対称な信号分布につながるわけではありませんが、計算された歪度は、尖度/過剰度よりも堅牢な基準です。

各サイクルでチャンネルごとに1サンプル（最初の表の入出力を参照）を追加でき、1サイクルでチャンネルごとに複数のサンプルをサンプル数に追加することもできます（2番目の表の入出力を参照）。

追加のコメント

初期結果を計算するためには、3つの値が必要です。さらに、標準偏差はゼロであってはなりません。入力値が桁数で大きく異なる場合、結果が不正確になる可能性があります。

メモリ特性

サンプル数Nは、新しいデータセットが入力されるたびに自動的に増加し、現在の結果を計算します。つまり、ファンクションブロックがインスタンス化されてからの全ての入力値を使用します。サンプル数のゼロリセット（FBの内部メモリの削除）は、ResetData() メソッド、またはCallEx() メソッドが使用される場合、変数bResetData によって提供されます。

NaN（非数）の発生

入力値の数が特定のチャンネルの結果を計算するのに十分でない場合、または分散がゼロの場合、そのチャンネルに対してIEEE 754に従ってNaN（非数）の値が返されます。このエラー値があるかどうかは、ファンクションLrealIsNaN()を使用して確認できます。原因として、これまでのところ十分な入力データが転送されていないか、個々のチャンネルの入力値としてNaN値しか転送されていないことが考えられます。例えば、断線やスイッチングエラーによってセンサデータが転送されなかった場合など、時系列データの値が一定であれば、分散がゼロになることがあります。

入力値セットに特別な定数 NaN が含まれる場合、この時間ステップでは、チャンネルの統計に値は追加されません。つまり、NaN は欠損値の指標として扱われます。



NaN値の取り扱い

NaN値につながる上記の状況を除外できない、または安全に無視できない場合、アプリケーションプログラムはこれらのエラー値を適切に処理できるようにしてください。

マルチチャンネル入力データ処理時の動作

複数のチャンネル ($nChannels > 1$) を処理する場合、各チャンネルが異なる戻り値になる可能性があります。この場合、戻り値はファンクションブロックで個別に問い合わせできます。全てのチャンネルではなく、1つ以上のチャンネルの結果が許容されない場合、ファンクションブロックの値は `eCM_InfrTime_AmbiguousChannelResults` に対応します。全てのチャンネルの結果が許容されない場合、ファンクションブロック上の値は `eCM_ErrRTime_ErrornousChannelResults` に対応します。

全チャンネルの戻り値のリストは、`GetChannelErrors()` のメソッドで照会できます。

複数のサブチャンネル ($nSubChannels > 0$) を処理する場合、入出力データのフォーマットには特に注意が必要です。入力データが上流のファンクションブロックによるマルチチャンネルの結果で構成されている場合、 $nChannels$ の値を採用する必要があります。この場合、パラメータ $nSubChannels$ を使用して追加の設定を行います。

例：マルチチャンネルスペクトル (`FB_CMA_MagnitudeSpectrum` など) の周波数チャンネルの統計的考察 (`FB_CMA_Quantiles` など) では、 $nChannels$ の値は、入力信号の数と同じである必要があります。サブチャンネルの数 $nSubChannels$ は、スペクトルの長さに対応します。

サンプルの実装

実装用のサンプルは以下のリンクから入手できます：[統計的手法 \[▶ 327\]](#)

入力と出力

入力バッファと出力バッファは、以下の定義（形状）に該当します。変数パラメータは、ファンクションブロック入力 `stInitPars` の一部です。

バージョン	入力バッファ（マルチアレイ入力ストリーム） 要素タイプ、次元数、次元サイズ	出力バッファ（マルチアレイ出力ストリーム） 要素タイプ、次元数、次元サイズ
標準バージョン ($nSubChannels = 0$)	LREAL, 1, $nChannels$	LREAL, 1, $nChannels$
複数データセット用標準バージョン ($nSubChannels = 0$)	LREAL, 2, $nChannels \times$ 指定なし*	LREAL, 1, $nChannels$
マルチチャンネルバージョン ($nSubChannels > 0$)	LREAL, 2, $nChannels \times nSubChannels$	LREAL, 2, $nChannels \times nSubChannels$
複数データセット用マルチチャンネルバージョン ($nSubChannels > 0$)	LREAL, 3, $nChannels \times nSubChannels \times$ 指定なし*	LREAL, 2, $nChannels \times nSubChannels$

*: この次元の長さは任意に選択できるため、アプリケーションや先行のアルゴリズムの出力バッファに合わせるができます。

```

VAR_INPUT
    stInitPars      : ST_CM_EmpiricalMoments_InitPars; // init parameter
    nOwnID          : UDINT;                          // ID for this FB instance
    aDestIDs: ARRAY[1..cCMA_MaxDest] OF UDINT;        // IDs of destinations for output
    nResultBuffers  : UDINT := 4;                     // number of MultiArrays which should be initialized for results (0 for no initialization)
    tTransferTimeout : LTIME := LTIME#500US;          // timeout checking off during access to inter-task FIFOs
END_VAR

```

入力パラメータ

このファンクションブロックの入力パラメータは初期化パラメータを表し、FBインスタンスの宣言で割り当て済みである必要があります。(または、`Init()`メソッド) 一度しか割り当てできません。実行中の変更はできません。

- `stInitPars: ST_CM_EmpiricalMoments_InitPars [▶ 283]` 型の初期化パラメータを持つファンクションブロック固有の構造体。パラメータは、上記の入出力バッファの定義と紐づいていなければなりません。
- `nOwnID`: 一意のIDでファンクションブロックインスタンスを識別します。これは常に0より大きくなければなりません。この目的のためには、列挙を定義するアプローチが確立されています。

- **aDestIDs:** 転送先のIDを指定することで、結果の転送先を定義します。出力バッファの定義（上記を参照）は、選択された各転送先の入力バッファの定義と紐づいていなければなりません。
- **nResultBuffers:** このファンクションブロックは、指定された数のマルチアレイバッファにより転送トレイストリームを初期化します。デフォルト値は4です。
- **tTransferTimeout:** 内部マルチアレイ転送の同期タイムアウトの設定です。[並列処理 \[▶ 72\]](#)の項を参照してください。

出力パラメータ

```
VAR_OUTPUT
    bError          : BOOL;                // TRUE if an error occurs. Reset by next method call.
    hrErrorCode      : HRESULT;            // '< 0' = error; '> 0' = info; '0' = no error/info
    ipErrorMessage   : I_TcMessage := fbErrorMessage; // Shows detailed information about occurred errors, warnings and more.
    nCntResults      : ULINT;              // Counts outgoing results (MultiArrays were calculated and sent to transfer tray).
END_VAR
```

- **bError:** エラーが発生した場合の出力は、TRUEです。
- **hrErrorCode:** エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。
- **ipErrorMessage:** 現在の戻り値に関するより詳細な情報を含みます。[エラーの説明と情報 \[▶ 81\]](#)の項を参照してください。この特別なインターフェースポイントは、常に有効かつ割り当て済みであるよう、内部的に保護されています。

メソッド

```
METHOD Init : HRESULT
VAR_INPUT
    stInitPars      : ST_CM_EmpiricalMoments_InitPars; // init parameter
    nOwnID           : UDINT;                          // ID for this FB instance
    aDestIDs         : ARRAY[1..cCMA_MaxDest] OF UDINT; // IDs of destinations for output
    nResultBuffers   : UDINT := 4;                    // number of MultiArrays which should be initialized for results (0 for no initialization)
END_VAR
```

Call():

このメソッドは、現在の入力データにアルゴリズムを適用するために、サイクルごとに呼び出されます。メソッドが新しい結果もエラーも示さない場合、ファンクションブロックは入力データを待機します。これは、解析チェーンのプロセスにおける規則的な動作です。

- **戻り値:** エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD Call : HRESULT
VAR_OUTPUT
    bNewResult      : BOOL;                // TRUE every time when outgoing MultiArray was calculated and sent to transfer tray.
    bError          : BOOL;                // TRUE if an error occurs.
    hrErrorCode      : HRESULT;            // '< 0' = error; '> 0' = info; '0' = no error/info
END_VAR
```

- **bError:** エラーが発生した場合の出力は、TRUEです。
- **hrErrorCode:** エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。この出力はメソッドの戻り値と同じです。



タイムアウトが発生した場合、または結果を格納するマルチアレイバッファがない場合、入力データも結果データも失われません。次の呼び出しに転送されます。

CallEx() :

このメソッドは、入力信号から内部メモリを更新するために、各サイクルで呼び出されます。結果はnAppendData サイクルごとに出力されます。別の方法としては、Call() があります。

メソッドが新しい結果もエラーも示さない場合、ファンクションブロックは入力データを待機します。これは、解析チェーンのプロセスにおける規則的な動作です。

- **戻り値**：エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD CallEx : HRESULT
VAR_INPUT
    nAppendData : UDINT;          // count of data buffers which are appended until calculation (1= calculate always)
    bResetData  : BOOL;           // automatic reset of dataset buffer after each calculation
END_VAR
VAR_OUTPUT
    bNewResult  : BOOL;           // TRUE every time when outgoing MultiArray was calculated and sent to transfer tray.
    bError      : BOOL;           // TRUE if an error occurs.
    hrErrorCode : HRESULT;        // '< 0' = error; '> 0' = info; '0' = no error/info
END_VAR
```

- **nAppendData**:計算を実行する前に、いくつかの入力データバッファを収集するかを定義します。これは、正確な結果を得るために複数のデータブロックを追加することが望ましいためです。
- **bResetData**:これを設定した場合、内部データバッファは計算後に完全に削除されます。
- **bError**:エラーが発生した場合の出力は、TRUEです。
- **hrErrorCode**:エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。この出力はメソッドの戻り値と同じです。



タイムアウトが発生した場合、または結果を格納するマルチアレイバッファがない場合、入力データも結果データも失われません。次の呼び出しに転送されます。

Init():

このメソッドは通常、状態監視アプリケーションでは必要ありません。これはファンクションブロック初期化の代替手段を提供するものです。Init() メソッドは、PLCの初期化フェーズ中にのみ呼び出すことができます。実行時には使用できません。メソッドFB_init または、属性'call_after_init' の使用を参照してください（TwinCAT PLCリファレンスを参照）。さらに、これはファンクションブロックのカプセル化を容易にします。

Init()メソッドを使用して初期化を行う場合、ファンクションブロックインスタンスの入力パラメータを宣言で代入することはできません。

- **戻り値**：エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。
- **stInitPars**:[ST_CM_EmpiricalMoments_InitPars \[▶ 283\]](#) 型の初期化パラメータを持つファンクションブロック固有の構造体。パラメータは、上記の入出力バッファの定義と紐づいていなければなりません。
- **nOwnID**:一意のIDでファンクションブロックインスタンスを識別します。これは常に0より大きくなければなりません。この目的のためには、列挙を定義するアプローチが確立されています。
- **aDestIDs**:転送先のIDを指定することで、結果の転送先を定義します。出力バッファの定義（上記を参照）は、選択された各転送先の入力バッファの定義と紐づいていなければなりません。
- **nResultBuffers**:このファンクションブロックは、指定された数のマルチアレイバッファにより転送ストリームを初期化します。デフォルト値は4です。

ResetData() :

このメソッドは、既に追加された全てのデータレコードを削除します。ファンクションブロックのメモリ特性を参照してください。ResetData() の後、Call() メソッドが再度呼び出された場合、有効な結果を計算するために内部メモリを補充する必要があります。

- **戻り値**：エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD ResetData : HRESULT
VAR_INPUT
END_VAR
```

または、CallEx() の自動リセットを使用することもできます。

PassInputs() :

API PLC リファレンス [▶ 79]で説明されているとおり、FB_CMA_Source インスタンスが呼び出され、信号データがターゲットブロックに転送される限り、解析チェーンの以降の全てのファンクションブロックを周期的に呼び出す必要があります。

アルゴリズムを一定時間実行しない方が有用な場合があります。例えば、いくつかのアルゴリズムは、事前に教示や設定をした後でのみ実行すべきです。ファンクションブロックは周期的に呼び出す必要がありますが、ファンクションブロックに到着したデータが通信リング [▶ 72]内で転送されれば十分です。これは、Call() メソッドの代わりにPassInputs() メソッドを使用して行われます。アルゴリズム自体はここでは呼び出されず、したがって結果も計算されず、出力バッファも生成されません。

- **戻り値**：エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD PassInputs : HRESULT
VAR_INPUT
END_VAR
```

GetChannelErrors() :

このメソッドは、複数のチャンネル（nChannels > 1）を処理する際に、チャンネル固有の戻り値のリストを照会できるようにします。呼び出しは、ファンクションブロックの戻り値が eCM_InfRTTime_AmbiguousChannelResults または eCM_ErrRTTime_ErrornousChannelResults のいずれかに該当する場合に有用です。

- **戻り値**：エラーコード一覧の読み込み処理に関する情報です。この値は、クエリが成功した場合は TRUE に、そうでない場合は FALSE に設定されます。

```
METHOD GetChannelErrors : BOOL
VAR_IN_OUT
aChannelErrors : ARRAY[*] OF HRESULT;
END_VAR
```

- aChannelErrors:HRESULT 型の長さnChannelsのエラーリストです。

類似したファンクションブロック

ファンクションブロックFB_CMA_EmpiricalMean [▶ 123] は、入力値の経験平均値を計算します。

ファンクションブロックFB_CMA_EmpiricalStandardDeviation [▶ 133] は、入力値の経験的標準偏差を計算します。

ファンクションブロックFB_CMA_EmpiricalExcess [▶ 118] は、入力値の経験的な超過分を計算します。

ファンクションブロックFB_CMA_MomentCoefficients [▶ 172] は、パラメータ設定に応じて、経験平均値、および標準偏差、歪度、過剰度を計算します。

FB_CMA_HistArray [▶ 145]ファンクションブロックは、入力値分布のヒストグラムを計算します。

FB_CMA_Quantiles [▶ 193]ファンクションブロックは、経験分布の分位数を計算し、外れ値の周波数を評価します。

尖度の代替として、FB_CMA_CrestFactor [▶ 93]ファンクションブロックは、信号のピーク性（クレスト係数）の別の尺度を計算しますが、これは異常値に対してより敏感です。

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM, Tc3_CM_Base



CM 3.1 では一部の機能がすでに使用可能です。[互換性 \[▶ 61\]](#)の項を参照してください。

5.1.14 FB_CMA_EmpiricalStandardDeviation

シングルチャンネルおよびマルチチャンネルの実数値時系列データの経験的標準偏差を計算します。

このファンクションブロックは、入力信号を時系列データとして扱い、必要に応じて複数の独立したチャンネルとともに処理します。経験的標準偏差は、式にしたがって各チャンネルで計算されます。

$$s = \sqrt{\frac{1}{N-1} \sum_{n=0}^{N-1} (x[n] - \bar{x})^2}$$

FB_CMA_MomentCoefficients [▶ 172] とは対照的に、常にベッセル補正が適用されます。各サイクルでチャンネルごとに1サンプル（最初の表の入出力を参照）を追加でき、1サイクルでチャンネルごとに複数のサンプルをサンプル数に追加することもできます（2番目の表の入出力を参照）。

追加のコメント

初期結果を計算するためには、2つの値が必要です。入力値が桁数で大きく異なる場合、結果が不正確になる可能性があります。

メモリ特性

サンプル数Nは、新しいデータセットが入力されるたびに自動的に増加し、現在の結果を計算します。つまり、ファンクションブロックがインスタンス化されてからの全ての入力値を使用します。サンプル数のゼロリセット（FBの内部メモリの削除）は、ResetData() メソッド、またはCallEx() メソッドが使用される場合、変数bResetData によって提供されます。

NaN（非数）の発生

入力値の数が特定のチャンネルの結果を計算するのに十分でない場合、または分散がゼロの場合、そのチャンネルに対してIEEE 754に従ってNaN（非数）の値が返されます。このエラー値があるかどうかは、ファンクションLrealIsNaN()を使用して確認できます。原因として、これまでのところ十分な入力データが転送されていないか、個々のチャンネルの入力値としてNaN値しか転送されていないことが考えられます。例えば、断線やスイッチングエラーによってセンサデータが転送されなかった場合など、時系列データの値が一定であれば、分散がゼロになることがあります。

入力値セットに特別な定数 NaN が含まれる場合、この時間ステップでは、チャンネルの統計に値は追加されません。つまり、NaN は欠損値の指標として扱われます。

● NaN値の取り扱い



NaN値につながる上記の状況を除外できない、または安全に無視できない場合、アプリケーションプログラムはこれらのエラー値を適切に処理できるようにしてください。

マルチチャンネル入力データ処理時の動作

複数のチャンネル（nChannels > 1）を処理する場合、各チャンネルが異なる戻り値になる可能性があります。この場合、戻り値はファンクションブロックで個別に問い合わせできます。全てのチャンネルではなく、1つ以上のチャンネルの結果が許容されない場合、ファンクションブロックの値はeCM_InfRTime_AmbiguousChannelResultsに対応します。全てのチャンネルの結果が許容されない場合、ファンクションブロック上の値はeCM_ErrRTime_ErrornousChannelResults に対応します。

全チャンネルの戻り値のリストは、GetChannelErrors()のメソッドで照会できます。

複数のサブチャンネル(nSubChannels > 0)を処理する場合、入出力データのフォーマットには特に注意が必要です。入力データが上流のファンクションブロックによるマルチチャンネルの結果で構成されている場合、nChannels の値を採用する必要があります。この場合、パラメータnSubChannels を使用して追加の設定を行います。

例：マルチチャンネルスペクトル（FB_CMA_MagnitudeSpectrumなど）の周波数チャンネルの統計的考察（FB_CMA_Quantilesなど）では、nChannels の値は、入力信号の数と同じである必要があります。サブチャンネルの数nSubChannels は、スペクトルの長さに対応します。

サンプルの実装

実装用のサンプルは以下のリンクから入手できます：[統計的手法 \[▶ 327\]](#)

入力と出力

入力バッファと出力バッファは、以下の定義（形状）に該当します。変数パラメータは、ファンクションブロック入力stInitParsの一部です。

バージョン	入力バッファ（マルチアレイ入力ストリーム） 要素タイプ、次元数、次元サイズ	出力バッファ（マルチアレイ出力ストリーム） 要素タイプ、次元数、次元サイズ
標準バージョン (nSubChannels = 0)	LREAL, 1, nChannels	LREAL, 1, nChannels
複数データセット用標準バージョン (nSubChannels = 0)	LREAL, 2, nChannelsx 指定なし*	LREAL, 1, nChannels
マルチチャンネルバージョン (nSubChannels > 0)	LREAL, 2, nChannels x nSubChannels	LREAL, 2, nChannels x nSubChannels
複数データセット用マルチチャンネルバージョン (nSubChannels > 0)	LREAL, 3, nChannels x nSubChannels x 指定なし*	LREAL, 2, nChannels x nSubChannels

*:この次元の長さは任意に選択できるため、アプリケーションや先行のアルゴリズムの出力バッファに合わせるすることができます。

```
VAR_INPUT
    stInitPars      : ST_CM_EmpiricalMoments_InitPars; // init parameter
    nOwnID          : UDINT;                          // ID for this FB instance
    aDestIDs: ARRAY[1..cCMA_MaxDest] OF UDINT;        // IDs of destinations for output
    nResultBuffers  : UDINT := 4;                     // number of MultiArrays which should be in
    itialized for results (0 for no initialization)
    tTransferTimeout : LTIME := LTIME#500US;          // timeout checking off during access to in
    ter-task FIFOs
END_VAR
```

入力パラメータ

このファンクションブロックの入力パラメータは初期化パラメータを表し、FBインスタンスの宣言で割り当て済みである必要があります。(または、Init()メソッド) 一度しか割り当てできません。実行中の変更はできません。

- stInitPars: [ST_CM_EmpiricalMoments_InitPars \[▶ 283\]](#) 型の初期化パラメータを持つファンクションブロック固有の構造体。パラメータは、上記の入出力バッファの定義と紐づいていなければなりません。
- nOwnID: 一意のIDでファンクションブロックインスタンスを識別します。これは常に0より大きくなければなりません。この目的のためには、列挙を定義するアプローチが確立されています。
- aDestIDs: 転送先のIDを指定することで、結果の転送先を定義します。出力バッファの定義（上記を参照）は、選択された各転送先の入力バッファの定義と紐づいていなければなりません。
- nResultBuffers: このファンクションブロックは、指定された数のマルチアレイバッファにより転送トレイストリームを初期化します。デフォルト値は4です。
- tTransferTimeout: 内部マルチアレイ転送の同期タイムアウトの設定です。[並列処理 \[▶ 72\]](#)の項を参照してください。

出力パラメータ

```
VAR_OUTPUT
    bError          : BOOL;                          // TRUE if an error occurs. Reset by next metho
    d call.
    hrErrorCode      : HRESULT;                      // '< 0' = error; '> 0' = info; '0' = no error/
    info
    ipErrorMessage  : I_TcMessage := fbErrorMessage; // Shows detailed information about occurred er
    rors, warnings and more.
    nCntResults      : ULINT;                        // Counts outgoing results (MultiArrays were ca
    lculated and sent to transfer tray).
END_VAR
```

- **bError**:エラーが発生した場合の出力は、TRUEです。
- **hrErrorCode**:エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。
- **ipErrorMessage**:現在の戻り値に関するより詳細な情報を含みます。[エラーの説明と情報 \[▶ 81\]](#)の項を参照してください。この特別なインターフェイスポインタは、常に有効かつ割り当て済みであるよう、内部的に保護されています。

メソッド

```
METHOD Init : HRESULT
VAR_INPUT
    stInitPars      : ST_CM_EmpiricalMoments_InitPars; // init parameter
    nOwnID           : UDINT;                          // ID for this FB instance
    aDestIDs         : ARRAY[1..cCMA_MaxDest] OF UDINT; // IDs of destinations for output
    nResultBuffers   : UDINT := 4;                     // number of MultiArrays which should be initialized for results (0 for no initialization)
END_VAR
```

Call():

このメソッドは、現在の入力データにアルゴリズムを適用するために、サイクルごとに関呼び出されます。メソッドが新しい結果もエラーも示さない場合、ファンクションブロックは入力データを待機します。これは、解析チェーンのプロセスにおける規則的な動作です。

- **戻り値**: エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD Call : HRESULT
VAR_OUTPUT
    bNewResult      : BOOL; // TRUE every time when outgoing MultiArray was calculated and sent to transfer tray.
    bError           : BOOL; // TRUE if an error occurs.
    hrErrorCode      : HRESULT; // '< 0' = error; '> 0' = info; '0' = no error/info
END_VAR
```

- **bError**:エラーが発生した場合の出力は、TRUEです。
- **hrErrorCode**:エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。この出力はメソッドの戻り値と同じです。



タイムアウトが発生した場合、または結果を格納するマルチアレイバッファがない場合、入力データも結果データも失われません。次の呼び出しに転送されます。

CallEx() :

このメソッドは、入力信号から内部メモリを更新するために、各サイクルで呼び出されます。結果は **nAppendData** サイクルごとに出力されます。別の方法としては、**Call()** があります。

メソッドが新しい結果もエラーも示さない場合、ファンクションブロックは入力データを待機します。これは、解析チェーンのプロセスにおける規則的な動作です。

- **戻り値**: エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD CallEx : HRESULT
VAR_INPUT
    nAppendData      : UDINT; // count of data buffers which are appended until calculation (1= calculate always)
    bResetData       : BOOL; // automatic reset of dataset buffer after each calculation
END_VAR
VAR_OUTPUT
    bNewResult       : BOOL; // TRUE every time when outgoing MultiArray was calculated and sent to transfer tray.
    bError            : BOOL; // TRUE if an error occurs.
    hrErrorCode       : HRESULT; // '< 0' = error; '> 0' = info; '0' = no error/info
END_VAR
```

- **nAppendData**:計算を実行する前に、いくつかの入力データバッファを収集するかを定義します。これは、正確な結果を得るために複数のデータブロックを追加することが望ましいためです。
- **bResetData**:これを設定した場合、内部データバッファは計算後に完全に削除されます。

- **bError:**エラーが発生した場合の出力は、TRUEです。
- **hrErrorCode:**エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。この出力はメソッドの戻り値と同じです。



タイムアウトが発生した場合、または結果を格納するマルチアレイバッファがない場合、入力データも結果データも失われません。次の呼び出しに転送されます。

Init():

このメソッドは通常、状態監視アプリケーションでは必要ありません。これはファンクションブロック初期化の代替手段を提供するものです。Init() メソッドは、PLCの初期化フェーズ中にのみ呼び出すことができます。実行時には使用できません。メソッドFB_init または、属性'call_after_init'の使用を参照してください（TwinCAT PLCリファレンスを参照）。さらに、これはファンクションブロックのカプセル化を容易にします。

Init()メソッドを使用して初期化を行う場合、ファンクションブロックインスタンスの入力パラメータを宣言で代入することはできません。

- **戻り値:** エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。
- **stInitPars:** `ST_CM_EmpiricalMoments_InitPars` [▶ 283] 型の初期化パラメータを持つファンクションブロック固有の構造体。パラメータは、上記の入出力バッファの定義と紐づいていなければなりません。
- **nOwnID:**一意のIDでファンクションブロックインスタンスを識別します。これは常に0より大きくなければなりません。この目的のためには、列挙を定義するアプローチが確立されています。
- **aDestIDs:**転送先のIDを指定することで、結果の転送先を定義します。出力バッファの定義（上記を参照）は、選択された各転送先の入力バッファの定義と紐づいていなければなりません。
- **nResultBuffers:**このファンクションブロックは、指定された数のマルチアレイバッファにより転送ストリームを初期化します。デフォルト値は4です。

ResetData():

このメソッドは、既に追加された全てのデータレコードを削除します。ファンクションブロックのメモリ特性を参照してください。ResetData() の後、Call() メソッドが再度呼び出された場合、有効な結果を計算するために内部メモリを補充する必要があります。

- **戻り値:** エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD ResetData : HRESULT
VAR_INPUT
END_VAR
```

または、CallEx() の自動リセットを使用することもできます。

PassInputs():

API PLC リファレンス [▶ 79]で説明されているとおり、FB_CMA_Source インスタンスが呼び出され、信号データがターゲットブロックに転送される限り、解析チェーンの以降の全てのファンクションブロックを周期的に呼び出す必要があります。

アルゴリズムを一定時間実行しない方が有用な場合があります。例えば、いくつかのアルゴリズムは、事前に教示や設定をした後でのみ実行すべきです。ファンクションブロックは周期的に呼び出す必要がありますが、ファンクションブロックに到着したデータが[通信リング \[▶ 72\]](#)内で転送されれば十分です。これは、Call() メソッドの代わりにPassInputs() メソッドを使用して行われます。アルゴリズム自体はここでは呼び出されず、したがって結果も計算されず、出力バッファも生成されません。

- **戻り値:** エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD PassInputs : HRESULT
VAR_INPUT
END_VAR
```

GetChannelErrors():

このメソッドは、複数のチャンネル（ $nChannels > 1$ ）を処理する際に、チャンネル固有の戻り値のリストを照会できるようにします。呼び出しは、ファンクションブロックの戻り値が `eCM_InfRTTime_AmbiguousChannelResults` または `eCM_ErrRTTime_ErrornousChannelResults` のいずれかに該当する場合に有効です。

- **戻り値:** エラーコード一覧の読み込み処理に関する情報です。この値は、クエリが成功した場合は TRUE に、そうでない場合は FALSE に設定されます。

```
METHOD GetChannelErrors : BOOL
VAR_IN_OUT
    aChannelErrors : ARRAY[*] OF HRESULT;
END_VAR
```

- `aChannelErrors:HRESULT` 型の長さ `nChannels` のエラーリストです。

類似したファンクションブロック

ファンクションブロック `FB_CMA_EmpiricalMean` [▶ 123] は、入力値の経験平均値を計算します。

ファンクションブロック `FB_CMA_EmpiricalSkew` [▶ 128] は、入力値の経験的歪度を計算します。

ファンクションブロック `FB_CMA_EmpiricalExcess` [▶ 118] は、入力値の経験的な超過分を計算します。

ファンクションブロック `FB_CMA_MomentCoefficients` [▶ 172] は、パラメータ設定に応じて、経験平均値、および標準偏差、歪度、過剰度を計算します。

`FB_CMA_HistArray` [▶ 145] ファンクションブロックは、入力値分布のヒストグラムを計算します。

`FB_CMA_Quantiles` [▶ 193] ファンクションブロックは、経験分布の分位数を計算し、外れ値の周波数を評価します。

尖度の代替として、`FB_CMA_CrestFactor` [▶ 93] ファンクションブロックは、信号のピーク性（クレスト係数）の別の尺度を計算しますが、これは異常値に対してより敏感です。

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM, Tc3_CM_Base



CM 3.1 では一部の機能がすでに使用可能です。[互換性](#) [▶ 61]の項を参照してください。

5.1.15 FB_CMA_Envelope

時間信号の包絡線を計算します。

包絡線は、数学的には解析信号の絶対値として定義されます（`FB_CMA_AnalyticSignal` [▶ 83] 参照）。時間連続表示では、信号 $x(t)$ の包絡線 $x_{env}(t)$ は次のように定義されます：

$$x_{env}(t) = |x_{analytic}(t)|, \quad x_{analytic}(t) = x(t) + i\mathcal{H}[x(t)]$$

包絡線は、信号 $x(t)$ の振幅変調成分として解釈できます。

$$x(t) = x_{env}(t) \cos \varphi(t)$$

位相変調された成分 $\varphi(t)$ も計算できます。`FB_CMA_InstantaneousPhase` を [▶ 153] を参照してください。包絡線は、上昇または減衰のプロセスを効率的に評価するために使用できます。

ファンクションブロックによる包絡線の離散計算は、周波数帯域で効率的に行われます。入力ベクトルは、まず直前のバッファとオーバーラップした後、ウィンドウ関数を適用して乗算されます。その後、実入力値に対するFFTが適用されます。周波数領域内で信号にヒルベルト変換が適用され、その結果が時間領域に変換されます。得られた複素数の絶対値が計算されます。時間信号はオーバーラップ加算法を使用して加算されます。適切なウィンドウ関数を選択することで、段階的な変化のない連続的な出力信号を得ることができます。

包絡線は、平均値のない信号に対してのみ有効な結果を提供します。平均値を持つ信号を解析する場合は、あらかじめ信号の平均値を減算しておき、ファンクションブロックの結果に先に減算した値を足し戻す必要があります。

● ウィンドウの長さに注意

i nWindowLength の値はnFFT_Length の値以下になるようにしてください。FFTの長さは、必要な周波数分解能に合わせるすることができます。一般的には、FFT長の約4/5の値がウィンドウ長としてよく使用されます。

nFFT_Length がnWindowLengthよりも大きい場合、FFTの周波数分解能が（したがって、戻り値ベクトルの長さも）増加します。長さの差は、フーリエ変換の前にゼロで満たされます。これは、より高い周波数分解能を得るためや、時間領域での逆変換による計算で円形エイリアシングを避けるために有用です。ただし、より高い周波数分解能にもかかわらず、結果にはそれ以上の情報は含まれていません。

メモリ特性

オーバーラップ加算法が使用されるため、計算には現在の入力バッファと直近に転送された2つのバッファが使用されます。

NaN（非数）の発生

入力ベクトルに1つ以上のNaN（非数）値が含まれる場合、出力セクタ全体がNaNで満たされます。詳細については別項NaN値 [▶ 71]を参照してください。

● NaN値の取り扱い

i NaN値につながる上記の状況を除外できない、または安全に無視できない場合、アプリケーションプログラムはこれらのエラー値を適切に処理できるようにしてください。

マルチチャンネル入力データ処理時の動作

複数のチャンネル（nChannels > 1）を処理する場合、各チャンネルが異なる戻り値になる可能性があります。この場合、戻り値はファンクションブロックで個別に問い合わせできます。全てのチャンネルではなく、1つ以上のチャンネルの結果が許容されない場合、ファンクションブロックの値はeCM_InfRTime_AmbiguousChannelResultsに対応します。全てのチャンネルの結果が許容されない場合、ファンクションブロック上の値はeCM_ErrRTime_ErrornousChannelResults に対応します。

全チャンネルの戻り値のリストは、GetChannelErrors()のメソッドで照会できます。

入力と出力

入力バッファと出力バッファは、以下の定義（入力/出力形状）のいずれかに該当します。変数パラメータは、ファンクションブロック入力stInitPars の一部です。

バージョン	入力バッファ（マルチアレイ入力ストリーム） 要素タイプ、次元数、次元サイズ	出力バッファ（マルチアレイ出力ストリーム） 要素タイプ、次元数、次元サイズ
標準バージョン (nChannels = 1)	LREAL, 1, nWindowLength/2	LREAL, 1, nWindowLength/2
マルチチャンネルバージョン (nChannels > 1)	LREAL, 2, nChannels x nWindowLength/2	LREAL, 2, nChannels x nWindowLength/2

```
VAR_INPUT
    stInitPars      : ST_CM_Envelope_InitPars;           // init parameter
    nOwnID          : UDINT;                             // ID for this FB instance
    aDestIDs        : ARRAY[1..cCMA_MaxDest] OF UDINT;   // IDs of destinations for output
    nResultBuffers  : UDINT := 4;                       // number of MultiArrays which should be in
```

```

ialized for results (0 for no initialization)
    tTransferTimeout : LTIME := LTIME#500US;           // timeout checking off during access to in
ter-task FIFOs
END_VAR

```

入力パラメータ

このファンクションブロックの入力パラメータは初期化パラメータを表し、FBインスタンスの宣言で割り当て済みである必要があります。(または、Init()メソッド) 一度しか割り当てできません。実行中の変更はできません。

- **stInitPars:** `ST_CM_Envelope_InitPars` [▶ 284]型の初期化パラメータを持つファンクションブロック固有の構造体。パラメータは、上記の入出力バッファの定義と紐づいていなければなりません。
- **nOwnID:**一意のIDでファンクションブロックインスタンスを識別します。これは常に0より大きくなければなりません。この目的のためには、列挙を定義するアプローチが確立されています。
- **aDestIDs:**転送先のIDを指定することで、結果の転送先を定義します。出力バッファの定義（上記を参照）は、選択された各転送先の入力バッファの定義と紐づいていなければなりません。
- **nResultBuffers:**このファンクションブロックは、指定された数のマルチアレイバッファにより転送トレイトリウムを初期化します。デフォルト値は4です。
- **tTransferTimeout:**内部マルチアレイ転送の同期タイムアウトの設定です。[並列処理](#) [▶ 72]の項を参照してください。

出力パラメータ

```

VAR_OUTPUT
    bError          : BOOL;           // TRUE if an error occurs. Reset by next metho
d call.
    hrErrorCode      : HRESULT;       // '< 0' = error; '> 0' = info; '0' = no error/
info
    ipErrorMessage  : I_TcMessage := fbErrorMessage; // Shows detailed information about occurred er
rors, warnings and more.
    nCntResults      : ULINT;         // Counts outgoing results (MultiArrays were ca
lculated and sent to transfer tray).
END_VAR

```

- **bError:**エラーが発生した場合の出力は、TRUEです。
- **hrErrorCode:**エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧](#) [▶ 367]に記載されています。
- **ipErrorMessage:**現在の戻り値に関するより詳細な情報を含みます。[エラーの説明と情報](#) [▶ 81]の項を参照してください。この特別なインターフェースポイントは、常に有効かつ割り当て済みであるよう、内部的に保護されています。

メソッド

```

METHOD Init : HRESULT
VAR_INPUT
    stInitPars      : ST_CM_Envelope_InitPars;           // init parameter
    nOwnID           : UDINT;                             // ID for this FB instance
    aDestIDs         : ARRAY[1..cCMA_MaxDest] OF UDINT;  // IDs of destinations for output
    nResultBuffers   : UDINT := 4;                       // number of MultiArrays which should be init
ialized for results (0 for no initialization)
END_VAR

```

Call():

このメソッドは、現在の入力データにアルゴリズムを適用するために、サイクルごとに呼び出されます。メソッドが新しい結果もエラーも示さない場合、ファンクションブロックは入力データを待機します。これは、解析チェーンのプロセスにおける規則的な動作です。

- **戻り値:** エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧](#) [▶ 367]に記載されています。

```

METHOD Call : HRESULT
VAR_OUTPUT
    bNewResult      : BOOL;           // TRUE every time when outgoing MultiArray was calculated and sent t
o transfer tray.
    bError          : BOOL;           // TRUE if an error occurs.
    hrErrorCode      : HRESULT;       // '< 0' = error; '> 0' = info; '0' = no error/info
END_VAR

```

- **bError:**エラーが発生した場合の出力は、TRUEです。

- **hrErrorCode**:エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。この出力はメソッドの戻り値と同じです。



タイムアウトが発生した場合、または結果を格納するマルチアレイバッファがない場合、入力データも結果データも失われません。次の呼び出しに転送されます。

Init():

このメソッドは通常、状態監視アプリケーションでは必要ありません。これはファンクションブロック初期化の代替手段を提供するものです。Init() メソッドは、PLCの初期化フェーズ中にのみ呼び出すことができます。実行時には使用できません。メソッドFB_init または、属性'call_after_init' の使用を参照してください（TwinCAT PLCリファレンスを参照）。さらに、これはファンクションブロックのカプセル化を容易にします。

Init()メソッドを使用して初期化を行う場合、ファンクションブロックインスタンスの入力パラメータを宣言で代入することはできません。

- **戻り値**: エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。
- **stInitPars**: `ST_CM_Envelope_InitPars` [▶ 284]型の初期化パラメータを持つファンクションブロック固有の構造体。パラメータは、上記の入出力バッファの定義と紐づいていなければなりません。
- **nOwnID**: 一意のIDでファンクションブロックインスタンスを識別します。これは常に0より大きくなければなりません。この目的のためには、列挙を定義するアプローチが確立されています。
- **aDestIDs**: 転送先のIDを指定することで、結果の転送先を定義します。出力バッファの定義（上記を参照）は、選択された各転送先の入力バッファの定義と紐づいていなければなりません。
- **nResultBuffers**: このファンクションブロックは、指定された数のマルチアレイバッファにより転送ストリームを初期化します。デフォルト値は4です。

ResetData() :

このメソッドは、既に追加された全てのデータレコードを削除します。ファンクションブロックのメモリ特性を参照してください。ResetData() の後、Call() メソッドが再度呼び出された場合、有効な結果を計算するために内部メモリを補充する必要があります。

- **戻り値**: エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD ResetData : HRESULT
VAR_INPUT
END_VAR
```

PassInputs() :

API PLC リファレンス [▶ 79]で説明されているとおり、FB_CMA_Source インスタンスが呼び出され、信号データがターゲットブロックに転送される限り、解析チェーンの以降の全てのファンクションブロックを周期的に呼び出す必要があります。

アルゴリズムを一定時間実行しない方が有用な場合があります。例えば、いくつかのアルゴリズムは、事前に教示や設定をした後でのみ実行すべきです。ファンクションブロックは周期的に呼び出す必要がありますが、ファンクションブロックに到着したデータが[通信リング \[▶ 72\]](#)内で転送されれば十分です。これは、Call() メソッドの代わりにPassInputs() メソッドを使用して行われます。アルゴリズム自体はここでは呼び出されず、したがって結果も計算されず、出力バッファも生成されません。

- **戻り値**: エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD PassInputs : HRESULT
VAR_INPUT
END_VAR
```

GetChannelErrors() :

このメソッドは、複数のチャンネル（nChannels > 1）を処理する際に、チャンネル固有の戻り値のリストを照会できるようにします。呼び出しは、ファンクションブロックの戻り値がeCM_InfRTIME_AmbiguousChannelResults またはeCM_ErrRTIME_ErrornousChannelResults のいずれかに該当する場合に有効です。

- **戻り値:** エラーコード一覧の読み込み処理に関する情報です。この値は、クエリが成功した場合は TRUE に、そうでない場合は FALSE に設定されます。

```

METHOD GetChannelErrors : BOOL
VAR_IN_OUT
    aChannelErrors : ARRAY[*] OF HRESULT;
END_VAR

```

- aChannelErrors: HRESULT 型の長さ nChannels のエラーリストです。

類似したファンクションブロック

FB_CMA_AnalyticSignal [▶ 83] ブロックは、時系列データの解析信号を計算します。

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM, Tc3_CM_Base



CM 3.1 では一部の機能がすでに使用可能です。[互換性 \[▶ 61\]](#)の項を参照してください。

5.1.16 FB_CMA_EnvelopeSpectrum

時間信号の包絡線スペクトルを計算します。

包絡線スペクトルは、[FB_CMA_Envelope \[▶ 137\]](#) と [FB_CMA_PowerSpectrum \[▶ 189\]](#) の複合ファンクションブロックです。したがって、まず時間信号の包絡線が計算され、次にパワースペクトルが計算されます。このファンクションブロックは、ローラーベアリングの損傷を周波数分解して解析する際に非常に重要です（[ベアリングの監視 \[▶ 38\]](#) を参照）。

信号包絡線（エンベロープ）の計算に内部的に使用されるウィンドウ長とFFTの長さは、パワースペクトルの計算に関連し、選択されたウィンドウ関数とそのオーバーラップ（nOverlap）に応じて決定されます。ウィンドウ長のルールは $L = 2 * (nWindowLength - nOverlap)$ です。FFTの長さNは、次に大きい2の累乗に対応します： $N \geq 4 * L / 3$ 。



ウィンドウの長さに注意

nWindowLength の値は nFFT_Length の値以下になるようにしてください。FFTの長さは、必要な周波数分解能に合わせるすることができます。一般的には、FFT長の約4/5の値がウィンドウ長としてよく使用されます。

nFFT_Length が nWindowLength よりも大きい場合、FFTの周波数分解能が（したがって、戻り値ベクトルの長さも）増加します。長さの差は、フーリエ変換の前にゼロで満たされます。これは、より高い周波数分解能を得るためや、時間領域での逆変換による計算で円形エイリアシングを避けるために有用です。ただし、より高い周波数分解能にもかかわらず、結果にはそれ以上の情報は含まれていません。

メモリ特性

オーバーラップ加算法が使用されるため、計算には現在の入力バッファと直近に転送された2つのバッファが使用されます。

また、パワースペクトルを計算する際に、オーバーラップの広いウィンドウ関数(nOverlap)を使用すると、この遅延効果が大きくなる場合があります。

NaN（非数）の発生

入力ベクトルに1つ以上のNaN（非数）値が含まれる場合、出力セクタ全体がNaNで満たされます。詳細については別項 [NaN値 \[▶ 71\]](#) を参照してください。

● NaN値の取り扱い

i NaN値につながる上記の状況を除外できない、または安全に無視できない場合、アプリケーションプログラムはこれらのエラー値を適切に処理できるようにしてください。

マルチチャンネル入力データ処理時の動作

複数のチャンネル ($nChannels > 1$) を処理する場合、各チャンネルが異なる戻り値になる可能性があります。この場合、戻り値はファンクションブロックで個別に問い合わせできます。全てのチャンネルではなく、1つ以上のチャンネルの結果が許容されない場合、ファンクションブロックの値は `eCM_InfRTTime_AmbiguousChannelResults` に対応します。全てのチャンネルの結果が許容されない場合、ファンクションブロック上の値は `eCM_ErrRTTime_ErrornousChannelResults` に対応します。

全チャンネルの戻り値のリストは、`GetChannelErrors()` のメソッドで照会できます。

サンプルの実装

実装用のサンプルは以下のリンクから入手できます：[包絡線スペクトル \[▶ 344\]](#)。

入力と出力

入力バッファと出力バッファは、以下の定義（入力/出力形状）のいずれかに該当します。変数パラメータは、ファンクションブロック入力 `stInitPars` の一部です。

バージョン	入力バッファ（マルチアレイ入力ストリーム） 要素タイプ、次元数、次元サイズ	出力バッファ（マルチアレイ出力ストリーム） 要素タイプ、次元数、次元サイズ
標準バージョン ($nChannels = 1$)	LREAL, 1, $nWindowLength - nOverlap$	LREAL, 1, $nFFT\_Length / 2 + 1$
マルチチャンネルバージョン ($nChannels > 1$)	LREAL, 2, $nChannels \times nWindowLength - nOverlap$	LREAL, 2, $nChannels \times nFFT\_Length / 2 + 1$

```
VAR_INPUT
    stInitPars      : ST_CM_EnvelopeSpectrum_InitPars;    // init parameter
    nOwnID          : UDINT;                               // ID for this FB instance
    aDestIDs        : ARRAY[1..cCMA_MaxDest] OF UDINT;    // IDs of destinations for output
    nResultBuffers  : UDINT := 4;                          // number of MultiArrays which should be
    initialized for results (0 for no initialization)
    tTransferTimeout : LTIME := LTIME#500US;              // timeout checking off during access to
    inter-task FIFOs
END_VAR
```

入力パラメータ

このファンクションブロックの入力パラメータは初期化パラメータを表し、FBインスタンスの宣言で割り当て済みである必要があります。（または、`Init()`メソッド）一度しか割り当てできません。実行中の変更はできません。

- `stInitPars`: `ST_CM_EnvelopeSpectrum_InitPars` [▶ 284]型の初期化パラメータを持つファンクションブロック固有の構造体。パラメータは、上記の入出力バッファの定義と紐づいていなければなりません。
- `nOwnID`: 一意のIDでファンクションブロックインスタンスを識別します。これは常に0より大きくなければなりません。この目的のためには、列挙を定義するアプローチが確立されています。
- `aDestIDs`: 転送先のIDを指定することで、結果の転送先を定義します。出力バッファの定義（上記を参照）は、選択された各転送先の入力バッファの定義と紐づいていなければなりません。
- `nResultBuffers`: このファンクションブロックは、指定された数のマルチアレイバッファにより転送トレイストリームを初期化します。デフォルト値は4です。
- `tTransferTimeout`: 内部マルチアレイ転送の同期タイムアウトの設定です。[並列処理 \[▶ 72\]](#)の項を参照してください。

出力パラメータ

```

VAR_OUTPUT
    bError      : BOOL;                // TRUE if an error occurs. Reset by next method call.
    hrErrorCode  : HRESULT;            // '< 0' = error; '> 0' = info; '0' = no error/info
    ipErrorMessage : I_TcMessage := fbErrorMessage; // Shows detailed information about occurred errors, warnings and more.
    nCntResults   : ULINT;             // Counts outgoing results (MultiArrays were calculated and sent to transfer tray).
END_VAR

```

- **bError:**エラーが発生した場合の出力は、TRUEです。
- **hrErrorCode:**エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。
- **ipErrorMessage:**現在の戻り値に関するより詳細な情報を含みます。[エラーの説明と情報 \[▶ 81\]](#)の項を参照してください。この特別なインターフェースポインタは、常に有効かつ割り当て済みであるよう、内部的に保護されています。

メソッド

```

METHOD Init : HRESULT
VAR_INPUT
    stInitPars      : ST_CM_EnvelopeSpectrum_InitPars; // init parameter
    nOwnID          : UDINT;                          // ID for this FB instance
    aDestIDs        : ARRAY[1..cCMA_MaxDest] OF UDINT; // IDs of destinations for output
    nResultBuffers  : UDINT := 4;                     // number of MultiArrays which should be initialized for results (0 for no initialization)
END_VAR

```

Call():

このメソッドは、現在の入力データにアルゴリズムを適用するために、サイクルごとに呼び出されます。メソッドが新しい結果もエラーも示さない場合、ファンクションブロックは入力データを待機します。これは、解析チェーンのプロセスにおける規則的な動作です。

- **戻り値:**エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```

METHOD Call : HRESULT
VAR_OUTPUT
    bNewResult      : BOOL;                // TRUE every time when outgoing MultiArray was calculated and sent to transfer tray.
    bError          : BOOL;                // TRUE if an error occurs.
    hrErrorCode      : HRESULT;            // '< 0' = error; '> 0' = info; '0' = no error/info
END_VAR

```

- **bError:**エラーが発生した場合の出力は、TRUEです。
- **hrErrorCode:**エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。この出力はメソッドの戻り値と同じです。



タイムアウトが発生した場合、または結果を格納するマルチアレイバッファがない場合、入力データも結果データも失われません。次の呼び出しに転送されます。

Init():

このメソッドは通常、状態監視アプリケーションでは必要ありません。これはファンクションブロック初期化の代替手段を提供するものです。Init() メソッドは、PLCの初期化フェーズ中にのみ呼び出すことができます。実行時には使用できません。メソッドFB_init または、属性'call_after_init'の使用を参照してください（TwinCAT PLCリファレンスを参照）。さらに、これはファンクションブロックのカプセル化を容易にします。

Init()メソッドを使用して初期化を行う場合、ファンクションブロックインスタンスの入力パラメータを宣言で代入することはできません。

- **戻り値:**エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。
- **stInitPars:**ST_CM_EnvelopeSpectrum_InitPars [▶ 284]型の初期化パラメータを持つファンクションブロック固有の構造体。パラメータは、上記の入出力バッファの定義と紐づいていなければなりません。

- **nOwnID**:一意のIDでファンクションブロックインスタンスを識別します。これは常に0より大きくなければなりません。この目的のためには、列挙を定義するアプローチが確立されています。
- **aDestIDs**:転送先のIDを指定することで、結果の転送先を定義します。出力バッファの定義（上記を参照）は、選択された各転送先の入力バッファの定義と紐づいていなければなりません。
- **nResultBuffers**:このファンクションブロックは、指定された数のマルチアレイバッファにより転送トレイストリームを初期化します。デフォルト値は4です。

ResetData() :

このメソッドは、既に追加された全てのデータレコードを削除します。ファンクションブロックのメモリ特性を参照してください。ResetData() の後、Call() メソッドが再度呼び出された場合、有効な結果を計算するために内部メモリを補充する必要があります。

- **戻り値** : エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD ResetData : HRESULT
VAR_INPUT
END_VAR
```

PassInputs() :

API PLC リファレンス [▶ 79]で説明されているとおり、FB_CMA_Source インスタンスが呼び出され、信号データがターゲットブロックに転送される限り、解析チェーンの以降の全てのファンクションブロックを周期的に呼び出す必要があります。

アルゴリズムを一定時間実行しない方が有用な場合があります。例えば、いくつかのアルゴリズムは、事前に教示や設定をした後でのみ実行すべきです。ファンクションブロックは周期的に呼び出す必要がありますが、ファンクションブロックに到着したデータが[通信リング \[▶ 72\]](#)内で転送されれば十分です。これは、Call() メソッドの代わりにPassInputs() メソッドを使用して行われます。アルゴリズム自体はここでは呼び出されず、したがって結果も計算されず、出力バッファも生成されません。

- **戻り値** : エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD PassInputs : HRESULT
VAR_INPUT
END_VAR
```

GetChannelErrors() :

このメソッドは、複数のチャンネル（nChannels > 1）を処理する際に、チャンネル固有の戻り値のリストを照会できるようにします。呼び出しは、ファンクションブロックの戻り値がeCM_InfRTTime_AmbiguousChannelResults またはeCM_ErrRTTime_ErrornousChannelResults のいずれかに該当する場合に有効です。

- **戻り値**: エラーコード一覧の読み込み処理に関する情報です。この値は、クエリが成功した場合はTRUE に、そうでない場合はFALSE に設定されます。

```
METHOD GetChannelErrors : BOOL
VAR_IN_OUT
    aChannelErrors : ARRAY[*] OF HRESULT;
END_VAR
```

- **aChannelErrors**: HRESULT 型の長さnChannelsのエラーリストです。

類似したファンクションブロック

ファンクションブロックFB_CMA_Envelope [▶ 137]は、時系列の包絡線を計算します。

FB_CMA_PowerSpectrum [▶ 189]ブロックは、最後のステップの値の2乗によりパワースペクトルを計算します。

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM, Tc3_CM_Base



CM 3.1 では一部の機能がすでに使用可能です。[互換性 \[▶ 61\]](#)の項を参照してください。

5.1.17 FB_CMA_HistArray

シングルチャンネルおよびマルチチャンネルの実数値時系列データの度数分布を計算します。

ファンクションブロックFB_CMA_HistArray は、シングルチャンネルおよびマルチチャンネルの実数値入力データの度数分布（グラフィック表示ではヒストグラムと呼ぶ）を計算します。各チャンネルは独立したチャンネルとして考慮されます。周期的に入力されるデータバッファの周波数分布は、各チャンネルごとに計算され、入力バッファには個々の値と配列の両方を使用できます。

下限値と上限値、およびクラスの数（ビンとも呼ぶ）は、パラメータ設定のために転送されます。各クラスの制限値は、このように定義された全範囲にわたって同間隔で分布します（[ヒストグラム \[▶ 27\]](#)を参照）。下限を下回る値と上限を上回る値は、それぞれ追加の2つのビンでカウントされます。

戻り値は、符号なし64ビット整数値の2次元配列です。最初のインデックスはチャンネルの番号で、2番目のインデックスはヒストグラムのビンの番号です。下限値以下または上限値以上の値を持つ要素のカウントは、それぞれ最初または最後のビンに含まれます。

ヒストグラムカウンタが2の64乗（約1.8E19）を超えた場合、現在の実装ではエラーメッセージを生成することなくオーバーフローします。これは、100マイクロ秒単位でカウントすると、早くても5900万年後に起こることになります。

メモリ特性

サンプル数 N は、新しいデータセットが入力されるたびに自動的に増加し、現在の結果を計算します。つまり、ファンクションブロックがインスタンス化されてからの全ての入力値を使用します。サンプル数のゼロリセット（FBの内部メモリの削除）は、ResetData() メソッド、またはCallEx() メソッドが使用される場合、変数bResetData によって提供されます。

設定

初期化パラメータは、通常のヒストグラムのビン内のサンプルをカウントするために制限を指定します。Configure() メソッドを使用して、各チャンネルごとに調整できます。

NaN（非数）の発生

入力値セットに特別な定数 NaN が含まれる場合、この時間ステップでは、チャンネルの統計に値は追加されません。つまり、NaN は欠損値の指標として扱われます。出力にはNaN値は含まれないことが想定されます。

マルチチャンネル入力データ処理時の動作

複数のチャンネル（ $nChannels > 1$ ）を処理する場合、各チャンネルが異なる戻り値になる可能性があります。この場合、戻り値はファンクションブロックで個別に問い合わせできます。全てのチャンネルではなく、1つ以上のチャンネルの結果が許容されない場合、ファンクションブロックの値はeCM_InfRTTime_AmbiguousChannelResultsに対応します。全てのチャンネルの結果が許容されない場合、ファンクションブロック上の値はeCM_ErrRTTime_ErrornousChannelResults に対応します。

全チャンネルの戻り値のリストは、GetChannelErrors()のメソッドで照会できます。

複数のサブチャンネル($nSubChannels > 0$)を処理する場合、入出力データのフォーマットには特に注意が必要です。入力データが上流のファンクションブロックによるマルチチャンネルの結果で構成されている場合、 $nChannels$ の値を採用する必要があります。この場合、パラメータ $nSubChannels$ を使用して追加の設定を行います。

例：マルチチャンネルスペクトル（FB_CMA_MagnitudeSpectrumなど）の周波数チャンネルの統計的考察（FB_CMA_Quantilesなど）では、 $nChannels$ の値は、入力信号の数と同じである必要があります。サブチャンネルの数 $nSubChannels$ は、スペクトルの長さに対応します。

サンプルの実装

実装用のサンプルは、以下のリンクから入手できます：[ヒストグラム \[▶ 325\]](#)

入力と出力

入力バッファと出力バッファは、以下の定義（形状）に該当します。変数パラメータは、ファンクションブロック入力stInitParsの一部です。

バージョン	入力バッファ（マルチアレイ入力ストリーム） 要素タイプ、次元数、次元サイズ	出力バッファ（マルチアレイ出力ストリーム） 要素タイプ、次元数、次元サイズ
標準バージョン (nSubChannels = 0)	LREAL, 1, nChannels	ULINT, 2, nChannels x nBins+2
複数データセット用標準バージョン (nSubChannels = 0)	LREAL, 2, nChannelsx 指定なし*	ULINT, 2, nChannels x nBins+2
マルチチャンネルバージョン (nSubChannels > 0)	LREAL, 2, nChannels x nSubChannels	ULINT, 3, nChannels x nSubChannels x nBins+2
複数データセット用マルチチャンネルバージョン (nSubChannels > 0)	LREAL, 3, nChannels x nSubChannels x 指 定なし*	ULINT, 3, nChannels x nSubChannels x nBins+2

*:この次元の長さは任意に選択できるため、アプリケーションや先行のアルゴリズムの出力バッファに合わせることができます。

```
VAR_INPUT
    stInitPars      : ST_CM_HistArray_InitPars;           // init parameter
    nOwnID          : UDINT;                             // ID for this FB instance
    aDestIDs        : ARRAY[1..cCMA_MaxDest] OF UDINT;    // IDs of destinations for output
    nResultBuffers  : UDINT := 4;                         // number of MultiArrays which should be in
    tTransferTimeout : LTIME := LTIME#500US;             // timeout checking off during access to in
    ter-task FIFOs
END_VAR
```

入力パラメータ

このファンクションブロックの入力パラメータは初期化パラメータを表し、FBインスタンスの宣言で割り当て済みである必要があります。（または、Init()メソッド）一度しか割り当てできません。実行中の変更はできません。

- **stInitPars:**ST_CM_HistArray_InitPars [▶ 285]型の初期化パラメータを持つファンクションブロック固有の構造体。パラメータは、上記の入出力バッファの定義と紐づいていなければなりません。
- **nOwnID:**一意のIDでファンクションブロックインスタンスを識別します。これは常に0より大きくなければなりません。この目的のためには、列挙を定義するアプローチが確立されています。
- **aDestIDs:**転送先のIDを指定することで、結果の転送先を定義します。出力バッファの定義（上記を参照）は、選択された各転送先の入力バッファの定義と紐づいていなければなりません。
- **nResultBuffers:**このファンクションブロックは、指定された数のマルチアレイバッファにより転送トレイストリームを初期化します。デフォルト値は4です。
- **tTransferTimeout:**内部マルチアレイ転送の同期タイムアウトの設定です。[並列処理 \[▶ 72\]](#)の項を参照してください。

出力パラメータ

```
VAR_OUTPUT
    bError          : BOOL;                             // TRUE if an error occurs. Reset by next metho
    d call.
    hrErrorCode      : HRESULT;                          // '< 0' = error; '> 0' = info; '0' = no error/
    info
    ipErrorMessage  : I_TcMessage := fbErrorMessage;    // Shows detailed information about occurred er
    rors, warnings and more.
    nCntResults      : ULINT;                             // Counts outgoing results (MultiArrays were ca
    lculated and sent to transfer tray).
END_VAR
```

- **bError:**エラーが発生した場合の出力は、TRUEです。

- **hrErrorCode**:エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。
- **ipErrorMessage**:現在の戻り値に関するより詳細な情報を含みます。[エラーの説明と情報 \[▶ 81\]](#)の項を参照してください。この特別なインターフェースポインタは、常に有効かつ割り当て済みであるよう、内部的に保護されています。

メソッド

Call():

このメソッドは、現在の入力データにアルゴリズムを適用するために、サイクルごとに呼び出されます。メソッドが新しい結果もエラーも示さない場合、ファンクションブロックは入力データを待機します。これは、解析チェーンのプロセスにおける規則的な動作です。

- **戻り値**:エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD Call : HRESULT
VAR_OUTPUT
    bNewResult : BOOL;          // TRUE every time when outgoing MultiArray was calculated and sent to transfer tray.
    bError      : BOOL;          // TRUE if an error occurs.
    hrErrorCode : HRESULT;       // '< 0' = error; '> 0' = info; '0' = no error/info
END_VAR
```

- **bError**:エラーが発生した場合の出力は、TRUEです。
- **hrErrorCode**:エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。この出力はメソッドの戻り値と同じです。



タイムアウトが発生した場合、または結果を格納するマルチアレイバッファがない場合、入力データも結果データも失われません。次の呼び出しに転送されます。

CallEx() :

このメソッドは、各サイクルで入力信号からヒストグラムを計算するために呼び出されます。別の方法としてCall()があります。

ヒストグラムの評価は、一般的に新しい入力値の登録よりも計算負荷が大きくなります。したがって、CallEx()の方法を使用すると、必要なときだけ統計結果を計算し、設定パラメータによっては実行時間をかなり短縮することができます。

メソッドが新しい結果もエラーも示さない場合、ファンクションブロックは入力データを待機します。これは、解析チェーンのプロセスにおける規則的な動作です。

- **戻り値**:エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD CallEx : HRESULT
VAR_INPUT
    nAppendData : UDINT;        // count of data buffers which are appended until calculation (1= calculate always)
    bResetData   : BOOL;        // automatic reset of dataset buffer after each calculation
END_VAR
VAR_OUTPUT
    bNewResult : BOOL;          // TRUE every time when outgoing MultiArray was calculated and sent to transfer tray.
    bError      : BOOL;          // TRUE if an error occurs.
    hrErrorCode : HRESULT;       // '< 0' = error; '> 0' = info; '0' = no error/info
END_VAR
```

- **nAppendData**:計算を実行する前に、いくつかの入力データバッファを収集するかを定義します。これは、正確な結果を得るために複数のデータブロックを追加することが望ましいためです。
- **bResetData**:これを設定した場合、内部データバッファは計算後に完全に削除されます。
- **bError**:エラーが発生した場合の出力は、TRUEです。
- **hrErrorCode**:エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。この出力はメソッドの戻り値と同じです。



タイムアウトが発生した場合、または結果を格納するマルチアレイバッファがない場合、入力データも結果データも失われません。次の呼び出しに転送されます。

Configure() :

このメソッドを呼び出すことで、ヒストグラムの引数を再設定できます。これにより、ランタイム時に fMinBinned と fMaxBinned のパラメータを微調整できます。対応する PLC 配列は以下のように定義してください。

- **戻り値** : エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、エラーコード一覧に記載されています。

```
METHOD Configure : HRESULT
VAR_INPUT
    pArg      : POINTER TO LREAL; // pointer to 2-dimensional array (LREAL) of arguments
    nArgSize  : UDINT;           // size of arguments buffer in bytes
END_VAR
```

入力バッファは、以下の定義（入力形状）のいずれかに紐づいています。変数パラメータは、ファンクションブロック入力 stInitPars の一部です。チャンネルとサブチャンネルごとに設定する2つのパラメータは、[fMinBinned, fMaxBinned] です。

バージョン	入力バッファ（マルチアレイ入カストリーム） 要素タイプ、次元数、次元サイズ
全てのチャンネルおよびサブチャンネルで同一設定	LREAL, 1, 2
チャンネル別の設定 (nSubChannels = 0)	LREAL, 2, nChannels x 2
サブチャンネル別の設定 (nSubChannels > 0)	LREAL, 2, nSubChannels x 2
チャンネルおよびサブチャンネル別の設定 (nSubChannels > 0)	LREAL, 3, nChannels x nSubChannels x 2

```
METHOD Init : HRESULT
VAR_INPUT
    stInitPars : ST_CM_HistArray_InitPars; // init parameter
    nOwnID     : UDINT; // ID for this FB instance
    aDestIDs   : ARRAY[1..cCMA_MaxDest] OF UDINT; // IDs of destinations for output
    nResultBuffers : UDINT := 4; // number of MultiArrays which should be initialized for results (
0 for no initialization)
END_VAR
```

Init():

このメソッドは通常、状態監視アプリケーションでは必要ありません。これはファンクションブロック初期化の代替手段を提供するものです。Init() メソッドは、PLCの初期化フェーズ中にのみ呼び出すことができます。実行時には使用できません。メソッド FB_init または、属性 'call_after_init' の使用を参照してください（TwinCAT PLCリファレンスを参照）。さらに、これはファンクションブロックのカプセル化を容易にします。

Init()メソッドを使用して初期化を行う場合、ファンクションブロックインスタンスの入力パラメータを宣言で代入することはできません。

- **戻り値** : エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。
- **stInitPars**: [ST_CM_HistArray_InitPars \[▶ 285\]](#)型の初期化パラメータを持つファンクションブロック固有の構造体。パラメータは、上記の入出力バッファの定義と紐づいていなければなりません。
- **nOwnID**: 一意のIDでファンクションブロックインスタンスを識別します。これは常に0より大きくなければなりません。この目的のためには、列挙を定義するアプローチが確立されています。
- **aDestIDs**: 転送先のIDを指定することで、結果の転送先を定義します。出力バッファの定義（上記を参照）は、選択された各転送先の入力バッファの定義と紐づいていなければなりません。
- **nResultBuffers**: このファンクションブロックは、指定された数のマルチアレイバッファにより転送トレイストリームを初期化します。デフォルト値は4です。

ResetData() :

このメソッドは、既に追加された全てのデータレコードを削除します。ファンクションブロックのメモリ特性を参照してください。ResetData() の後、Call() メソッドが再度呼び出された場合、有効な結果を計算するために内部メモリを補充する必要があります。

- **戻り値** : エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD ResetData : HRESULT
VAR_INPUT
END_VAR
```

または、CallEx() の自動リセットを使用することもできます。

PassInputs() :

API PLC リファレンス [▶ 79]で説明されているとおり、FB_CMA_Source インスタンスが呼び出され、信号データがターゲットブロックに転送される限り、解析チェーンの以降の全てのファンクションブロックを周期的に呼び出す必要があります。

アルゴリズムを一定時間実行しない方が有用な場合があります。例えば、いくつかのアルゴリズムは、事前に教示や設定をした後でのみ実行すべきです。ファンクションブロックは周期的に呼び出す必要がありますが、ファンクションブロックに到着したデータが通信リング [▶ 72]内で転送されれば十分です。これは、Call() メソッドの代わりにPassInputs() メソッドを使用して行われます。アルゴリズム自体はここでは呼び出されず、したがって結果も計算されず、出力バッファも生成されません。

- **戻り値** : エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD PassInputs : HRESULT
VAR_INPUT
END_VAR
```

GetChannelErrors() :

このメソッドは、複数のチャンネル（nChannels > 1）を処理する際に、チャンネル固有の戻り値のリストを照会できるようにします。呼び出しは、ファンクションブロックの戻り値が eCM_InfRTIME_AmbiguousChannelResults または eCM_ErrRTIME_ErrornousChannelResults のいずれかに該当する場合に有用です。

- **戻り値** : エラーコード一覧の読み込み処理に関する情報です。この値は、クエリが成功した場合は TRUE に、そうでない場合は FALSE に設定されます。

```
METHOD GetChannelErrors : BOOL
VAR_IN_OUT
    aChannelErrors : ARRAY[*] OF HRESULT;
END_VAR
```

- aChannelErrors:HRESULT 型の長さnChannelsのエラーリストです。

類似したファンクションブロック

[FB_CMA_Quantiles \[▶ 193\]](#)ファンクションブロックは、入力値分布の分位数を計算します。

[FB_CMA_MomentCoefficients \[▶ 172\]](#)ファンクションブロックは、統計モーメント係数（平均値、標準偏差、歪度、尖度）を計算します。

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM, Tc3_CM_Base



CM 3.1 では一部の機能がすでに使用可能です。[互換性 \[▶ 61\]](#)の項を参照してください。

5.1.18 FB_CMA_InstantaneousFrequency

時間信号の瞬時周波数を計算します。

数学的な意味での瞬時周波数は、瞬時位相の時間微分として定義されます

([FB_CMA_InstantaneousPhase](#) [▶ 153] を参照)。時間連続表現では、信号 $x(t)$ の瞬時周波数 $\omega(t)$ は次のように定義されます：

$$\omega(t) = \frac{d}{dt} \varphi(t) = \frac{d}{dt} \arctan \frac{\mathcal{H}[x(t)]}{x(t)}, \quad x_{\text{analytic}} = x(t) + i\mathcal{H}[x(t)]$$

瞬時周波数は、信号 $x(t)$ の周波数変調された成分として解釈できます。

$$x(t) = x_{\text{env}}(t) \cos \varphi(t) = x_{\text{env}}(t) \cos \int \omega(t) dt$$

このようにして、信号 $x(t)$ は、瞬時周波数と[FB_CMA_Envelope](#) [▶ 137] を計算することによって、振幅と周波数の変調表現に変換できます。

ファンクションブロックの瞬時位相と瞬時周波数は、平均値のない信号に対してのみ有効な結果を提供します。平均値を持つ信号を解析する場合は、あらかじめ信号の平均値を差し引く必要があります。

瞬時周波数は、クランクシャフトのねじり振動の解析に適しています。ねじり振動は、例えば変動するトルクによって引き起こされ、通常は一定である速度に周波数変調をもたらします。

● ウィンドウの長さに注意

i $n_{\text{WindowLength}}$ の値は $n_{\text{FFT\_Length}}$ の値以下になるようにしてください。FFTの長さは、必要な周波数分解能に合わせることができます。一般的には、FFT長の約4/5の値がウィンドウ長としてよく使用されます。

$n_{\text{FFT\_Length}}$ が $n_{\text{WindowLength}}$ よりも大きい場合、FFTの周波数分解能が（したがって、戻り値ベクトルの長さも）増加します。長さの差は、フーリエ変換の前にゼロで満たされます。これは、より高い周波数分解能を得るためや、時間領域での逆変換による計算で円形エイリアシングを避けるために有用です。ただし、より高い周波数分解能にもかかわらず、結果にはそれ以上の情報は含まれていません。

メモリ特性

オーバーラップ加算法が使用されるため、計算には現在の入力バッファと直近に転送された2つのバッファが使用されます。

NaN（非数）の発生

入力ベクトルに1つ以上のNaN（非数）値が含まれる場合、出力セクタ全体がNaNで満たされます。詳細については別項[NaN値](#) [▶ 71]を参照してください。

● NaN値の取り扱い

i NaN値につながる上記の状況を除外できない、または安全に無視できない場合、アプリケーションプログラムはこれらのエラー値を適切に処理できるようにしてください。

マルチチャンネル入力データ処理時の動作

複数のチャンネル ($n_{\text{Channels}} > 1$) を処理する場合、各チャンネルが異なる戻り値になる可能性があります。この場合、戻り値はファンクションブロックで個別に問い合わせできます。全てのチャンネルではなく、1つ以上のチャンネルの結果が許容されない場合、ファンクションブロックの値は `eCM_InfrTime_AmbiguousChannelResults` に対応します。全てのチャンネルの結果が許容されない場合、ファンクションブロック上の値は `eCM_ErrRTime_ErrornousChannelResults` に対応します。

全チャンネルの戻り値のリストは、`GetChannelErrors()`のメソッドで照会できます。

入力と出力

入力バッファと出力バッファは、以下の定義（入力/出力形状）のいずれかに該当します。変数パラメータは、ファンクションブロック入力stInitParsの一部です。

バージョン	入力バッファ（マルチアレイ入力ストリーム） 要素タイプ、次元数、次元サイズ	出力バッファ（マルチアレイ出力ストリーム） 要素タイプ、次元数、次元サイズ
標準バージョン (nChannels = 1)	LREAL, 1, nWindowLength/2	LREAL, 1, nWindowLength/2
マルチチャンネルバージョン (nChannels > 1)	LREAL, 2, nChannels x nWindowLength/2	LREAL, 2, nChannels x nWindowLength/2

```
VAR_INPUT
    stInitPars      : ST_CM_InstantaneousFrequency_InitPars; // init parameter
    nOwnID          : UDINT;                                // ID for this FB instance
    aDestIDs        : ARRAY[1..cCMA_MaxDest] OF UDINT;      // IDs of destinations for output
    nResultBuffers  : UDINT := 4;                            // number of MultiArrays which should
    be initialized for results (0 for no initialization)
    tTransferTimeout : LTIME := LTIME#500US;                // timeout checking off during access
    to inter-task FIFOs
END_VAR
```

入力パラメータ

このファンクションブロックの入力パラメータは初期化パラメータを表し、FBインスタンスの宣言で割り当て済みである必要があります。（または、Init()メソッド）一度しか割り当てできません。実行中の変更はできません。

- stInitPars: [ST_CM_InstantaneousFrequency_InitPars \[▶ 286\]](#) 型の初期化パラメータを持つファンクションブロック固有の構造体。パラメータは、上記の入出力バッファの定義と紐づいていなければなりません。
- nOwnID: 一意のIDでファンクションブロックインスタンスを識別します。これは常に0より大きくなければなりません。この目的のためには、列挙を定義するアプローチが確立されています。
- aDestIDs: 転送先のIDを指定することで、結果の転送先を定義します。出力バッファの定義（上記を参照）は、選択された各転送先の入力バッファの定義と紐づいていなければなりません。
- nResultBuffers: このファンクションブロックは、指定された数のマルチアレイバッファにより転送トレイストリームを初期化します。デフォルト値は4です。
- tTransferTimeout: 内部マルチアレイ転送の同期タイムアウトの設定です。[並列処理 \[▶ 72\]](#)の項を参照してください。

出力パラメータ

```
VAR_OUTPUT
    bError          : BOOL;                                // TRUE if an error occurs. Reset by next method call.
    hrErrorCode      : HRESULT;                             // '< 0' = error; '> 0' = info; '0' = no error/info
    ipErrorMessage  : I_TcMessage := fbErrorMessage; // Shows detailed information about occurred errors, warnings and more.
    nCntResults      : ULINT;                               // Counts outgoing results (MultiArrays were calculated and sent to transfer tray).
END_VAR
```

- bError: エラーが発生した場合の出力は、TRUEです。
- hrErrorCode: エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。
- ipErrorMessage: 現在の戻り値に関するより詳細な情報を含みます。[エラーの説明と情報 \[▶ 81\]](#)の項を参照してください。この特別なインターフェースポイントは、常に有効かつ割り当て済みであるよう、内部的に保護されています。

メソッド

```
METHOD Init : HRESULT
VAR_INPUT
    stInitPars      : ST_CM_InstantaneousFrequency_InitPars; // init parameter
    nOwnID          : UDINT;                                // ID for this FB instance
```



```

    aDestIDs      : ARRAY[1..cCMA_MaxDest] OF UDINT; // IDs of destinations for output
    nResultBuffers : UDINT := 4;                    // number of MultiArrays which should be initialized for results (0 for no initialization)
END_VAR

```

Call():

このメソッドは、現在の入力データにアルゴリズムを適用するために、サイクルごとに呼び出されます。メソッドが新しい結果もエラーも示さない場合、ファンクションブロックは入力データを待機します。これは、解析チェーンのプロセスにおける規則的な動作です。

- **戻り値**：エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```

METHOD Call : HRESULT
VAR_OUTPUT
    bNewResult      : BOOL;          // TRUE every time when outgoing MultiArray was calculated and sent to transfer tray.
    bError          : BOOL;          // TRUE if an error occurs.
    hrErrorCode      : HRESULT;      // '< 0' = error; '> 0' = info; '0' = no error/info
END_VAR

```

- **bError**:エラーが発生した場合の出力は、TRUEです。
- **hrErrorCode**:エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。この出力はメソッドの戻り値と同じです。



タイムアウトが発生した場合、または結果を格納するマルチアレイバッファがない場合、入力データも結果データも失われません。次の呼び出しに転送されます。

Init():

このメソッドは通常、状態監視アプリケーションでは必要ありません。これはファンクションブロック初期化の代替手段を提供するものです。Init() メソッドは、PLCの初期化フェーズ中にのみ呼び出すことができます。実行時には使用できません。メソッドFB_init または、属性'call_after_init' の使用を参照してください（TwinCAT PLCリファレンスを参照）。さらに、これはファンクションブロックのカプセル化を容易にします。

Init()メソッドを使用して初期化を行う場合、ファンクションブロックインスタンスの入力パラメータを宣言で代入することはできません。

- **戻り値**：エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。
- **stInitPars**:[ST_CM_InstantaneousFrequency_InitPars \[▶ 286\]](#) 型の初期化パラメータを持つファンクションブロック固有の構造体。パラメータは、上記の入出力バッファの定義と紐づいていなければなりません。
- **nOwnID**:一意のIDでファンクションブロックインスタンスを識別します。これは常に0より大きくなければなりません。この目的のためには、列挙を定義するアプローチが確立されています。
- **aDestIDs**:転送先のIDを指定することで、結果の転送先を定義します。出力バッファの定義（上記を参照）は、選択された各転送先の入力バッファの定義と紐づいていなければなりません。
- **nResultBuffers**:このファンクションブロックは、指定された数のマルチアレイバッファにより転送ストリームを初期化します。デフォルト値は4です。

ResetData() :

このメソッドは、既に追加された全てのデータレコードを削除します。ファンクションブロックのメモリ特性を参照してください。ResetData() の後、Call() メソッドが再度呼び出された場合、有効な結果を計算するために内部メモリを補充する必要があります。

- **戻り値**：エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```

METHOD ResetData : HRESULT
VAR_INPUT
END_VAR

```

PassInputs() :

API PLC リファレンス [▶ 79]で説明されているとおり、FB_CMA_Source インスタンスが呼び出され、信号データがターゲットブロックに転送される限り、解析チェーンの以降の全てのファンクションブロックを周期的に呼び出す必要があります。

アルゴリズムを一定時間実行しない方が有用な場合があります。例えば、いくつかのアルゴリズムは、事前に教示や設定をした後でのみ実行すべきです。ファンクションブロックは周期的に呼び出す必要がありますが、ファンクションブロックに到着したデータが通信リング [▶ 72]内で転送されれば十分です。これは、Call() メソッドの代わりにPassInputs() メソッドを使用して行われます。アルゴリズム自体はここでは呼び出されず、したがって結果も計算されず、出力バッファも生成されません。

- **戻り値**：エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD PassInputs : HRESULT
VAR_INPUT
END_VAR
```

GetChannelErrors() :

このメソッドは、複数のチャンネル（nChannels > 1）を処理する際に、チャンネル固有の戻り値のリストを照会できるようにします。呼び出しは、ファンクションブロックの戻り値が eCM_InfRTTime_AmbiguousChannelResults または eCM_ErrRTTime_ErrornousChannelResults のいずれかに該当する場合に有用です。

- **戻り値**：エラーコード一覧の読み込み処理に関する情報です。この値は、クエリが成功した場合は TRUE に、そうでない場合は FALSE に設定されます。

```
METHOD GetChannelErrors : BOOL
VAR_IN_OUT
aChannelErrors : ARRAY[*] OF HRESULT;
END_VAR
```

- aChannelErrors: HRESULT 型の長さnChannelsのエラーリストです。

類似したファンクションブロック

ファンクションブロック [FB_CMA_InstantaneousPhase \[▶ 153\]](#) は、時間信号の瞬時位相を計算します。

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM, Tc3_CM_Base



CM 3.1 では一部の機能がすでに使用可能です。[互換性 \[▶ 61\]](#)の項を参照してください。

5.1.19 FB_CMA_InstantaneousPhase

時間信号の瞬時位相を計算します。

信号 $x(t)$ の瞬時位相 $\varphi(t)$ は、解析信号の位相によって定義されます（[FB_CMA_AnalyticSignal \[▶ 83\]](#)を参照）：

$$\varphi(t) = \arctan \frac{\mathcal{H}[x(t)]}{x(t)}, \quad x_{\text{analytic}} = x(t) + i\mathcal{H}[x(t)]$$

瞬時位相は、信号 $x(t)$ の位相変調成分と解釈できます：

$$x(t) = x_{\text{env}}(t) \cos \varphi(t)$$

信号の振幅変調成分（包絡線）も計測できます（[FB_CMA_Envelope \[▶ 137\]](#)を参照）。

ファンクションブロックの瞬時位相と瞬時周波数は、平均値のない信号に対してのみ有効な結果を提供します。平均値を持つ信号を解析する場合は、あらかじめ信号の平均値を差し引く必要があります。

● ウィンドウの長さに注意

i nWindowLength の値はnFFT_Length の値以下になるようにしてください。FFTの長さは、必要な周波数分解能に合わせることができます。一般的には、FFT長の約4/5の値がウィンドウ長としてよく使用されます。

nFFT_Length がnWindowLengthよりも大きい場合、FFTの周波数分解能が（したがって、戻り値ベクトルの長さも）増加します。長さの差は、フーリエ変換の前にゼロで満たされます。これは、より高い周波数分解能を得るためや、時間領域での逆変換による計算で円形エイリアシングを避けるために有用です。ただし、より高い周波数分解能にもかかわらず、結果にはそれ以上の情報は含まれていません。

メモリ特性

オーバーラップ加算法が使用されるため、計算には現在の入力バッファと直近に転送された2つのバッファが使用されます。

NaN（非数）の発生

入力ベクトルに1つ以上のNaN（非数）値が含まれる場合、出力セクタ全体がNaNで満たされます。詳細については別項NaN値 [▶ 71]を参照してください。

● NaN値の取り扱い

i NaN値につながる上記の状況を除外できない、または安全に無視できない場合、アプリケーションプログラムはこれらのエラー値を適切に処理できるようにしてください。

マルチチャンネル入力データ処理時の動作

複数のチャンネル（nChannels > 1）を処理する場合、各チャンネルが異なる戻り値になる可能性があります。この場合、戻り値はファンクションブロックで個別に問い合わせできます。全てのチャンネルではなく、1つ以上のチャンネルの結果が許容されない場合、ファンクションブロックの値はeCM_InfRTIME_AmbiguousChannelResultsに対応します。全てのチャンネルの結果が許容されない場合、ファンクションブロック上の値はeCM_ErrRTIME_ErrornousChannelResults に対応します。

全チャンネルの戻り値のリストは、GetChannelErrors()のメソッドで照会できます。

入力と出力

入力バッファと出力バッファは、以下の定義（入力/出力形状）のいずれかに該当します。変数パラメータは、ファンクションブロック入力stInitPars の一部です。

バージョン	入力バッファ（マルチアレイ入力ストリーム） 要素タイプ、次元数、次元サイズ	出力バッファ（マルチアレイ出力ストリーム） 要素タイプ、次元数、次元サイズ
標準バージョン (nChannels = 1)	LREAL, 1, nWindowLength/2	LREAL, 1, nWindowLength/2
マルチチャンネルバージョン (nChannels > 1)	LREAL, 2, nChannels x nWindowLength/2	LREAL, 2, nChannels x nWindowLength/2

```

VAR_INPUT
    stInitPars      : ST_CM_InstantaneousPhase_InitPars; // init parameter
    nOwnID          : UDINT;                             // ID for this FB instance
    aDestIDs        : ARRAY[1..cCMA_MaxDest] OF UDINT;    // IDs of destinations for output
    nResultBuffers  : UDINT := 4;                         // number of MultiArrays which should be
initialised for results (0 for no initialization)
    tTransferTimeout : LTIME := LTIME#500US;             // timeout checking off during access to
inter-task FIFOs
END_VAR

```

入力パラメータ

このファンクションブロックの入力パラメータは初期化パラメータを表し、FBインスタンスの宣言で割り当て済みである必要があります。（または、Init()メソッド）一度しか割り当てできません。実行中の変更はできません。

- **stInitPars:** `ST_CM_InstantaneousPhase_InitPars` [▶ 286]型の初期化パラメータを持つファンクションブロック固有の構造体。パラメータは、上記の入出力バッファの定義と紐づいていなければなりません。
- **nOwnID:**一意のIDでファンクションブロックインスタンスを識別します。これは常に0より大きくなければなりません。この目的のためには、列挙を定義するアプローチが確立されています。
- **aDestIDs:**転送先のIDを指定することで、結果の転送先を定義します。出力バッファの定義（上記を参照）は、選択された各転送先の入力バッファの定義と紐づいていなければなりません。
- **nResultBuffers:**このファンクションブロックは、指定された数のマルチアレイバッファにより転送トレイストリームを初期化します。デフォルト値は4です。
- **tTransferTimeout:**内部マルチアレイ転送の同期タイムアウトの設定です。[並列処理](#) [▶ 72]の項を参照してください。

出力パラメータ

```
VAR_OUTPUT
    bError      : BOOL;                // TRUE if an error occurs. Reset by next method call.
    hrErrorCode  : HRESULT;            // '< 0' = error; '> 0' = info; '0' = no error/info
    ipErrorMessage : I_TcMessage := fbErrorMessage; // Shows detailed information about occurred errors, warnings and more.
    nCntResults   : ULINT;             // Counts outgoing results (MultiArrays were calculated and sent to transfer tray).
END_VAR
```

- **bError:**エラーが発生した場合の出力は、TRUEです。
- **hrErrorCode:**エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧](#) [▶ 367]に記載されています。
- **ipErrorMessage:**現在の戻り値に関するより詳細な情報を含みます。[エラーの説明と情報](#) [▶ 81]の項を参照してください。この特別なインターフェイスポインタは、常に有効かつ割り当て済みであるよう、内部的に保護されています。

メソッド

```
METHOD Init : HRESULT
VAR_INPUT
    stInitPars      : ST_CM_InstantaneousPhase_InitPars; // init parameter
    nOwnID          : UDINT;                             // ID for this FB instance
    aDestIDs        : ARRAY[1..cCMA_MaxDest] OF UDINT; // IDs of destinations for output
    nResultBuffers  : UDINT := 4;                       // number of MultiArrays which should be initialized for results (0 for no initialization)
END_VAR
```

Call():

このメソッドは、現在の入力データにアルゴリズムを適用するために、サイクルごとに呼び出されます。メソッドが新しい結果もエラーも示さない場合、ファンクションブロックは入力データを待機します。これは、解析チェーンのプロセスにおける規則的な動作です。

- **戻り値:** エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧](#) [▶ 367]に記載されています。

```
METHOD Call : HRESULT
VAR_OUTPUT
    bNewResult      : BOOL;                // TRUE every time when outgoing MultiArray was calculated and sent to transfer tray.
    bError          : BOOL;                // TRUE if an error occurs.
    hrErrorCode      : HRESULT;            // '< 0' = error; '> 0' = info; '0' = no error/info
END_VAR
```

- **bError:**エラーが発生した場合の出力は、TRUEです。
- **hrErrorCode:**エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧](#) [▶ 367]に記載されています。この出力はメソッドの戻り値と同じです。



タイムアウトが発生した場合、または結果を格納するマルチアレイバッファがない場合、入力データも結果データも失われません。次の呼び出しに転送されます。

Init():

このメソッドは通常、状態監視アプリケーションでは必要ありません。これはファンクションブロック初期化の代替手段を提供するものです。Init() メソッドは、PLCの初期化フェーズ中にのみ呼び出すことができます。実行時には使用できません。メソッドFB_init または、属性'call_after_init' の使用を参照してください（TwinCAT PLCリファレンスを参照）。さらに、これはファンクションブロックのカプセル化を容易にします。

Init()メソッドを使用して初期化を行う場合、ファンクションブロックインスタンスの入力パラメータを宣言で代入することはできません。

- **戻り値**：エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。
- **stInitPars**:ST_CM InstantaneousPhase InitPars [▶ 286]型の初期化パラメータを持つファンクションブロック固有の構造体。パラメータは、上記の入出力バッファの定義と紐づいていなければなりません。
- **nOwnID**:一意のIDでファンクションブロックインスタンスを識別します。これは常に0より大きくなければなりません。この目的のためには、列挙を定義するアプローチが確立されています。
- **aDestIDs**:転送先のIDを指定することで、結果の転送先を定義します。出力バッファの定義（上記を参照）は、選択された各転送先の入力バッファの定義と紐づいていなければなりません。
- **nResultBuffers**:このファンクションブロックは、指定された数のマルチアレイバッファにより転送ストリームを初期化します。デフォルト値は4です。

ResetData() :

このメソッドは、既に追加された全てのデータレコードを削除します。ファンクションブロックのメモリ特性を参照してください。ResetData() の後、Call() メソッドが再度呼び出された場合、有効な結果を計算するために内部メモリを補充する必要があります。

- **戻り値**：エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD ResetData : HRESULT
VAR_INPUT
END_VAR
```

PassInputs() :

API PLC リファレンス [▶ 79]で説明されているとおり、FB_CMA_Source インスタンスが呼び出され、信号データがターゲットブロックに転送される限り、解析チェーンの以降の全てのファンクションブロックを周期的に呼び出す必要があります。

アルゴリズムを一定時間実行しない方が有用な場合があります。例えば、いくつかのアルゴリズムは、事前に教示や設定をした後でのみ実行すべきです。ファンクションブロックは周期的に呼び出す必要がありますが、ファンクションブロックに到着したデータが通信リング [▶ 72]内で転送されれば十分です。これは、Call() メソッドの代わりにPassInputs() メソッドを使用して行われます。アルゴリズム自体はここでは呼び出されず、したがって結果も計算されず、出力バッファも生成されません。

- **戻り値**：エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD PassInputs : HRESULT
VAR_INPUT
END_VAR
```

GetChannelErrors() :

このメソッドは、複数のチャンネル（nChannels > 1）を処理する際に、チャンネル固有の戻り値のリストを照会できるようにします。呼び出しは、ファンクションブロックの戻り値が eCM_InfRTTime_AmbiguousChannelResults または eCM_ErrRTTime_ErrornousChannelResults のいずれかに該当する場合に有効です。

- **戻り値**: エラーコード一覧の読み込み処理に関する情報です。この値は、クエリが成功した場合は TRUE に、そうでない場合は FALSE に設定されます。

```
METHOD GetChannelErrors : BOOL
VAR_IN_OUT
aChannelErrors : ARRAY[*] OF HRESULT;
END_VAR
```


- aChannelErrors:HRESULT 型の長さnChannelsのエラーリストです。

類似したファンクションブロック

ファンクションブロックFB_CMA_InstantaneousFrequency [▶ 150]は、時間信号の瞬時周波数を計算します。

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM, Tc3_CM_Base



CM 3.1 では一部の機能がすでに使用可能です。[互換性 \[▶ 61\]](#)の項を参照してください。

5.1.20 FB_CMA_IntegratedRMS

シングルチャンネルおよびマルチチャンネルの実数値時系列データのRMS値（オプションで積分）を計算します。

シングルチャンネルおよびマルチチャンネルの時系列データのRMS値を計算します。使用する周波数領域と時系列の積分順序の両方を定義できます。加速度信号の場合、振動加速度、振動速度、振動振幅のそれぞれについて、定義された周波数領域におけるRMS値が得られます。

このファンクションブロックは、入力信号を複数の独立したチャンネルを持つ信号として扱います。各チャンネルについて、周波数領域内で最大3つの異なる積分次数の値が、定義された周波数間隔で積分され、その後、RMS値が計算されます。ファンクションブロックは、DIN ISO 10816およびDIN ISO 7919またはDIN ISO 20816に準拠した振動評価に適しています（[振動評価 \[▶ 33\]](#) 参照）。

サンプリングレートと積分間隔の制限値はパラメータ設定できます。再現性のあるスケーリングを得るためには、入力信号と周波数をSI単位でスケーリングする必要があります。つまり、加速度値は $1\text{m}/(\text{s})^2$ 、周波数は $1/\text{s} = 1\text{Hz}$ です。積分の最大次数は0から2の間で設定できます。計算する積分RMS値の数は、 $(nOrder+1)$ で指定します。結果は、これらの値と対応するインデックスの配列として転送されます。

多くの場合、基礎となる短期的なパワースペクトルは、信号スペクトルの統計的推定には適していないため、周波数で平均化しても値が変動することがあります。したがって、十分な長さのウィンドウ長を使用することが望ましいです。加えて、多くの場合、複数の連続した値の幾何平均値を計算することで、変動を抑えるのが望ましいです。

メモリ特性

ウェルチ法を使用しているため、現在の入力データバッファと最後に転送されたバッファの両方が計算に使用されます。組み込まれるバッファの数は、選択したオーバーラップ(nOverlap)に依存します。周波数解析では、時系列データの段階的な変化が考慮されます。したがって、正しい結果を得るためには、使用する入力データバッファは、間隔やジャンプなしに連結していなければなりません。

NaN（非数）の発生

入力ベクトルに1つ以上のNaN（非数）値が含まれる場合、出力セクタ全体がNaNで満たされます。詳細については別項NaN値 [▶ 71]を参照してください。



NaN値の取り扱い

NaN値につながる上記の状況を除外できない、または安全に無視できない場合、アプリケーションプログラムはこれらのエラー値を適切に処理できるようにしてください。

マルチチャンネル入力データ処理時の動作

複数のチャンネル（ $nChannels > 1$ ）を処理する場合、各チャンネルが異なる戻り値になる可能性があります。この場合、戻り値はファンクションブロックで個別に問い合わせできます。全てのチャンネルではなく、1つ以上のチャンネルの結果が許容されない場合、ファンクションブロックの値は `eCM_InfrTime_AmbiguousChannelResults` に対応します。全てのチャンネルの結果が許容されない場合、ファンクションブロック上の値は `eCM_ErrTime_ErrornousChannelResults` に対応します。

全チャンネルの戻り値のリストは、`GetChannelErrors()`のメソッドで照会できます。

入力と出力

入力バッファと出力バッファは、以下の定義（入力/出力形状）のいずれかに該当します。変数パラメータは、ファンクションブロック入力 `stInitPars` の一部です。

バージョン	入力バッファ（マルチアレイ入力ストリーム） 要素タイプ、次元数、次元サイズ	出力バッファ（マルチアレイ出力ストリーム） 要素タイプ、次元数、次元サイズ
標準バージョン ($nMaxBands = 1$)	LREAL, 2, $nChannels \times nWindowLength - nOverlap$	LREAL, 2, $nChannels \times nOrder + 1$
複数の周波数帯に対応したバージョン ($nMaxBands > 1$)	LREAL, 2, $nChannels \times nWindowLength - nOverlap$	LREAL, 3, $nChannels \times nOrder + 1 \times nMaxBands$

```
VAR_INPUT
    stInitPars      : ST_CM_IntegratedRMS_InitPars;    // init parameter
    nOwnID          : UDINT;                          // ID for this FB instance
    aDestIDs        : ARRAY[1..cCMA_MaxDest] OF UDINT; // IDs of destinations for output
    nResultBuffers  : UDINT := 4;                    // number of MultiArrays which should be in
    itialized for results (0 for no initialization)
    tTransferTimeout : LTIME := LTIME#500US;         // timeout checking off during access to in
    ter-task FIFOs
END_VAR
```

入力パラメータ

このファンクションブロックの入力パラメータは初期化パラメータを表し、FBインスタンスの宣言で割り当て済みである必要があります。（または、`Init()`メソッド）一度しか割り当てできません。実行中の変更はできません。

- `stInitPars:ST_CM_IntegratedRMS_InitPars` [▶ 287]型の初期化パラメータを持つファンクションブロック固有の構造体。パラメータは、上記の入出力バッファの定義と紐づいていなければなりません。
- `nOwnID`:一意のIDでファンクションブロックインスタンスを識別します。これは常に0より大きくなければなりません。この目的のためには、列挙を定義するアプローチが確立されています。
- `aDestIDs`:転送先のIDを指定することで、結果の転送先を定義します。出力バッファの定義（上記を参照）は、選択された各転送先の入力バッファの定義と紐づいていなければなりません。
- `nResultBuffers`:このファンクションブロックは、指定された数のマルチアレイバッファにより転送トレイストリームを初期化します。デフォルト値は4です。
- `tTransferTimeout`:内部マルチアレイ転送の同期タイムアウトの設定です。[並列処理](#) [▶ 72]の項を参照してください。

出力パラメータ

```
VAR_OUTPUT
    bError          : BOOL;                          // TRUE if an error occurs. Reset by next metho
    d call.
    hrErrorCode      : HRESULT;                      // '< 0' = error; '> 0' = info; '0' = no error/
    info
    ipErrorMessage  : I_TcMessage := fbErrorMessage; // Shows detailed information about occurred er
    rors, warnings and more.
    nCntResults      : ULINT;                        // Counts outgoing results (MultiArrays were ca
    lculated and sent to transfer tray).
END_VAR
```

- `bError`:エラーが発生した場合の出力は、TRUEです。

- **hrErrorCode**:エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。
- **ipErrorMessage**:現在の戻り値に関するより詳細な情報を含みます。[エラーの説明と情報 \[▶ 81\]](#)の項を参照してください。この特別なインターフェースポインタは、常に有効かつ割り当て済みであるよう、内部的に保護されています。

メソッド

Call():

このメソッドは、現在の入力データにアルゴリズムを適用するために、サイクルごとに呼び出されます。メソッドが新しい結果もエラーも示さない場合、ファンクションブロックは入力データを待機します。これは、解析チェーンのプロセスにおける規則的な動作です。

- **戻り値**:エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD Call : HRESULT
VAR_OUTPUT
    bNewResult : BOOL;          // TRUE every time when outgoing MultiArray was calculated and sent to transfer tray.
    bError      : BOOL;          // TRUE if an error occurs.
    hrErrorCode : HRESULT;       // '< 0' = error; '> 0' = info; '0' = no error/info
END_VAR
```

- **bError**:エラーが発生した場合の出力は、TRUEです。
- **hrErrorCode**:エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。この出力はメソッドの戻り値と同じです。



タイムアウトが発生した場合、または結果を格納するマルチアレイバッファがない場合、入力データも結果データも失われません。次の呼び出しに転送されます。

Configure():

このメソッドを呼び出す際には、最初に周波数帯域を設定してください。対応するPLC配列は以下のように定義してください。Configure()メソッドは、異なる引数セットを持つ新しい設定にも使用できます。

- **戻り値**:エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD Configure : HRESULT
VAR_INPUT
    pArg : POINTER TO LREAL; // pointer to array (LREAL) of arguments
    nArgSize : UDINT;         // size of arguments buffer in bytes
END_VAR
```

入力バッファは、以下の定義（入力形状）のいずれかに紐づいています。変数パラメータは、ファンクションブロック入力stInitParsの一部です。周波数帯域とチャンネルごとに設定する2つのパラメータは、[fLowerFrequencyLimit, fUpperFrequencyLimit] です。

バージョン	入力バッファ（マルチアレイ 入カストリーム） 要素タイプ、次元数、次元サイズ
全てのチャンネルで同一設定	LREAL, 1, 2
周波数帯域別の設定 (nMaxBands >= 1)	LREAL, 2, nMaxBands x 2
チャンネル別の設定 (nMaxBands = 1)	LREAL, 2, nChannels x 2
チャンネルと周波数帯域別の設定	LREAL, 3, nChannels x nMaxBands x 2

```
METHOD Init : HRESULT
VAR_INPUT
    stInitPars : ST_CM_IntegratedRMS_InitPars; // init parameter
    nOwnID : UDINT; // ID for this FB instance
    aDestIDs : ARRAY[1..cCMA_MaxDest] OF UDINT; // IDs of destinations for output
```

```
nResultBuffers : UDINT := 4; // number of MultiArrays which should be initialized for results (0 for no initialization)
END_VAR
```

Init():

このメソッドは通常、状態監視アプリケーションでは必要ありません。これはファンクションブロック初期化の代替手段を提供するものです。Init() メソッドは、PLCの初期化フェーズ中にのみ呼び出すことができます。実行時には使用できません。メソッドFB_init または、属性'call_after_init'の使用を参照してください（TwinCAT PLCリファレンスを参照）。さらに、これはファンクションブロックのカプセル化を容易にします。

Init()メソッドを使用して初期化を行う場合、ファンクションブロックインスタンスの入力パラメータを宣言で代入することはできません。

- **戻り値**：エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。
- **stInitPars:ST_CM_IntegratedRMS_InitPars [▶ 287]**型の初期化パラメータを持つファンクションブロック固有の構造体。パラメータは、上記の入出力バッファの定義と紐づいていなければなりません。
- **nOwnID**:一意のIDでファンクションブロックインスタンスを識別します。これは常に0より大きくなければなりません。この目的のためには、列挙を定義するアプローチが確立されています。
- **aDestIDs**:転送先のIDを指定することで、結果の転送先を定義します。出力バッファの定義（上記を参照）は、選択された各転送先の入力バッファの定義と紐づいていなければなりません。
- **nResultBuffers**:このファンクションブロックは、指定された数のマルチアレイバッファにより転送ストリームを初期化します。デフォルト値は4です。

ResetData() :

このメソッドは、既に追加された全てのデータレコードを削除します。ファンクションブロックのメモリ特性を参照してください。ResetData() の後、Call() メソッドが再度呼び出された場合、有効な結果を計算するために内部メモリを補充する必要があります。

- **戻り値**：エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD ResetData : HRESULT
VAR_INPUT
END_VAR
```

PassInputs() :

API PLC リファレンス [▶ 79]で説明されているとおり、FB_CMA_Source インスタンスが呼び出され、信号データがターゲットブロックに転送される限り、解析チェーンの以降の全てのファンクションブロックを周期的に呼び出す必要があります。

アルゴリズムを一定時間実行しない方が有用な場合があります。例えば、いくつかのアルゴリズムは、事前に教示や設定をした後でのみ実行すべきです。ファンクションブロックは周期的に呼び出す必要がありますが、ファンクションブロックに到着したデータが[通信リング \[▶ 72\]](#)内で転送されれば十分です。これは、Call() メソッドの代わりにPassInputs() メソッドを使用して行われます。アルゴリズム自体はここでは呼び出されず、したがって結果も計算されず、出力バッファも生成されません。

- **戻り値**：エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD PassInputs : HRESULT
VAR_INPUT
END_VAR
```

GetChannelErrors() :

このメソッドは、複数のチャンネル（nChannels > 1）を処理する際に、チャンネル固有の戻り値のリストを照会できるようにします。呼び出しは、ファンクションブロックの戻り値がeCM_InfRTime_AmbiguousChannelResults またはeCM_ErrRTime_ErrornousChannelResults のいずれかに該当する場合に有効です。

- **戻り値**: エラーコード一覧の読み込み処理に関する情報です。この値は、クエリが成功した場合はTRUE に、そうでない場合はFALSE に設定されます。

```

METHOD GetChannelErrors : BOOL
VAR_IN_OUT
    aChannelErrors : ARRAY[*] OF HRESULT;
END_VAR

```

- aChannelErrors:HRESULT 型の長さnChannelsのエラーリストです。

類似したファンクションブロック

ファンクションブロックFB_CMA_MultiBandRMS [▶ 177] は、設定可能な周波数帯域におけるシングルおよびマルチチャンネルの実数値時系列データのRMS値を計算します。

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM, Tc3_CM_Base



CM 3.1 では一部の機能がすでに使用可能です。[互換性 \[▶ 61\]](#)の項を参照してください。

5.1.21 FB_CMA_MagnitudeSpectrum

入力信号の実数値のマグニチュードスペクトル（振幅スペクトルともいう）を計算します。

ファンクションブロックFB_CMA_MagnitudeSpectrum は、実数値入力信号からマグニチュードスペクトルを計算します。ファンクションブロックは複数の機能を実行します。[データストリームの解析 \[▶ 16\]](#) と [周波数解析 \[▶ 34\]](#) を参照してください：

入力データバッファは、まず直前のバッファとオーバーラップした後、ウィンドウ関数で乗算されます。パラメータnFFT_Length の値がnWindowLength より大きい場合、ウィンドウ付き時間信号は、必要なFFT入力長に達するよう最初と最後を同じ数のゼロで埋められます（ゼロパディング）。その後、実数値のFFTが適用され、結果として得られる複素数の絶対値が計算されます。パラメータbTransformToDecibel がTRUE の場合、値はデシベル値に変換されます。これらのデシベル値は、マグニチュードスペクトルとパワースペクトルにおいて同じです。つまり、マグニチュードスペクトルでは、デシベル値の計算において2乗の影響が考慮されており、その影響は2倍の係数として反映されています。さらに、マグニチュードスペクトルは、パラメータeScalingType を使ってスケーリングすることができます（[スペクトルのスケーリング \[▶ 25\]](#)を参照）。

FB_CMA_MagnitudeSpectrum ファンクションブロックは、[FB_CMA_PowerSpectrum \[▶ 189\]](#) と同様の動作をします。その差は、[FB_CMA_PowerSpectrum \[▶ 189\]](#) の結果の2乗です。

多くの場合、短期的なスペクトルは信号スペクトルを統計的に推定するのに適していません。多くの場合、複数の周波数または連続したスペクトルで平均化することにより、推定値の変動を抑えることが望ましいです。

スケーリング

加速度スペクトル密度などの結果値のスケーリングは、デフォルトでFFTの定義と一致しています。これは、ウィンドウ長とウィンドウ関数の影響が排除されることを意味します。[スペクトルスケーリングオプション \[▶ 371\]](#) の項で説明されている表形式のパラメータは、絶対計測の算術スケーリングに使用できます。

メモリ特性

ウェルチ法を使用しているため、現在の入力データバッファと最後に転送されたバッファの両方が計算に使用されます。組み込まれるバッファの数は、選択したオーバーラップ(nOverlap)に依存します。周波数解析では、時系列データの段階的な変化が考慮されます。したがって、正しい結果を得るためには、使用する入力データバッファは、間隔やジャンプなしに連結していなければなりません。

NaN（非数）の発生

入力ベクトルに1つ以上のNaN（非数）値が含まれる場合、出力セクタ全体がNaNで満たされます。詳細については別項NaN値 [▶ 71]を参照してください。

● NaN値の取り扱い

I NaN値につながる上記の状況を除外できない、または安全に無視できない場合、アプリケーションプログラムはこれらのエラー値を適切に処理できるようにしてください。

マルチチャンネル入力データ処理時の動作

複数のチャンネル（ $nChannels > 1$ ）を処理する場合、各チャンネルが異なる戻り値になる可能性があります。この場合、戻り値はファンクションブロックで個別に問い合わせできます。全てのチャンネルではなく、1つ以上のチャンネルの結果が許容されない場合、ファンクションブロックの値はeCM_InfRTime_AmbiguousChannelResultsに対応します。全てのチャンネルの結果が許容されない場合、ファンクションブロック上の値はeCM_ErrRTime_ErrornousChannelResultsに対応します。

全チャンネルの戻り値のリストは、GetChannelErrors()のメソッドで照会できます。

サンプルの実装

実装用のサンプルは以下のリンクから入手できます：[マグニチュードスペクトル： \[▶ 311\]](#)。

入力と出力

入力バッファと出力バッファは、以下の定義（入力/出力形状）のいずれかに該当します。変数パラメータは、ファンクションブロック入力stInitParsの一部です。

バージョン	入力バッファ（マルチアレイ入力ストリーム） 要素タイプ、次元数、次元サイズ	出力バッファ（マルチアレイ出力ストリーム） 要素タイプ、次元数、次元サイズ
標準バージョン ($nChannels = 1$)	LREAL, 1, nWindowLength-nOverlap	LREAL, 1, nFFT_Length/2+1
マルチチャンネルバージョン ($nChannels > 1$)	LREAL, 2, $nChannels \times nWindowLength - nOverlap$	LREAL, 2, $nChannels \times nFFT\_Length/2+1$

```

VAR_INPUT
    stInitPars      : ST_CM_MagnitudeSpectrum_InitPars; // init parameter
    nOwnID          : UDINT; // ID for this FB instance
    aDestIDs        : ARRAY[1..cCMA_MaxDest] OF UDINT; // IDs of destinations for output
    nResultBuffers  : UDINT := 4; // number of MultiArrays which should be initialized for results (0 for no initialization)
    tTransferTimeout : LTIME := LTIME#500US; // timeout checking off during access to inter-task FIFOs
END_VAR

```

入力パラメータ

このファンクションブロックの入力パラメータは初期化パラメータを表し、FBインスタンスの宣言で割り当て済みである必要があります。（または、Init()メソッド）一度しか割り当てできません。実行中の変更はできません。

- **stInitPars:** ST_CM_MagnitudeSpectrum_InitPars [▶ 288]型の初期化パラメータを持つファンクションブロック固有の構造体。パラメータは、上記の入出力バッファの定義と紐づいていなければなりません。
- **nOwnID:**一意のIDでファンクションブロックインスタンスを識別します。これは常に0より大きくなければなりません。この目的のためには、列挙を定義するアプローチが確立されています。
- **aDestIDs:**転送先のIDを指定することで、結果の転送先を定義します。出力バッファの定義（上記を参照）は、選択された各転送先の入力バッファの定義と紐づいていなければなりません。
- **nResultBuffers:**このファンクションブロックは、指定された数のマルチアレイバッファにより転送トレーストリームを初期化します。デフォルト値は4です。
- **tTransferTimeout:**内部マルチアレイ転送の同期タイムアウトの設定です。[並列処理 \[▶ 72\]](#)の項を参照してください。

出力パラメータ

```
VAR_OUTPUT
    bError          : BOOL;                // TRUE if an error occurs. Reset by next method call.
    hrErrorCode      : HRESULT;            // '< 0' = error; '> 0' = info; '0' = no error/info
    ipErrorMessage  : I_TcMessage := fbErrorMessage; // Shows detailed information about occurred errors, warnings and more.
    nCntResults      : ULINT;              // Counts outgoing results (MultiArrays were calculated and sent to transfer tray).
END_VAR
```

- **bError:**エラーが発生した場合の出力は、TRUEです。
- **hrErrorCode:**エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。
- **ipErrorMessage:**現在の戻り値に関するより詳細な情報を含みます。[エラーの説明と情報 \[▶ 81\]](#)の項を参照してください。この特別なインターフェースポインタは、常に有効かつ割り当て済みであるよう、内部的に保護されています。

メソッド

```
METHOD Init : HRESULT
VAR_INPUT
    stInitPars      : ST_CM MagnitudeSpectrum_InitPars; // init parameter
    nOwnID           : UDINT;                            // ID for this FB instance
    aDestIDs         : ARRAY[1..cCMA_MaxDest] OF UDINT;  // IDs of destinations for output
    nResultBuffers   : UDINT := 4;                       // number of MultiArrays which should be initialized for results (0 for no initialization)
END_VAR
```

Call():

このメソッドは、現在の入力データにアルゴリズムを適用するために、サイクルごとに呼び出されます。メソッドが新しい結果もエラーも示さない場合、ファンクションブロックは入力データを待機します。これは、解析チェーンのプロセスにおける規則的な動作です。

- **戻り値:**エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD Call : HRESULT
VAR_OUTPUT
    bNewResult      : BOOL;                // TRUE every time when outgoing MultiArray was calculated and sent to transfer tray.
    bError           : BOOL;                // TRUE if an error occurs.
    hrErrorCode      : HRESULT;            // '< 0' = error; '> 0' = info; '0' = no error/info
END_VAR
```

- **bError:**エラーが発生した場合の出力は、TRUEです。
- **hrErrorCode:**エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。この出力はメソッドの戻り値と同じです。



タイムアウトが発生した場合、または結果を格納するマルチアレイバッファがない場合、入力データも結果データも失われません。次の呼び出しに転送されます。

Init():

このメソッドは通常、状態監視アプリケーションでは必要ありません。これはファンクションブロック初期化の代替手段を提供するものです。Init() メソッドは、PLCの初期化フェーズ中にのみ呼び出すことができます。実行時には使用できません。メソッドFB_init または、属性'call_after_init' の使用を参照してください (TwinCAT PLCリファレンスを参照)。さらに、これはファンクションブロックのカプセル化を容易にします。

Init()メソッドを使用して初期化を行う場合、ファンクションブロックインスタンスの入力パラメータを宣言で代入することはできません。

- **戻り値:**エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

- **stInitPars:** ST_CM_MagnitudeSpectrum_InitPars [▶ 288]型の初期化パラメータを持つファンクションブロック固有の構造体。パラメータは、上記の入出力バッファの定義と紐づいていなければなりません。
- **nOwnID:**一意のIDでファンクションブロックインスタンスを識別します。これは常に0より大きくなければなりません。この目的のためには、列挙を定義するアプローチが確立されています。
- **aDestIDs:**転送先のIDを指定することで、結果の転送先を定義します。出力バッファの定義（上記を参照）は、選択された各転送先の入力バッファの定義と紐づいていなければなりません。
- **nResultBuffers:**このファンクションブロックは、指定された数のマルチアレイバッファにより転送トレイストリームを初期化します。デフォルト値は4です。

ResetData() :

このメソッドは、既に追加された全てのデータレコードを削除します。ファンクションブロックのメモリ特性を参照してください。ResetData() の後、Call() メソッドが再度呼び出された場合、有効な結果を計算するために内部メモリを補充する必要があります。

- **戻り値:** エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、エラーコード一覧 [▶ 367]に記載されています。

```
METHOD ResetData : HRESULT
VAR_INPUT
END_VAR
```

PassInputs() :

API PLC リファレンス [▶ 79]で説明されているとおり、FB_CMA_Source インスタンスが呼び出され、信号データがターゲットブロックに転送される限り、解析チェーンの以降の全てのファンクションブロックを周期的に呼び出す必要があります。

アルゴリズムを一定時間実行しない方が有用な場合があります。例えば、いくつかのアルゴリズムは、事前に教示や設定をした後でのみ実行すべきです。ファンクションブロックは周期的に呼び出す必要がありますが、ファンクションブロックに到着したデータが通信リング [▶ 72]内で転送されれば十分です。これは、Call() メソッドの代わりにPassInputs() メソッドを使用して行われます。アルゴリズム自体はここでは呼び出されず、したがって結果も計算されず、出力バッファも生成されません。

- **戻り値:** エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、エラーコード一覧 [▶ 367]に記載されています。

```
METHOD PassInputs : HRESULT
VAR_INPUT
END_VAR
```

GetChannelErrors() :

このメソッドは、複数のチャンネル（nChannels > 1）を処理する際に、チャンネル固有の戻り値のリストを照会できるようにします。呼び出しは、ファンクションブロックの戻り値がeCM_InfrTime_AmbiguousChannelResults またはeCM_ErrRTTime_ErrornousChannelResults のいずれかに該当する場合に有用です。

- **戻り値:** エラーコード一覧の読み込み処理に関する情報です。この値は、クエリが成功した場合はTRUE に、そうでない場合はFALSE に設定されます。

```
METHOD GetChannelErrors : BOOL
VAR_IN_OUT
    aChannelErrors : ARRAY[*] OF HRESULT;
END_VAR
```

- **aChannelErrors:** HRESULT 型の長さnChannelsのエラーリストです。

類似したファンクションブロック

FB_CM_PowerSpectrum [▶ 189]ブロックは、最後のステップの値を2乗してパワースペクトルを計算します。

FB_CMA_PowerCepstrum [▶ 185]ファンクションブロックは、高調波を強調する変換を行います。

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM, Tc3_CM_Base



CM 3.1 では一部の機能がすでに使用可能です。[互換性 \[▶ 61\]](#)の項を参照してください。

5.1.22 FB_CMA_MeanStressCorrection

分類された反転点から無応力反転点の平均値を計算します。

ファンクションブロックFB_CMA_MeanStressCorrectionは、レンジ・ミーンカウントを平均応力の影響を除いたレンジ・ミーンカウントへ変換するために、グッドマン法およびガーバー法に基づいた平均応力補正を実装します。カウントは多次元マルチアレイの形式で処理されます。

コンポーネントの疲労損傷は、ファンクションブロックFB_CMA_RainflowCounting [▶ 198] でカウントされたハーフループを基に、ファンクションブロックFB_CMA_MinersRule [▶ 168]で計算できます。損傷監視の場合、計数結果の平均応力補正は、ファンクションブロックFB_CMA_MeanStressCorrection [▶ 165] を使用して実行できます。

マルチチャンネル入力データ処理時の動作

複数のチャンネル ($nChannels > 1$) を処理する場合、各チャンネルが異なる戻り値になる可能性があります。この場合、戻り値はファンクションブロックで個別に問い合わせできます。全てのチャンネルではなく、1つ以上のチャンネルの結果が許容されない場合、ファンクションブロックの値はeCM_InfrTime_AmbiguousChannelResultsに対応します。全てのチャンネルの結果が許容されない場合、ファンクションブロック上の値はeCM_ErrRTime_ErronousChannelResults に対応します。

全チャンネルの戻り値のリストは、GetChannelErrors()のメソッドで照会できます。

サンプルの実装

実装用のサンプルは以下のリンクから入手できます：[疲労分析 \[▶ 351\]](#)

入力と出力

入力バッファと出力バッファは、以下の定義（入力/出力形状）のいずれかに該当します。変数パラメータは、ファンクションブロック入力stInitPars の一部です。

バージョン	入力バッファ（マルチアレイ 入力ストリーム） 要素タイプ、次元数、次元サイズ	出力バッファ（マルチアレイ 出力ストリーム） 要素タイプ、次元数、次元サイズ
標準バージョン ($nChannels = 1$)	UDINT, 2, ($nBins+2$) x ($nBinsMean+2$)	UDINT, 2, ($nBins+2$) x ($nBinsMean+2$)
マルチチャンネルバージョン ($nChannels > 1$)	UDINT, 3, $nChannels$ x ($nBins+2$) x ($nBinsMean+2$)	UDINT, 3, $nChannels$ x ($nBins+2$) x ($nBinsMean+2$)

*:この次元の長さは任意に選択できるため、アプリケーションや先行のアルゴリズムの出力バッファに合わせる必要があります。

```

VAR_INPUT
    stInitPars      : ST_CM_MeanStressCorrection_InitPars; // init parameter
    nOwnID          : UDINT;                               // ID for this FB instance
    aDestIDs        : ARRAY[1..cCMA_MaxDest] OF UDINT;     // IDs of destinations for output
    nResultBuffers  : UDINT := 4;                           // number of MultiArrays which should be
// initialized for results (0 for no initialization)
    tTransferTimeout : LTIME := LTIME#500US;              // timeout checking off during access to
// inter-task FIFOs
END_VAR

```

入力パラメータ

このファンクションブロックの入力パラメータは初期化パラメータを表し、FBインスタンスの宣言で割り当て済みである必要があります。(または、Init()メソッド) 一度しか割り当てできません。実行中の変更はできません。

- **stInitPars:** ST_CM_MeanStressCorrection_InitPars [▶ 289] 型の初期化パラメータを持つファンクションブロック固有の構造体。パラメータは、上記の入出力バッファの定義と紐づいていなければなりません。
- **nOwnID:** 一意のIDでファンクションブロックインスタンスを識別します。これは常に0より大きくなければなりません。この目的のためには、列挙を定義するアプローチが確立されています。
- **aDestIDs:** 転送先のIDを指定することで、結果の転送先を定義します。出力バッファの定義（上記を参照）は、選択された各転送先の入力バッファの定義と紐づいていなければなりません。
- **nResultBuffers:** このファンクションブロックは、指定された数のマルチアレイバッファにより転送トレイストリームを初期化します。デフォルト値は4です。
- **tTransferTimeout:** 内部マルチアレイ転送の同期タイムアウトの設定です。[並列処理 \[▶ 72\]](#)の項を参照してください。

出力パラメータ

```
VAR_OUTPUT
    bError      : BOOL;                // TRUE if an error occurs. Reset by next method call.
    hrErrorCode  : HRESULT;            // '< 0' = error; '> 0' = info; '0' = no error/info
    ipErrorMessage : I_TcMessage := fbErrorMessage; // Shows detailed information about occurred errors, warnings and more.
    nCntResults   : ULINT;             // Counts outgoing results (MultiArrays were calculated and sent to transfer tray).
END_VAR
```

- **bError:** エラーが発生した場合の出力は、TRUEです。
- **hrErrorCode:** エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。
- **ipErrorMessage:** 現在の戻り値に関するより詳細な情報を含みます。[エラーの説明と情報 \[▶ 81\]](#)の項を参照してください。この特別なインターフェースポインタは、常に有効かつ割り当て済みであるよう、内部的に保護されています。

メソッド

Call():

このメソッドは、現在の入力データにアルゴリズムを適用するために、サイクルごとに呼び出されます。メソッドが新しい結果もエラーも示さない場合、ファンクションブロックは入力データを待機します。これは、解析チェーンのプロセスにおける規則的な動作です。

- **戻り値:** エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD Call : HRESULT
VAR_OUTPUT
    bNewResult   : BOOL;                // TRUE every time when outgoing MultiArray was calculated and sent to transfer tray.
    bError       : BOOL;                // TRUE if an error occurs.
    hrErrorCode   : HRESULT;            // '< 0' = error; '> 0' = info; '0' = no error/info
END_VAR
```

- **bError:** エラーが発生した場合の出力は、TRUEです。
- **hrErrorCode:** エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。この出力はメソッドの戻り値と同じです。



タイムアウトが発生した場合、または結果を格納するマルチアレイバッファがない場合、入力データも結果データも失われません。次の呼び出しに転送されます。

Configure() :

平均応力補正は、このメソッドを呼び出すことで設定できます。E_CM_MeanStressCorrection [▶ 273] 型の値で設定できます。詳しくは疲労寿命解析と損傷計算 [▶ 49] を参照してください。

- **戻り値**：エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、エラーコード一覧に記載されています。

METHOD Configure : HRESULT

VAR_INPUT

pArg : POINTER TO UDINT; // pointer to 1-dimensional array (UDINT) of arguments
nArgSize : UDINT; // size of arguments buffer in bytes

END_VAR

入力バッファは、以下の定義（入力形状）のいずれかに紐づいています。変数パラメータは、ファンクションブロック入力stInitPars の一部です。

バージョン	入力バッファ（マルチアレイ 入カストリーム） 要素タイプ、次元数、次元サイズ
全てのチャンネルおよびサブチャンネルで同一設定	UDINT, 1, 1
チャンネル固有の設定	UDINT, 2, nChannels x 1

METHOD Init : HRESULT

VAR_INPUT

stInitPars : ST_CM_MeanStressCorrection_InitPars; // init parameter
nOwnID : UDINT; // ID for this FB instance
aDestIDs : ARRAY[1..cCMA_MaxDest] OF UDINT; // IDs of destinations for output
nResultBuffers : UDINT := 4; // number of MultiArrays which should be
initialized for results (0 for no initialization)

END_VAR

Init():

このメソッドは通常、状態監視アプリケーションでは必要ありません。これはファンクションブロック初期化の代替手段を提供するものです。Init() メソッドは、PLCの初期化フェーズ中にのみ呼び出すことができます。実行時には使用できません。メソッドFB_init または、属性'call_after_init' の使用を参照してください（TwinCAT PLCリファレンスを参照）。さらに、これはファンクションブロックのカプセル化を容易にします。

Init()メソッドを使用して初期化を行う場合、ファンクションブロックインスタンスの入力パラメータを宣言で代入することはできません。

- **戻り値**：エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、エラーコード一覧 [▶ 367] に記載されています。
- stInitPars: ST_CM_MeanStressCorrection_InitPars [▶ 289] 型の初期化パラメータを持つファンクションブロック固有の構造体。パラメータは、上記の入出力バッファの定義と紐づいていなければなりません。
- nOwnID:一意のIDでファンクションブロックインスタンスを識別します。これは常に0より大きくなければなりません。この目的のためには、列挙を定義するアプローチが確立されています。
- aDestIDs:転送先のIDを指定することで、結果の転送先を定義します。出力バッファの定義（上記を参照）は、選択された各転送先の入力バッファの定義と紐づいていなければなりません。
- nResultBuffers:このファンクションブロックは、指定された数のマルチアレイバッファにより転送トレイストリームを初期化します。デフォルト値は4です。

PassInputs() :

API PLC リファレンス [▶ 79]で説明されているとおり、FB_CMA_Source インスタンスが呼び出され、信号データがターゲットブロックに転送される限り、解析チェーンの以降の全てのファンクションブロックを周期的に呼び出す必要があります。

アルゴリズムを一定時間実行しない方が有用な場合があります。例えば、いくつかのアルゴリズムは、事前に教示や設定をした後でのみ実行すべきです。ファンクションブロックは周期的に呼び出す必要がありますが、ファンクションブロックに到着したデータが通信リング [▶ 72]内で転送されれば十分です。これは、Call() メソッドの代わりにPassInputs() メソッドを使用して行われます。アルゴリズム自体はここでは呼び出されず、したがって結果も計算されず、出力バッファも生成されません。

- **戻り値**：エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD PassInputs : HRESULT
VAR_INPUT
END_VAR
```

GetChannelErrors() :

このメソッドは、複数のチャンネル（nChannels > 1）を処理する際に、チャンネル固有の戻り値のリストを照会できるようにします。呼び出しは、ファンクションブロックの戻り値が eCM_InfRTTime_AmbiguousChannelResults または eCM_ErrRTTime_ErrornousChannelResults のいずれかに該当する場合に有効です。

- **戻り値**：エラーコード一覧の読み込み処理に関する情報です。この値は、クエリが成功した場合は TRUE に、そうでない場合は FALSE に設定されます。

```
METHOD GetChannelErrors : BOOL
VAR_IN_OUT
aChannelErrors : ARRAY[*] OF HRESULT;
END_VAR
```

- aChannelErrors:HRESULT 型の長さnChannelsのエラーリストです。

類似したファンクションブロック

ファンクションブロックFB_CMA_RainflowCounting [[▶ 198](#)]は、3点ルールに基づいて計算された荷重と平均値のカウント行列を使用します。

ファンクションブロックFB_CMA_MinersRule [[▶ 168](#)] のアルゴリズムは、カウントされたハーフサイクルに基づいてコンポーネントの疲労損傷を計算するために使用できます。

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM, Tc3_CM_Base

5.1.23 FB_CMA_MinersRule

部品の疲労損傷を計算する。

ファンクションブロックFB_CMA_MinersRuleは、荷重と平均値、およびカウントされたハーフループの数に基づいてコンポーネントの疲労損傷を計算します。

その結果は、計算された総損傷（疲労計算の背景の [\[▶ 49\]](#)値Dを参照）と、基礎となるハーフループの数、すなわち[fDamage],[fCycles] を含む1次元配列aResult : ARRAY[0..1] OF LREAL, です。

コンポーネントの疲労損傷は、ファンクションブロックFB_CMA_MinersRuleで [\[▶ 168\]](#)、ファンクションブロックFB_CMA_RainflowCountingで [\[▶ 198\]](#)カウントされたハーフループに基づいて計算できます。損傷モニタリングの場合、ファンクションブロックFB_CMA_MeanStressCorrectionを [\[▶ 165\]](#)使用して、計数結果の平均応力補正を行うことができます。

マルチチャンネル入力データ処理時の動作

複数のチャンネル（nChannels > 1）を処理する場合、各チャンネルが異なる戻り値になる可能性があります。この場合、戻り値はファンクションブロックで個別に問い合わせできます。全てのチャンネルではなく、1つ以上のチャンネルの結果が許容されない場合、ファンクションブロックの値は eCM_InfRTTime_AmbiguousChannelResultsに対応します。全てのチャンネルの結果が許容されない場合、ファンクションブロック上の値はeCM_ErrRTTime_ErrornousChannelResults に対応します。

全チャンネルの戻り値のリストは、GetChannelErrors()のメソッドで照会できます。

サンプルの実装

サンプルは以下のリンクから入手できます：[疲労分析 \[▶ 351\]](#)

入力と出力

入力バッファと出力バッファは、以下の定義（入力/出力形状）のいずれかに該当します。変数パラメータは、ファンクションブロック入力stInitParsの一部です。

バージョン	入力バッファ（マルチアレイ 入力ストリーム） 要素タイプ、次元数、次元サイズ	出力バッファ（マルチアレイ 出力ストリーム） 要素タイプ、次元数、次元サイズ
標準バージョン (nChannels = 1)	UDINT, 2, (nBins+2) x (nBinsMean+2)	LREAL, 1, 2
マルチチャンネルバージョン (nChannels > 1)	LREAL, 2, nChannels x (nBins+2) x (nBinsMean+2)	LREAL, 2, nチャンネル×2

```
VAR_INPUT
    stInitPars      : ST_CM_MinersRule_InitPars;           // init parameter
    nOwnID          : UDINT;                               // ID for this FB instance
    aDestIDs        : ARRAY[1..cCMA_MaxDest] OF UDINT;     // IDs of destinations for output
    nResultBuffers  : UDINT := 4;                          // number of MultiArrays which should be in
    initialized for results (0 for no initialization)
    tTransferTimeout : LTIME := LTIME#500US;              // timeout checking off during access to in
    ter-task FIFOs
END_VAR
```

入力パラメータ

このファンクションブロックの入力パラメータは初期化パラメータを表し、FBインスタンスの宣言で割り当て済みである必要があります。（または、Init()メソッド）一度しか割り当てできません。実行中の変更はできません。

- **stInitPars:** [ST_CM_MinersRule_InitPars \[▶ 290\]](#) 型の初期化パラメータを持つファンクションブロック固有の構造体。パラメータは、上記の入出力バッファの定義と紐づいていなければなりません。
- **nOwnID:**一意のIDでファンクションブロックインスタンスを識別します。これは常に0より大きくなければなりません。この目的のためには、列挙を定義するアプローチが確立されています。
- **aDestIDs:**転送先のIDを指定することで、結果の転送先を定義します。出力バッファの定義（上記を参照）は、選択された各転送先の入力バッファの定義と紐づいていなければなりません。
- **nResultBuffers:**このファンクションブロックは、指定された数のマルチアレイバッファにより転送トレイストリームを初期化します。デフォルト値は4です。
- **tTransferTimeout:**内部マルチアレイ転送の同期タイムアウトの設定です。[並列処理 \[▶ 72\]](#)の項を参照してください。

出力パラメータ

```
VAR_OUTPUT
    bError          : BOOL;                               // TRUE if an error occurs. Reset by next metho
    d call.
    hrErrorCode      : HRESULT;                           // '< 0' = error; '> 0' = info; '0' = no error/
    info
    ipErrorMessage  : I_TcMessage := fbErrorMessage;     // Shows detailed information about occurred er
    rors, warnings and more.
    nCntResults      : ULINT;                             // Counts outgoing results (MultiArrays were ca
    lculated and sent to transfer tray).
END_VAR
```

- **bError:**エラーが発生した場合の出力は、TRUEです。
- **hrErrorCode:**エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。
- **ipErrorMessage:**現在の戻り値に関するより詳細な情報を含みます。[エラーの説明と情報 \[▶ 81\]](#)の項を参照してください。この特別なインターフェースポインタは、常に有効かつ割り当て済みであるよう、内部的に保護されています。

メソッド

Call():

このメソッドは、現在の入力データにアルゴリズムを適用するために、サイクルごとに呼び出されます。メソッドが新しい結果もエラーも示さない場合、ファンクションブロックは入力データを待機します。これは、解析チェーンのプロセスにおける規則的な動作です。

- **戻り値**：エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD Call : HRESULT
VAR_OUTPUT
    bNewResult      : BOOL;          // TRUE every time when outgoing MultiArray was calculated and sent to transfer tray.
    bError          : BOOL;          // TRUE if an error occurs.
    hrErrorCode      : HRESULT;      // '< 0' = error; '> 0' = info; '0' = no error/info
END_VAR
```

- **bError**:エラーが発生した場合の出力は、TRUEです。
- **hrErrorCode**:エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。この出力はメソッドの戻り値と同じです。



タイムアウトが発生した場合、または結果を格納するマルチアレイバッファがない場合、入力データも結果データも失われません。次の呼び出しに転送されます。

Configure() :

このメソッドを呼び出すことで、エラーが特定されるまでのハーフループの回数を、定義されたビンごとに設定することができます。

- **戻り値**：エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧](#)に記載されています。

```
METHOD Configure : HRESULT
VAR_INPUT
    pArg      : POINTER TO LREAL; // pointer to 1-dimensional array (LREAL) of arguments
    nArgSize  : UDINT;            // size of arguments buffer in bytes
END_VAR
```

入力バッファは、以下の定義（入力形状）のいずれかに紐づいています。変数パラメータは、ファンクションブロック入力stInitPars の一部です。

バージョン	入力バッファ（マルチアレイ入カストリーム） 要素タイプ、次元数、次元サイズ
全てのチャンネルおよびサブチャンネルで同一設定	LREAL, 1, nBins+2
チャンネル固有の設定	LREAL, 2, nChannels x (nBins+2)

```
METHOD Init : HRESULT
VAR_INPUT
    stInitPars      : ST_CM_RainflowCounting_InitPars; // init parameter
    nOwnID          : UDINT;                          // ID for this FB instance
    aDestIDs        : ARRAY[1..cCMA_MaxDest] OF UDINT; // IDs of destinations for output
    nResultBuffers  : UDINT := 4;                     // number of MultiArrays which should be initialized for results (0 for no initialization)
END_VAR
```

Init():

このメソッドは通常、状態監視アプリケーションでは必要ありません。これはファンクションブロック初期化の代替手段を提供するものです。Init() メソッドは、PLCの初期化フェーズ中にのみ呼び出すことができます。実行時には使用できません。メソッドFB_init または、属性'call_after_init' の使用を参照してください（TwinCAT PLCリファレンスを参照）。さらに、これはファンクションブロックのカプセル化を容易にします。

Init()メソッドを使用して初期化を行う場合、ファンクションブロックインスタンスの入力パラメータを宣言で代入することはできません。

- **戻り値**：エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

- **stInitPars:** `ST_CM_MinersRule_InitPars` [▶ 290] 型の初期化パラメータを持つファンクションブロック固有の構造体。パラメータは、上記の入出力バッファの定義と紐づいていなければなりません。
- **nOwnID:** 一意のIDでファンクションブロックインスタンスを識別します。これは常に0より大きくなければなりません。この目的のためには、列挙を定義するアプローチが確立されています。
- **aDestIDs:** 転送先のIDを指定することで、結果の転送先を定義します。出力バッファの定義（上記を参照）は、選択された各転送先の入力バッファの定義と紐づいていなければなりません。
- **nResultBuffers:** このファンクションブロックは、指定された数のマルチアレイバッファにより転送ストリームを初期化します。デフォルト値は4です。

ResetData() :

このメソッドは、既に追加された全てのデータレコードを削除します。ファンクションブロックのメモリ特性を参照してください。ResetData() の後、Call() メソッドが再度呼び出された場合、有効な結果を計算するために内部メモリを補充する必要があります。

- **戻り値:** エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧](#) [▶ 367] に記載されています。

```
METHOD ResetData : HRESULT
VAR_INPUT
END_VAR
```

PassInputs() :

API PLC リファレンス [▶ 79] で説明されているとおり、FB_CMA_Source インスタンスが呼び出され、信号データがターゲットブロックに転送される限り、解析チェーンの以降の全てのファンクションブロックを周期的に呼び出す必要があります。

アルゴリズムを一定時間実行しない方が有用な場合があります。例えば、いくつかのアルゴリズムは、事前に教示や設定をした後でのみ実行すべきです。ファンクションブロックは周期的に呼び出す必要がありますが、ファンクションブロックに到着したデータが通信リング [▶ 72] 内で転送されれば十分です。これは、Call() メソッドの代わりに PassInputs() メソッドを使用して行われます。アルゴリズム自体はここでは呼び出されず、したがって結果も計算されず、出力バッファも生成されません。

- **戻り値:** エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧](#) [▶ 367] に記載されています。

```
METHOD PassInputs : HRESULT
VAR_INPUT
END_VAR
```

GetChannelErrors() :

このメソッドは、複数のチャンネル（nChannels > 1）を処理する際に、チャンネル固有の戻り値のリストを照会できるようにします。呼び出しは、ファンクションブロックの戻り値が eCM_InfRTTime_AmbiguousChannelResults または eCM_ErrRTTime_ErrornousChannelResults のいずれかに該当する場合に有用です。

- **戻り値:** エラーコード一覧の読み込み処理に関する情報です。この値は、クエリが成功した場合は TRUE に、そうでない場合は FALSE に設定されます。

```
METHOD GetChannelErrors : BOOL
VAR_IN_OUT
aChannelErrors : ARRAY[*] OF HRESULT;
END_VAR
```

- **aChannelErrors:** HRESULT 型の長さ nChannels のエラーリストです。

類似したファンクションブロック

ファンクションブロック `FB_CMA_RainflowCounting` [▶ 198] は、3点ルールに基づいて計算された荷重と平均値のカウント行列を使用します。

ファンクションブロック `FB_CMA_MeanStressCorrection` [▶ 165] により、監視対象部品の引張強度に関する平均応力補正を実行できます。

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM, Tc3_CM_Base

5.1.24 FB_CMA_MomentCoefficients

シングルチャンネルおよびマルチチャンネルの実数値時系列データの平均値、経験的標準偏差、歪度、超過値を計算します。

このファンクションブロックは、入力信号を時系列データとして扱い、必要に応じて複数の独立したチャンネルとともに処理します。各チャンネルのモーメント係数は、オプションで4次まで計算されます。計算するモーメントの最大次数は、パラメータnOrder で設定できます。モーメント係数の適用に特化した列挙型 E_CM_MCoefOrder [▶ 272]も用意されています。結果は、これらの係数と対応するインデックスの配列として転送されます。

デフォルトでは、経験的標準偏差、歪度、超過の計算にはベッセル補正は適用されません。初期化パラメータで、補正オプションをオンにできます (bPopulationEstimates を参照)。期待通りの結果を得るためには、パラメータをTRUE に設定する必要があります。ベッセル補正の影響は、サンプルサイズが大きくなるほど小さくなります。補正された経験的標準偏差と補正されていない経験的標準偏差との間の相対偏差は、明確に決定できます。以下の表がヒントになります：

サンプル数 N	相対偏差 / %
10	-5.13
100	-0.501
1000	-0,05001
10000	-0,0050001

ファンクションブロックの出力：ランダムサンプルの範囲 N (全てのnOrderに対して)、算術平均値 (nOrder = 1)、経験的標準偏差 (nOrder = 2)、歪度 (nOrder = 3)、過剰度 (nOrder = 4)。

経験的に計算されたモーメントの定義

算術平均値

$$\bar{x} = \frac{1}{N} \sum_{n=0}^{N-1} x[n]$$

経験的標準偏差 (ベッセル補正なし)

$$s' = \sqrt{\frac{1}{N} \sum_{n=0}^{N-1} (x[n] - \bar{x})^2}$$

経験的標準偏差 (ベッセル補正あり)

$$s = \sqrt{\frac{1}{N-1} \sum_{n=0}^{N-1} (x[n] - \bar{x})^2}$$

経験的歪度 (ベッセルの補正なし v' 、ベッセル補正あり v)

$$v' = \frac{1}{N} \sum_{n=0}^{N-1} \left(\frac{x[n] - \bar{x}}{s'} \right)^3 \quad v = \frac{1}{N} \sum_{n=0}^{N-1} \left(\frac{x[n] - \bar{x}}{s} \right)^3$$

経験的過剰度（ベッセル補正なし E' 、ベッセル補正あり E ）

$$K' = \frac{1}{N} \sum_{n=0}^{N-1} \left(\frac{x[n] - \bar{x}}{s'} \right)^4 \quad K = \frac{1}{N} \sum_{n=0}^{N-1} \left(\frac{x[n] - \bar{x}}{s} \right)^4$$

$$E' = K' - 3$$

$$E = K - 3$$

したがって過剰度 E は、尖度 K と値3との差であり、これは正規分布の尖度に相当します。算出された尖度の正規分布に基づく評価方法が記述されています。

各サイクルでチャンネルごとに1サンプル（最初の表の入出力を参照）を追加でき、1サイクルでチャンネルごとに複数のサンプルをサンプル数に追加することもできます（2番目の表の入出力を参照）。

追加のコメント

標準偏差と高次モーメントの計算には、最低限のサンプル値が必要です。ベッセル補正が無効の場合、平均値と標準偏差はサンプルサイズ1で計算されます。歪度と過剰度を計算するには、2つの値が必要です。ベッセル補正が有効の場合、必要な最小サンプルサイズは次の次数に対応します（平均値：1、標準偏差：2、歪度：3、過剰度：4）。さらに、歪度と過剰度を計算するためには、分散がゼロであってはなりません。

入力値が何桁で大きく異なる場合、高いモーメントの結果が不正確になる可能性があります。

メモリ特性

サンプル数 N は、新しいデータセットが入力されるたびに自動的に増加し、現在の結果を計算します。つまり、ファンクションブロックがインスタンス化されてからの全ての入力値を使用します。サンプル数のゼロリセット（FBの内部メモリの削除）は、ResetData() メソッド、またはCallEx() メソッドが使用される場合、変数bResetData によって提供されます。

NaN（非数）の発生

入力値の数が特定のチャンネルの結果を計算するのに十分でない場合、または分散がゼロの場合、そのチャンネルに対してIEEE 754に従ってNaN（非数）の値が返されます。このエラー値があるかどうかは、ファンクションLrealIsNaN()を使用して確認できます。原因として、これまでのところ十分な入力データが転送されていないか、個々のチャンネルの入力値としてNaN値しか転送されていないことが考えられます。例えば、断線やスイッチングエラーによってセンサデータが転送されなかった場合など、時系列データの値が一定であれば、分散がゼロになることがあります。

入力値セットに特別な定数 NaN が含まれる場合、この時間ステップでは、チャンネルの統計に値は追加されません。つまり、NaN は欠損値の指標として扱われます。

● NaN値の取り扱い

i NaN値につながる上記の状況を除外できない、または安全に無視できない場合、アプリケーションプログラムはこれらのエラー値を適切に処理できるようにしてください。

マルチチャンネル入力データ処理時の動作

複数のチャンネル（nChannels > 1）を処理する場合、各チャンネルが異なる戻り値になる可能性があります。この場合、戻り値はファンクションブロックで個別に問い合わせできます。全てのチャンネルではなく、1つ以上のチャンネルの結果が許容されない場合、ファンクションブロックの値はeCM_InfRTime_AmbiguousChannelResultsに対応します。全てのチャンネルの結果が許容されない場合、ファンクションブロック上の値はeCM_ErrRTime_ErrornousChannelResults に対応します。

全チャンネルの戻り値のリストは、GetChannelErrors()のメソッドで照会できます。

複数のサブチャンネル(nSubChannels > 0)を処理する場合、入出力データのフォーマットには特に注意が必要です。入力データが上流のファンクションブロックによるマルチチャンネルの結果で構成されている場合、nChannels の値を採用する必要があります。この場合、パラメータnSubChannels を使用して追加の設定を行います。

例：マルチチャンネルスペクトル（FB_CMA_MagnitudeSpectrumなど）の周波数チャンネルの統計的考察（FB_CMA_Quantilesなど）では、nChannels の値は、入力信号の数と同じである必要があります。サブチャンネルの数nSubChannels は、スペクトルの長さに対応します。

入力と出力

入力バッファと出力バッファは、以下の定義（形状）に該当します。変数パラメータは、ファンクションブロック入力stInitPars の一部です。

バージョン	入力バッファ（マルチアレイ入力ストリーム） 要素タイプ、次元数、次元サイズ	出力バッファ（マルチアレイ出力ストリーム） 要素タイプ、次元数、次元サイズ
標準バージョン (nSubChannels = 0)	LREAL, 1, nChannels	LREAL, 2, nChannels x nOrder+1
複数データセット用標準バージョン (nSubChannels = 0)	LREAL, 2, nChannelsx 指定なし*	LREAL, 2, nChannels x nOrder+1
マルチチャンネルバージョン (nSubChannels > 0)	LREAL, 2, nChannels x nSubChannels	LREAL, 3, nChannels x nSubChannels x nOrder+1
複数データセット用マルチチャンネルバージョン (nSubChannels > 0)	LREAL, 3, nChannels x nSubChannels x指定なし*	LREAL, 3, nChannels x nSubChannels x nOrder+1

*:この次元の長さは任意に選択できるため、アプリケーションや先行のアルゴリズムの出力バッファに合わせるができます。

```
VAR_INPUT
    stInitPars      : ST_CM_MomentCoefficients_InitPars; // init parameter
    nOwnID          : UDINT;                             // ID for this FB instance
    aDestIDs        : ARRAY[1..cCMA_MaxDest] OF UDINT;    // IDs of destinations for output
    nResultBuffers  : UDINT := 4;                         // number of MultiArrays which should be
    initialized for results (0 for no initialization)
    tTransferTimeout : LTIME := LTIME#500US;             // timeout checking off during access to
    inter-task FIFOs
END_VAR
```

入力パラメータ

このファンクションブロックの入力パラメータは初期化パラメータを表し、FBインスタンスの宣言で割り当て済みである必要があります。(または、Init()メソッド) 一度しか割り当てできません。実行中の変更はできません。

- **stInitPars:**ST_CM_MomentCoefficients_InitPars [▶ 290]型の初期化パラメータを持つファンクションブロック固有の構造体。パラメータは、上記の入出力バッファの定義と紐づいていなければなりません。
- **nOwnID:**一意のIDでファンクションブロックインスタンスを識別します。これは常に0より大きくなければなりません。この目的のためには、列挙を定義するアプローチが確立されています。
- **aDestIDs:**転送先のIDを指定することで、結果の転送先を定義します。出力バッファの定義（上記を参照）は、選択された各転送先の入力バッファの定義と紐づいていなければなりません。
- **nResultBuffers:**このファンクションブロックは、指定された数のマルチアレイバッファにより転送トレーストリームを初期化します。デフォルト値は4です。
- **tTransferTimeout:**内部マルチアレイ転送の同期タイムアウトの設定です。[並列処理 \[▶ 72\]](#)の項を参照してください。

出力パラメータ

```
VAR_OUTPUT
    bError          : BOOL;                             // TRUE if an error occurs. Reset by next method call.
    hrErrorCode      : HRESULT;                          // '< 0' = error; '> 0' = info; '0' = no error/info
    ipErrorMessage  : I_TcMessage := fbErrorMessage; // Shows detailed information about occurred errors, warnings and more.
```

```

    nCntResults      : ULINT;                                // Counts outgoing results (MultiArrays were ca
lculated and sent to transfer tray).
END_VAR

```

- **bError**:エラーが発生した場合の出力は、TRUEです。
- **hrErrorCode**:エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。
- **ipErrorMessage**:現在の戻り値に関するより詳細な情報を含みます。[エラーの説明と情報 \[▶ 81\]](#)の項を参照してください。この特別なインターフェースポインタは、常に有効かつ割り当て済みであるよう、内部的に保護されています。

メソッド

```

METHOD Init : HRESULT
VAR_INPUT
    stInitPars      : ST_CM_MomentCoefficients_InitPars;    // init parameter
    nOwnID          : UDINT;                                // ID for this FB instance
    aDestIDs        : ARRAY[1..cCMA_MaxDest] OF UDINT;       // IDs of destinations for output
    nResultBuffers  : UDINT := 4;                            // number of MultiArrays which should be ini
tialized for results (0 for no initialization)
END_VAR

```

Call():

このメソッドは、現在の入力データにアルゴリズムを適用するために、サイクルごとに呼び出されます。メソッドが新しい結果もエラーも示さない場合、ファンクションブロックは入力データを待機します。これは、解析チェーンのプロセスにおける規則的な動作です。

- **戻り値**:エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```

METHOD Call : HRESULT
VAR_OUTPUT
    bNewResult      : BOOL;                                // TRUE every time when outgoing MultiArray was calculated and sent t
o transfer tray.
    bError          : BOOL;                                // TRUE if an error occurs.
    hrErrorCode     : HRESULT;                             // '< 0' = error; '> 0' = info; '0' = no error/info
END_VAR

```

- **bError**:エラーが発生した場合の出力は、TRUEです。
- **hrErrorCode**:エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。この出力はメソッドの戻り値と同じです。



タイムアウトが発生した場合、または結果を格納するマルチアレイバッファがない場合、入力データも結果データも失われません。次の呼び出しに転送されます。

CallEx() :

このメソッドは、入力信号から内部メモリを更新するために、各サイクルで呼び出されます。結果は **nAppendData** サイクルごとに出力されます。別の方法としては、**Call()** があります。

メソッドが新しい結果もエラーも示さない場合、ファンクションブロックは入力データを待機します。これは、解析チェーンのプロセスにおける規則的な動作です。

- **戻り値**:エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```

METHOD CallEx : HRESULT
VAR_INPUT
    nAppendData     : UDINT;                                // count of data buffers which are appended until calculation (1= cal
culate always)
    bResetData      : BOOL;                                // automatic reset of dataset buffer after each calculation
END_VAR
VAR_OUTPUT
    bNewResult      : BOOL;                                // TRUE every time when outgoing MultiArray was calculated and sent
to transfer tray.
    bError          : BOOL;                                // TRUE if an error occurs.
    hrErrorCode     : HRESULT;                             // '< 0' = error; '> 0' = info; '0' = no error/info
END_VAR

```

- **nAppendData**: 計算を実行する前に、いくつかの入力データバッファを収集するかを定義します。これは、正確な結果を得るために複数のデータブロックを追加することが望ましいためです。
- **bResetData**: これを設定した場合、内部データバッファは計算後に完全に削除されます。
- **bError**: エラーが発生した場合の出力は、TRUEです。
- **hrErrorCode**: エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。この出力はメソッドの戻り値と同じです。



タイムアウトが発生した場合、または結果を格納するマルチアレイバッファがない場合、入力データも結果データも失われません。次の呼び出しに転送されます。

Init():

このメソッドは通常、状態監視アプリケーションでは必要ありません。これはファンクションブロック初期化の代替手段を提供するものです。Init() メソッドは、PLCの初期化フェーズ中にのみ呼び出すことができます。実行時には使用できません。メソッドFB_init または、属性'call_after_init'の使用を参照してください（TwinCAT PLCリファレンスを参照）。さらに、これはファンクションブロックのカプセル化を容易にします。

Init()メソッドを使用して初期化を行う場合、ファンクションブロックインスタンスの入力パラメータを宣言で代入することはできません。

- **戻り値**: エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。
- **stInitPars**: ST_CM_MomentCoefficients InitPars [▶ 290]型の初期化パラメータを持つファンクションブロック固有の構造体。パラメータは、上記の入出力バッファの定義と紐づいていなければなりません。
- **nOwnID**: 一意のIDでファンクションブロックインスタンスを識別します。これは常に0より大きくなければなりません。この目的のためには、列挙を定義するアプローチが確立されています。
- **aDestIDs**: 転送先のIDを指定することで、結果の転送先を定義します。出力バッファの定義（上記を参照）は、選択された各転送先の入力バッファの定義と紐づいていなければなりません。
- **nResultBuffers**: このファンクションブロックは、指定された数のマルチアレイバッファにより転送ストリームを初期化します。デフォルト値は4です。

ResetData() :

このメソッドは、既に追加された全てのデータレコードを削除します。ファンクションブロックのメモリ特性を参照してください。ResetData() の後、Call() メソッドが再度呼び出された場合、有効な結果を計算するために内部メモリを補充する必要があります。

- **戻り値**: エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD ResetData : HRESULT
VAR_INPUT
END_VAR
```

または、CallEx() の自動リセットを使用することもできます。

PassInputs() :

API PLC リファレンス [▶ 79]で説明されているとおり、FB_CMA_Source インスタンスが呼び出され、信号データがターゲットブロックに転送される限り、解析チェーンの以降の全てのファンクションブロックを周期的に呼び出す必要があります。

アルゴリズムを一定時間実行しない方が有用な場合があります。例えば、いくつかのアルゴリズムは、事前に教示や設定をした後でのみ実行すべきです。ファンクションブロックは周期的に呼び出す必要がありますが、ファンクションブロックに到着したデータが[通信リング \[▶ 72\]](#)内で転送されれば十分です。これは、Call() メソッドの代わりにPassInputs() メソッドを使用して行われます。アルゴリズム自体はここでは呼び出されず、したがって結果も計算されず、出力バッファも生成されません。

- **戻り値**: エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。


```
METHOD PassInputs : HRESULT
VAR_INPUT
END_VAR
```

GetChannelErrors() :

このメソッドは、複数のチャンネル（`nChannels > 1`）を処理する際に、チャンネル固有の戻り値のリストを照会できるようにします。呼び出しは、ファンクションブロックの戻り値が `eCM_InfRTIME_AmbiguousChannelResults` または `eCM_ErrRTIME_ErrornousChannelResults` のいずれかに該当する場合に有効です。

- **戻り値:** エラーコード一覧の読み込み処理に関する情報です。この値は、クエリが成功した場合は `TRUE` に、そうでない場合は `FALSE` に設定されます。

```
METHOD GetChannelErrors : BOOL
VAR_IN_OUT
aChannelErrors : ARRAY[*] OF HRESULT;
END_VAR
```

- `aChannelErrors`: `HRESULT` 型の長さ `nChannels` のエラーリストです。

類似したファンクションブロック

ファンクションブロック `FB_CMA_EmpiricalMean` [▶ 123] は、入力値の経験平均値を計算します。

ファンクションブロック `FB_CMA_EmpiricalStandardDeviation` [▶ 133] は、入力値の経験的標準偏差を計算します。

ファンクションブロック `FB_CMA_EmpiricalSkew` [▶ 128] は、入力値の経験的歪度を計算します。

ファンクションブロック `FB_CMA_EmpiricalExcess` [▶ 118] は、入力値の経験的な超過分を計算します。

`FB_CMA_HistArray` [▶ 145] ファンクションブロックは、入力値分布のヒストグラムを計算します。

`FB_CMA_Quantiles` [▶ 193] ファンクションブロックは、経験分布の分位数を計算し、外れ値の周波数を評価します。

尖度の代替として、`FB_CMA_CrestFactor` [▶ 93] ファンクションブロックは、信号のピーク性（クレスト係数）の別の尺度を計算しますが、これは異常値に対してより敏感です。

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM, Tc3_CM_Base



CM 3.1 では一部の機能がすでに使用可能です。[互換性](#) [▶ 61]の項を参照してください。

5.1.25 FB_CMA_MultiBandRMS

シングルチャンネルおよびマルチチャンネルの実数値時系列データに対して、設定可能な周波数帯域ごとのRMS値を計算します。

ファンクションブロック `FB_CMA_MultiBandRMS` は、個別に設定可能な周波数帯域に関連して、シングルおよびマルチチャンネル時系列の信号のRMS値を計算します。

チャンネル数は、入力ストリームを介して記述されます。チャンネルに設定された周波数帯域の最大数と内部フーリエ変換のパラメータは、`ST_CM_MultiBandRMS_InitPars` [▶ 291] を経由して転送されます。周波数帯域は、`Configure()` メソッドを呼び出すことで設定します。

このファンクションブロックは、例えばベアリングの損傷周波数のモニタリングに使用できます。

`FB_CMA_IntegratedRMS` [▶ 157] の定義範囲 :

ファンクションブロックFB_CMA_IntegratedRMSには追加機能があり、入力された時系列データを周波数帯域制限付きのRMS計算の前に時間積分できます（オプションで2次まで）。したがって、このファンクションブロックは、定義した周波数帯域の振動加速度、振動速度、振動変位などのRMS値を直接計算できます。

設定

値を持つ3次元配列が、設定パラメータとしてファンクションブロックのConfigure()メソッドに転送されます（入力チャンネルが1つの場合は2次元配列になることもあります）。そうでなければ全てのチャンネルが同一の設定になります。周波数帯域の下限と上限は配列で指定されます。次に、ファンクションブロックは、入力データに基づいて、各チャンネルの周波数帯域のRMS値を計算します。

メモリ特性

ウェルチ法を使用しているため、現在の入力データバッファと最後に転送されたバッファの両方が計算に使用されます。組み込まれるバッファの数は、選択したオーバーラップ(nOverlap)に依存します。周波数解析では、時系列データの段階的な変化が考慮されます。したがって、正しい結果を得るためには、使用する入力データバッファは、間隔やジャンプなしに連結していなければなりません。

NaN（非数）の発生

入力ベクトルに1つ以上のNaN（非数）値が含まれる場合、出力セクタ全体がNaNで満たされます。詳細については別項NaN値 [▶ 71]を参照してください。



NaN値の取り扱い

NaN値につながる上記の状況を除外できない、または安全に無視できない場合、アプリケーションプログラムはこれらのエラー値を適切に処理できるようにしてください。

マルチチャンネル入力データ処理時の動作

複数のチャンネル（nChannels > 1）を処理する場合、各チャンネルが異なる戻り値になる可能性があります。この場合、戻り値はファンクションブロックで個別に問い合わせできます。全てのチャンネルではなく、1つ以上のチャンネルの結果が許容されない場合、ファンクションブロックの値はeCM_InfRTime_AmbiguousChannelResultsに対応します。全てのチャンネルの結果が許容されない場合、ファンクションブロック上の値はeCM_ErrRTime_ErrornousChannelResultsに対応します。

全チャンネルの戻り値のリストは、GetChannelErrors()のメソッドで照会できます。

サンプルの実装

実装用のサンプルは以下のリンクから入手できます：https://infosys.beckhoff.com/content/1033/TF3600_TC3_ConditionMonitoring/Resources/3394505867.zip。

入力と出力

入力バッファと出力バッファは、以下の定義（入力/出力形状）のいずれかに該当します。変数パラメータは、ファンクションブロック入力stInitParsの一部です。任意のチャンネル数（nChannels >= 1）を処理する場合のルールは以下の通りです：

バージョン	入力バッファ（マルチアレイ入力ストリーム） 要素タイプ、次元数、次元サイズ	出力バッファ（マルチアレイ出力ストリーム） 要素タイプ、次元数、次元サイズ
標準バージョン (nMaxBands >= 1)	LREAL, 2, nChannels x nWindowLength- nOverlap	LREAL, 2, nChannels x nMaxBands

```

VAR_INPUT
    stInitPars      : ST_CM_MultiBandRMS_InitPars;      // init parameter
    nOwnID          : UDINT;                            // ID for this FB instance
    aDestIDs        : ARRAY[1..cCMA_MaxDest] OF UDINT;  // IDs of destinations for output
    nResultBuffers  : UDINT := 4;                      // number of MultiArrays which should be in
initialized for results (0 for no initialization)
    tTransferTimeout : LTIME := LTIME#500US;           // timeout checking off during access to in
ter-task FIFOs
END_VAR

```

入力パラメータ

このファンクションブロックの入力パラメータは初期化パラメータを表し、FBインスタンスの宣言で割り当て済みである必要があります。(または、Init()メソッド) 一度しか割り当てできません。実行中の変更はできません。

- **stInitPars:** ST_CM_MultiBandRMS_InitPars [▶ 291]型の初期化パラメータを持つファンクションブロック固有の構造体。パラメータは、上記の入出力バッファの定義と紐づいていなければなりません。
- **nOwnID:**一意のIDでファンクションブロックインスタンスを識別します。これは常に0より大きくなければなりません。この目的のためには、列挙を定義するアプローチが確立されています。
- **aDestIDs:**転送先のIDを指定することで、結果の転送先を定義します。出力バッファの定義（上記を参照）は、選択された各転送先の入力バッファの定義と紐づいていなければなりません。
- **nResultBuffers:**このファンクションブロックは、指定された数のマルチアレイバッファにより転送トレイストリームを初期化します。デフォルト値は4です。
- **tTransferTimeout:**内部マルチアレイ転送の同期タイムアウトの設定です。[並列処理 \[▶ 72\]](#)の項を参照してください。

出力パラメータ

```
VAR_OUTPUT
    bError      : BOOL;                // TRUE if an error occurs. Reset by next method call.
    hrErrorCode  : HRESULT;            // '< 0' = error; '> 0' = info; '0' = no error/info
    ipErrorMessage : I_TcMessage := fbErrorMessage; // Shows detailed information about occurred errors, warnings and more.
    nCntResults   : ULINT;             // Counts outgoing results (MultiArrays were calculated and sent to transfer tray).
END_VAR
```

- **bError:**エラーが発生した場合の出力は、TRUEです。
- **hrErrorCode:**エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。
- **ipErrorMessage:**現在の戻り値に関するより詳細な情報を含みます。[エラーの説明と情報 \[▶ 81\]](#)の項を参照してください。この特別なインターフェースポインタは、常に有効かつ割り当て済みであるよう、内部的に保護されています。

メソッド

```
METHOD Init : HRESULT
VAR_INPUT
    stInitPars      : ST_CM_MultiBandRMS_InitPars; // init parameter
    nOwnID          : UDINT;                       // ID for this FB instance
    aDestIDs        : ARRAY[1..cCMA_MaxDest] OF UDINT; // IDs of destinations for output
    nResultBuffers  : UDINT := 4; // number of MultiArrays which should be initialized for results (0 for no initialization)
END_VAR
```

Call():

このメソッドは、現在の入力データにアルゴリズムを適用するために、サイクルごとに呼び出されます。メソッドが新しい結果もエラーも示さない場合、ファンクションブロックは入力データを待機します。これは、解析チェーンのプロセスにおける規則的な動作です。

- **戻り値:** エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD Call : HRESULT
VAR_OUTPUT
    bNewResult      : BOOL;                // TRUE every time when outgoing MultiArray was calculated and sent to transfer tray.
    bError          : BOOL;                // TRUE if an error occurs.
    hrErrorCode      : HRESULT;            // '< 0' = error; '> 0' = info; '0' = no error/info
END_VAR
```

- **bError:**エラーが発生した場合の出力は、TRUEです。
- **hrErrorCode:**エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。この出力はメソッドの戻り値と同じです。



タイムアウトが発生した場合、または結果を格納するマルチアレイバッファがない場合、入力データも結果データも失われません。次の呼び出しに転送されます。

Configure() :

このメソッドを呼び出す際には、最初に周波数帯域を設定してください。対応するPLC配列は以下のように定義してください。Configure()メソッドは、異なる引数セットを持つ新しい設定にも使用できます。

- **戻り値** : エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD Configure : HRESULT
VAR_INPUT
    pArg      : POINTER TO LREAL; // pointer to array (LREAL) of arguments
    nArgSize  : UDINT;           // size of arguments buffer in bytes
END_VAR
```

入力バッファは、以下の定義（入力形状）のいずれかに紐づいています。変数パラメータは、ファンクションブロック入力stInitPars の一部です。周波数帯域とチャンネルごとに設定する2つのパラメータは、[fLowerFrequencyLimit, fUpperFrequencyLimit] です。

バージョン	入力バッファ（マルチアレイ 入カストリーム） 要素タイプ、次元数、次元サイズ
全てのチャンネルで同一設定	LREAL, 1, 2
周波数帯域別の設定 (nMaxBands >= 1)	LREAL, 2, nMaxBands x 2
チャンネル別の設定 (nMaxBands = 1)	LREAL, 2, nChannels x 2
チャンネルと周波数帯域別の設定	LREAL, 3, nChannels x nMaxBands x 2

Init():

このメソッドは通常、状態監視アプリケーションでは必要ありません。これはファンクションブロック初期化の代替手段を提供するものです。Init() メソッドは、PLCの初期化フェーズ中にのみ呼び出すことができます。実行時には使用できません。メソッドFB_init または、属性'call_after_init' の使用を参照してください（TwinCAT PLCリファレンスを参照）。さらに、これはファンクションブロックのカプセル化を容易にします。

Init()メソッドを使用して初期化を行う場合、ファンクションブロックインスタンスの入力パラメータを宣言で代入することはできません。

- **戻り値** : エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。
- **stInitPars**: ST_CM_MultiBandRMS_InitPars [▶ 291]型の初期化パラメータを持つファンクションブロック固有の構造体。パラメータは、上記の入出力バッファの定義と紐づいていなければなりません。
- **nOwnID**: 一意のIDでファンクションブロックインスタンスを識別します。これは常に0より大きくなければなりません。この目的のためには、列挙を定義するアプローチが確立されています。
- **aDestIDs**: 転送先のIDを指定することで、結果の転送先を定義します。出力バッファの定義（上記を参照）は、選択された各転送先の入力バッファの定義と紐づいていなければなりません。
- **nResultBuffers**: このファンクションブロックは、指定された数のマルチアレイバッファにより転送ストリームを初期化します。デフォルト値は4です。

ResetData() :

このメソッドは、既に追加された全てのデータレコードを削除します。ファンクションブロックのメモリ特性を参照してください。ResetData() の後、Call() メソッドが再度呼び出された場合、有効な結果を計算するために内部メモリを補充する必要があります。

- **戻り値**：エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD ResetData : HRESULT
VAR_INPUT
END_VAR
```

PassInputs() :

API PLC リファレンス [▶ 79]で説明されているとおり、FB_CMA_Source インスタンスが呼び出され、信号データがターゲットブロックに転送される限り、解析チェーンの以降の全てのファンクションブロックを周期的に呼び出す必要があります。

アルゴリズムを一定時間実行しない方が有用な場合があります。例えば、いくつかのアルゴリズムは、事前に教示や設定をした後でのみ実行すべきです。ファンクションブロックは周期的に呼び出す必要がありますが、ファンクションブロックに到着したデータが[通信リング \[▶ 72\]](#)内で転送されれば十分です。これは、Call() メソッドの代わりにPassInputs() メソッドを使用して行われます。アルゴリズム自体はここでは呼び出されず、したがって結果も計算されず、出力バッファも生成されません。

- **戻り値**：エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD PassInputs : HRESULT
VAR_INPUT
END_VAR
```

GetChannelErrors() :

このメソッドは、複数のチャンネル（nChannels > 1）を処理する際に、チャンネル固有の戻り値のリストを照会できるようにします。呼び出しは、ファンクションブロックの戻り値が eCM_InfRTTime_AmbiguousChannelResults または eCM_ErrRTTime_ErrornousChannelResults のいずれかに該当する場合に有用です。

- **戻り値**：エラーコード一覧の読み込み処理に関する情報です。この値は、クエリが成功した場合は TRUE に、そうでない場合は FALSE に設定されます。

```
METHOD GetChannelErrors : BOOL
VAR_IN_OUT
aChannelErrors : ARRAY[*] OF HRESULT;
END_VAR
```

- aChannelErrors:HRESULT 型の長さnChannelsのエラーリストです。

類似したファンクションブロック

ファンクションブロックFB_CMA_IntegratedRMS [▶ 157]は、入力された時系列データを周波数帯域制限付きのRMS計算の前に時間積分できるという追加機能を備えています（オプションで2次まで）。

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM, Tc3_CM_Base



CM 3.1 では一部の機能がすでに使用可能です。[互換性 \[▶ 61\]](#)の項を参照してください。

5.1.26 FB_CMA_OrderPowerSpectrum

実数値入力信号の次数分析を計算します。

次数分析は回転機械の振動解析に用いられます。この次数分析の特長は、風力タービンやランピングモータなど、解析中に回転部品の回転速度が一定でない場合でも、信頼性の高い結果が得られることです。

周波数解析では、例えばFB_CMA_PowerSpectrum を使用して、時間信号（例えば振動信号）を周波数領域に変換します。スペクトル値は周波数で表します。一方、次数分析では、変換前の時間信号がエンコード信号を用いて変換され、シャフトの回転に沿って振動信号のサンプリング点が空間的に等間隔となるよう補間さ

れます。以下の考察は、基本的な考え方を理解するのに役立ちます。振動信号と位置信号は時間的に等距離でサンプリングされます。波速度が一定であれば、振動信号は単純にシャフトの回転に沿った等距離の位置に投影できます。シャフトの回転中に速度が変化すると、サンプリング点の間隔は等間隔ではなくなります。例えば、シャフトの速度が上がると、2つのサンプリング点の間隔が広がります。これは、高速時には2つの時間的なサンプリングの間にシャフトがより多く回転するためです。エンコード信号を含めることで、この影響を適切な補間によって補正できます。その後、パワースペクトル (FB_CMA_PowerSpectrum) が計算されます。この場合、次数スペクトルは周波数ではなく、次数によって示されます。

周波数軸は、観測されたシャフトの最高速度fMaxRPM (1分あたりの回転数) でスケーリングされます。オーダー軸の場合、次のようになります：

以下は周波数軸に適用されます：

```
fSampleRate      := cOversamples * (1000.0 / fTaskCycleTime);
fResolutionFreq  := fSampleRate / cFFTLenght;
fNyquistFreq     := fSampleRate / 2;
```

以下はオーダー軸に適用されます：

```
fMaxOrder        := fNyquistFreq / (fMaxRPM / 60);
fResolutionOrder := fResolutionFreq / (fMaxRPM / 60);
```

パラメータは、構造体ST_CM_OrderPowerSpectrum_InitPars [▶ 293]を介して設定します。

ファンクションブロックFB_CMA_OrderPowerSpectrum は、各チャンネルに対して1つの振動信号と1つの位置信号を想定しています。これらは、ファンクションブロックFB_CMA_SourcePaired [▶ 236]によって、解析チェーンにタプル形式で導入されます。

メモリ特性

ウェルチ法を使用しているため、現在の入力データバッファと最後に転送されたバッファの両方が計算に使用されます。組み込まれるバッファの数は、選択したオーバーラップ(nOverlap)に依存します。周波数解析では、時系列データの段階的な変化が考慮されます。したがって、正しい結果を得るためには、使用する入力データバッファは、間隔やジャンプなしに連結していなければなりません。

NaN (非数) の発生

入力ベクトルに1つ以上のNaN (非数) 値が含まれる場合、出力セクタ全体がNaNで満たされます。詳細については別項NaN値 [▶ 71]を参照してください。

● NaN値の取り扱い

i NaN値につながる上記の状況を除外できない、または安全に無視できない場合、アプリケーションプログラムはこれらのエラー値を適切に処理できるようにしてください。

マルチチャンネル入力データ処理時の動作

複数のチャンネル (nChannels > 1) を処理する場合、各チャンネルが異なる戻り値になる可能性があります。この場合、戻り値はファンクションブロックで個別に問い合わせできます。全てのチャンネルではなく、1つ以上のチャンネルの結果が許容されない場合、ファンクションブロックの値はeCM_InfRTime_AmbiguousChannelResultsに対応します。全てのチャンネルの結果が許容されない場合、ファンクションブロック上の値はeCM_ErrRTime_ErrornousChannelResults に対応します。

全チャンネルの戻り値のリストは、GetChannelErrors()のメソッドで照会できます。

サンプルの実装

実装用のサンプルは以下のリンクからご覧いただけます：[次数分析 \[▶ 349\]](#)

入力と出力

入力バッファと出力バッファは、以下の定義 (入力/出力形状) のいずれかに該当します。変数パラメータは、ファンクションブロック入力stInitPars の一部です。

以下は、シングルチャンネルを処理する場合に適用されます (nChannels = 1)：

マルチアレイの所在	要素の型	次元	次元変数
入カストリームA (振動)	LREAL	1	nWindowLength - nOverlap
入カストリームB (ポジション)	LREAL	1	(nWindowLength - nOverlap) / (fSampleRateSignal / fSampleRatePosition)
出カストリーム	LREAL	1	nFFT_Length/2+1

以下は、マルチチャンネルを処理する場合に適用されます (nChannels > 1) :

マルチアレイの所在	要素の型	次元	次元変数
入カストリームA (振動)	LREAL	2	nChannels x (nWindowLength - nOverlap)
入カストリームB (位置)	LREAL	1	(nWindowLength - nOverlap) / (fSampleRateSignal / fSampleRatePosition)
出カストリーム	LREAL	2	nChannels x nFFT_Length/2+1

入カパラメータ

このファンクションブロックの入カパラメータは初期化パラメータを表し、FBインスタンスの宣言で割り当て済みである必要があります。(または、Init()メソッド) 一度しか割り当てできません。実行中の変更はできません。

```
VAR_INPUT
    stInitPars      : ST_CM_OrderPowerSpectrum_InitPars; // init parameter
    nOwnID          : UDINT; // ID for this FB instance
    aDestIDs        : ARRAY[1..cCMA_MaxDest] OF UDINT; // IDs of destinations for output
    nResultBuffers  : UDINT := 4; // number of MultiArrays which should be
// initialized for results (0 for no initialization)
    tTransferTimeout : LTIME := LTIME#500US; // timeout checking off during access to
// inter-task FIFOs
END_VAR
```

- **stInitPars:** [ST_CM_OrderPowerSpectrum_InitPars \[▶ 293\]](#) 型の初期化パラメータを持つファンクションブロック固有の構造体。パラメータは、上記の入出力バッファの定義と紐づいていなければなりません。
- **nOwnID:**一意のIDでファンクションブロックインスタンスを識別します。これは常に0より大きくなければなりません。この目的のためには、列挙を定義するアプローチが確立されています。
- **aDestIDs:**転送先のIDを指定することで、結果の転送先を定義します。出力バッファの定義（上記を参照）は、選択された各転送先の入力バッファの定義と紐づいていなければなりません。
- **nResultBuffers:**このファンクションブロックは、指定された数のマルチアレイバッファにより転送トレイストリームを初期化します。デフォルト値は4です。
- **tTransferTimeout:**内部マルチアレイ転送の同期タイムアウトの設定です。[並列処理 \[▶ 72\]](#)の項を参照してください。

出カパラメータ

```
VAR_OUTPUT
    bError          : BOOL; // TRUE if an error occurs. Reset by next method call.
    hrErrorCode      : HRESULT; // '< 0' = error; '> 0' = info; '0' = no error/info
    ipErrorMessage : I_TcMessage := fbErrorMessage; // Shows detailed information about occurred errors, warnings and more.
    nCntResults     : ULINT; // Counts outgoing results (MultiArrays were calculated and sent to transfer tray).
END_VAR
```

- **bError:**エラーが発生した場合の出力は、TRUEです。
- **hrErrorCode:**エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

- **ipErrorMessage:**現在の戻り値に関するより詳細な情報を含みます。エラーの説明と情報 [▶ 81]の項を参照してください。この特別なインターフェースポイントは、常に有効かつ割り当て済みであるよう、内部的に保護されています。

メソッド

```
METHOD Init : HRESULT
VAR_INPUT
    stInitPars      : ST_CM_OrderPowerSpectrum_InitPars; // init parameter
    nOwnID          : UDINT; // ID for this FB instance
    aDestIDs        : ARRAY[1..cCMA_MaxDest] OF UDINT; // IDs of destinations for output
    nResultBuffers  : UDINT := 4; // number of MultiArrays which should be in
                                // itialized for results (0 for no initialization)
END_VAR
```

Call():

このメソッドは、現在の入力データにアルゴリズムを適用するために、サイクルごとに呼び出されます。メソッドが新しい結果もエラーも示さない場合、ファンクションブロックは入力データを待機します。これは、解析チェーンのプロセスにおける規則的な動作です。

- **戻り値:** エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、エラーコード一覧 [▶ 367]に記載されています。

```
METHOD Call : HRESULT
VAR_OUTPUT
    bNewResult      : BOOL; // TRUE every time when outgoing MultiArray was calculated and sent t
    o transfer tray.
    bError          : BOOL; // TRUE if an error occurs.
    hrErrorCode      : HRESULT; // '< 0' = error; '> 0' = info; '0' = no error/info
END_VAR
```

- **bError:** エラーが発生した場合の出力は、TRUEです。
- **hrErrorCode:** エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、エラーコード一覧 [▶ 367]に記載されています。この出力はメソッドの戻り値と同じです。



タイムアウトが発生した場合、または結果を格納するマルチアレイバッファがない場合、入力データも結果データも失われません。次の呼び出しに転送されます。

Init():

このメソッドは通常、状態監視アプリケーションでは必要ありません。これはファンクションブロック初期化の代替手段を提供するものです。Init() メソッドは、PLCの初期化フェーズ中にのみ呼び出すことができます。実行時には使用できません。メソッドFB_init または、属性'call_after_init' の使用を参照してください (TwinCAT PLCリファレンスを参照)。さらに、これはファンクションブロックのカプセル化を容易にします。

Init()メソッドを使用して初期化を行う場合、ファンクションブロックインスタンスの入力パラメータを宣言で代入することはできません。

- **戻り値:** エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、エラーコード一覧 [▶ 367]に記載されています。
- **stInitPars:** ST_CM_OrderPowerSpectrum_InitPars [▶ 293] 型の初期化パラメータを持つファンクションブロック固有の構造体。パラメータは、上記の入出力バッファの定義と紐づいていなければなりません。
- **nOwnID:** 一意のIDでファンクションブロックインスタンスを識別します。これは常に0より大きくなければなりません。この目的のためには、列挙を定義するアプローチが確立されています。
- **aDestIDs:** 転送先のIDを指定することで、結果の転送先を定義します。出力バッファの定義（上記を参照）は、選択された各転送先の入力バッファの定義と紐づいていなければなりません。
- **nResultBuffers:** このファンクションブロックは、指定された数のマルチアレイバッファにより転送トレイストリームを初期化します。デフォルト値は4です。

ResetData() :

このメソッドは、既に追加された全てのデータレコードを削除します。ファンクションブロックのメモリ特性を参照してください。ResetData() の後、Call() メソッドが再度呼び出された場合、有効な結果を計算するために内部メモリを補充する必要があります。

- **戻り値**：エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD ResetData : HRESULT
VAR_INPUT
END_VAR
```

PassInputs() :

API PLC リファレンス [▶ 79]で説明されているとおり、FB_CMA_Source インスタンスが呼び出され、信号データがターゲットブロックに転送される限り、解析チェーンの以降の全てのファンクションブロックを周期的に呼び出す必要があります。

アルゴリズムを一定時間実行しない方が有用な場合があります。例えば、いくつかのアルゴリズムは、事前に教示や設定をした後でのみ実行すべきです。ファンクションブロックは周期的に呼び出す必要がありますが、ファンクションブロックに到着したデータが通信リング [▶ 72]内で転送されれば十分です。これは、Call() メソッドの代わりにPassInputs() メソッドを使用して行われます。アルゴリズム自体はここでは呼び出されず、したがって結果も計算されず、出力バッファも生成されません。

- **戻り値**：エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD PassInputs : HRESULT
VAR_INPUT
END_VAR
```

GetChannelErrors() :

このメソッドは、複数のチャンネル（nChannels > 1）を処理する際に、チャンネル固有の戻り値のリストを照会できるようにします。呼び出しは、ファンクションブロックの戻り値が eCM_InfRTTime_AmbiguousChannelResults または eCM_ErrRTTime_ErrornousChannelResults のいずれかに該当する場合に有効です。

- **戻り値**：エラーコード一覧の読み込み処理に関する情報です。この値は、クエリが成功した場合は TRUE に、そうでない場合は FALSE に設定されます。

```
METHOD GetChannelErrors : BOOL
VAR_IN_OUT
aChannelErrors : ARRAY[*] OF HRESULT;
END_VAR
```

- aChannelErrors: HRESULT 型の長さnChannelsのエラーリストです。

ファンクションブロックFB_CMA_PowerSpectrum [▶ 189] は、実数値入力データのパワースペクトルを計算します。

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM, Tc3_CM_Base

5.1.27 FB_CMA_PowerCepstrum

実数値入力信号のパワーケプストラムを計算します。

パワーケプストラム $C_p(\tau)$ は、この場合、時間連続表現で次のように定義されます：

$$C_p(\tau) = X^{-1} \{ \log(|X(\omega)|^2) \} = X^{-1} \{ 2 \log |X(\omega)| \}, \quad \text{Quefreny } \tau$$

したがって、対数化されたパワースペクトルの逆フーリエ変換として定義されます（FB_CMA_PowerSpectrum [▶ 189]を参照）。順変換と逆変換は、結果を時間領域に戻します。

このファンクションブロックはギアユニットの監視 [▶ 45]に役立ちます。

数値的な実装では、PowerSpectrumが最初に計算されます。入力データバッファは、直前のバッファとオーバーラップした後、ウィンドウ関数で乗算されます。パラメータnFFT_Lengthの値がnWindowLengthより大きい場合、ウィンドウ付き時間信号は、必要なFFT入力長に達するよう最初と最後を同じ数のゼロで埋められます（ゼロパディング）。その後、実数値に対してFFTが適用され、得られた複素数の絶対値とその2乗が計算されます。次にその値を対数化します。対数化の前に、引数はパラメータfLogThresholdの値と比較されます。それより小さい場合は、値域エラーやゼロの対数を計算しようとする試みを回避するため、この値に設定されます。続いて、もう1度逆フーリエ変換が行われます。結果は複素数の値を持つ配列です。

● 複素数値の結果の評価

1 実際には、アプリケーションに応じて、絶対値、絶対値の2乗、または複素数値のパワーケプストラムの実数部のみが評価されます。これはユーザが必要に応じて実装しなければなりません。

パワーケプストラムには、微妙に異なる定義がいくつも存在します。ここで使われている定義は、J.コレラースとロバートB.ランダルによる一般的な定義に基づいています。参考文献 [▶ 59]を参照してください。

複素ケプストラムとの違い：

パワーケプストラムは複素ケプストラムとは異なり、複素信号スペクトルの対数化フーリエ逆変換として定義されます。パワーケプストラムは絶対値計算のため、複素ケプストラムに比べて信号の位相角の特性に影響されにくいです。さらに、複素ケプストラムは実数値の結果を提供します。

メモリ特性

ウェルチ法を使用しているため、現在の入力データバッファと最後に転送されたバッファの両方が計算に使用されます。組み込まれるバッファの数は、選択したオーバーラップ(nOverlap)に依存します。周波数解析では、時系列データの段階的な変化が考慮されます。したがって、正しい結果を得るためには、使用する入力データバッファは、間隔やジャンプなしに連結していなければなりません。

NaN（非数）の発生

入力ベクトルに1つ以上のNaN（非数）値が含まれる場合、出力セクタ全体がNaNで満たされます。詳細については別項NaN値 [▶ 71]を参照してください。

● NaN値の取り扱い

1 NaN値につながる上記の状況を除外できない、または安全に無視できない場合、アプリケーションプログラムはこれらのエラー値を適切に処理できるようにしてください。

マルチチャンネル入力データ処理時の動作

複数のチャンネル（nChannels > 1）を処理する場合、各チャンネルが異なる戻り値になる可能性があります。この場合、戻り値はファンクションブロックで個別に問い合わせできます。全てのチャンネルではなく、1つ以上のチャンネルの結果が許容されない場合、ファンクションブロックの値はeCM_InfRTime_AmbiguousChannelResultsに対応します。全てのチャンネルの結果が許容されない場合、ファンクションブロック上の値はeCM_ErrRTime_ErrorousChannelResultsに対応します。

全チャンネルの戻り値のリストは、GetChannelErrors()のメソッドで照会できます。

サンプルの実装

実装用のサンプルは以下のリンクから入手できます：[パワーケプストラム \[▶ 346\]](#)

入力と出力

入力バッファと出力バッファは、以下の定義（入力/出力形状）のいずれかに該当します。変数パラメータは、ファンクションブロック入力stInitParsの一部です。

バージョン	入力バッファ（マルチアレイ入力ストリーム） 要素タイプ、次元数、次元サイズ	出力バッファ（マルチアレイ出力ストリーム） 要素タイプ、次元数、次元サイズ
標準バージョン (nChannels = 1)	LREAL, 1, nWindowLength-nOverlap	LCOMPLEX, 1, nFFT_Length/2+1

バージョン	入力バッファ（マルチアレイ入力ストリーム） 要素タイプ、次元数、次元サイズ	出力バッファ（マルチアレイ出力ストリーム） 要素タイプ、次元数、次元サイズ
マルチチャンネルバージョン (nChannels > 1)	LREAL, 2, nChannels x nWindowLength- nOverlap	LCOMPLEX, 2, nChannels x nFFT_Length/2+1

```

VAR_INPUT
    stInitPars      : ST_CM_PowerCepstrum_InitPars;      // init parameter
    nOwnID           : UDINT;                             // ID for this FB instance
    aDestIDs         : ARRAY[1..cCMA_MaxDest] OF UDINT;   // IDs of destinations for output
    nResultBuffers   : UDINT := 4;                       // number of MultiArrays which should be in
    itialized for results (0 for no initialization)
    tTransferTimeout : LTIME := LTIME#500US;             // timeout checking off during access to in
    ter-task FIFOs
END_VAR

```

入力パラメータ

このファンクションブロックの入力パラメータは初期化パラメータを表し、FBインスタンスの宣言で割り当て済みである必要があります。(または、Init()メソッド) 一度しか割り当てできません。実行中の変更はできません。

- **stInitPars:** ST_CM_PowerCepstrum_InitPars [▶ 294]型の初期化パラメータを持つファンクションブロック固有の構造体。パラメータは、上記の入出力バッファの定義と紐づいていなければなりません。
- **nOwnID:**一意のIDでファンクションブロックインスタンスを識別します。これは常に0より大きくなければなりません。この目的のためには、列挙を定義するアプローチが確立されています。
- **aDestIDs:**転送先のIDを指定することで、結果の転送先を定義します。出力バッファの定義（上記を参照）は、選択された各転送先の入力バッファの定義と紐づいていなければなりません。
- **nResultBuffers:**このファンクションブロックは、指定された数のマルチアレイバッファにより転送トレイストリームを初期化します。デフォルト値は4です。
- **tTransferTimeout:**内部マルチアレイ転送の同期タイムアウトの設定です。並列処理 [▶ 72]の項を参照してください。

出力パラメータ

```

VAR_OUTPUT
    bError          : BOOL;                               // TRUE if an error occurs. Reset by next metho
    d call.
    hrErrorCode      : HRESULT;                           // '< 0' = error; '> 0' = info; '0' = no error/
    info
    ipErrorMessage  : I_TcMessage := fbErrorMessage;     // Shows detailed information about occurred er
    rors, warnings and more.
    nCntResults      : ULINT;                             // Counts outgoing results (MultiArrays were ca
    lculated and sent to transfer tray).
END_VAR

```

- **bError:**エラーが発生した場合の出力は、TRUEです。
- **hrErrorCode:**エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、エラーコード一覧 [▶ 367]に記載されています。
- **ipErrorMessage:**現在の戻り値に関するより詳細な情報を含みます。エラーの説明と情報 [▶ 81]の項を参照してください。この特別なインターフェースポインタは、常に有効かつ割り当て済みであるよう、内部的に保護されています。

メソッド

```

METHOD Init : HRESULT
VAR_INPUT
    stInitPars      : ST_CM_PowerCepstrum_InitPars;      // init parameter
    nOwnID           : UDINT;                             // ID for this FB instance
    aDestIDs         : ARRAY[1..cCMA_MaxDest] OF UDINT;   // IDs of destinations for output
    nResultBuffers   : UDINT := 4;                       // number of MultiArrays which should be initialized for results
    (0 for no initialization)
END_VAR

```

Call():

このメソッドは、現在の入力データにアルゴリズムを適用するために、サイクルごとに呼び出されます。メソッドが新しい結果もエラーも示さない場合、ファンクションブロックは入力データを待機します。これは、解析チェーンのプロセスにおける規則的な動作です。

- **戻り値**：エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

METHOD Call : HRESULT

```
VAR_OUTPUT
    bNewResult    : BOOL;           // TRUE every time when outgoing MultiArray was calculated and sent to transfer tray.
    bError        : BOOL;           // TRUE if an error occurs.
    hrErrorCode    : HRESULT;       // '< 0' = error; '> 0' = info; '0' = no error/info
END_VAR
```

- **bError**:エラーが発生した場合の出力は、TRUEです。
- **hrErrorCode**:エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。この出力はメソッドの戻り値と同じです。



タイムアウトが発生した場合、または結果を格納するマルチアレイバッファがない場合、入力データも結果データも失われません。次の呼び出しに転送されます。

Init():

このメソッドは通常、状態監視アプリケーションでは必要ありません。これはファンクションブロック初期化の代替手段を提供するものです。Init() メソッドは、PLCの初期化フェーズ中にのみ呼び出すことができます。実行時には使用できません。メソッドFB_init または、属性'call_after_init' の使用を参照してください (TwinCAT PLCリファレンスを参照)。さらに、これはファンクションブロックのカプセル化を容易にします。

Init()メソッドを使用して初期化を行う場合、ファンクションブロックインスタンスの入力パラメータを宣言で代入することはできません。

- **戻り値**：エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。
- **stInitPars**:ST_CM_PowerCepstrum_InitPars [▶ 294]型の初期化パラメータを持つファンクションブロック固有の構造体。パラメータは、上記の入出力バッファの定義と紐づいていなければなりません。
- **nOwnID**:一意のIDでファンクションブロックインスタンスを識別します。これは常に0より大きくなければなりません。この目的のためには、列挙を定義するアプローチが確立されています。
- **aDestIDs**:転送先のIDを指定することで、結果の転送先を定義します。出力バッファの定義（上記を参照）は、選択された各転送先の入力バッファの定義と紐づいていなければなりません。
- **nResultBuffers**:このファンクションブロックは、指定された数のマルチアレイバッファにより転送ストリームを初期化します。デフォルト値は4です。

ResetData() :

このメソッドは、既に追加された全てのデータレコードを削除します。ファンクションブロックのメモリ特性を参照してください。ResetData() の後、Call() メソッドが再度呼び出された場合、有効な結果を計算するために内部メモリを補充する必要があります。

- **戻り値**：エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

METHOD ResetData : HRESULT

```
VAR_INPUT
END_VAR
```

PassInputs() :

API PLC リファレンス [▶ 79]で説明されているとおり、FB_CMA_Source インスタンスが呼び出され、信号データがターゲットブロックに転送される限り、解析チェーンの以降の全てのファンクションブロックを周期的に呼び出す必要があります。

アルゴリズムを一定時間実行しない方が有用な場合があります。例えば、いくつかのアルゴリズムは、事前に教示や設定をした後でのみ実行すべきです。ファンクションブロックは周期的に呼び出す必要があります

が、ファンクションブロックに到着したデータが通信リング [▶ 72]内で転送されれば十分です。これは、Call() メソッドの代わりにPassInputs() メソッドを使用して行われます。アルゴリズム自体はここでは呼び出されず、したがって結果も計算されず、出力バッファも生成されません。

- **戻り値**：エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD PassInputs : HRESULT
VAR_INPUT
END_VAR
```

GetChannelErrors() :

このメソッドは、複数のチャンネル（nChannels > 1）を処理する際に、チャンネル固有の戻り値のリストを照会できるようにします。呼び出しは、ファンクションブロックの戻り値が eCM_InfRTTime_AmbiguousChannelResults または eCM_ErrRTTime_ErrornousChannelResults のいずれかに該当する場合に有効です。

- **戻り値**：エラーコード一覧の読み込み処理に関する情報です。この値は、クエリが成功した場合は TRUE に、そうでない場合は FALSE に設定されます。

```
METHOD GetChannelErrors : BOOL
VAR_IN_OUT
aChannelErrors : ARRAY[*] OF HRESULT;
END_VAR
```

- aChannelErrors:HRESULT 型の長さnChannelsのエラーリストです。

類似したファンクションブロック

ファンクションブロックFB_CMA_Envelope [▶ 137]は、線形および非線形システム成分を持つパルス状励振の解析にも適しています。

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM, Tc3_CM_Base



CM 3.1 では一部の機能がすでに使用可能です。[互換性 \[▶ 61\]](#)の項を参照してください。

5.1.28 FB_CMA_PowerSpectrum

実数値入力信号のパワースペクトルを計算し、オプションでデシベルスケールリングを適用します。

ファンクションブロックは、実数値入力信号のパワースペクトル（コレログラムまたはピリオドグラムとも呼ばれる）を計算します。ファンクションブロックは、複数の機能を実行します。[データストリーム解析 \[▶ 16\]](#)と [周波数解析 \[▶ 34\]](#)を参照してください：

入力データバッファは、まず直前のバッファとオーバーラップした後、ウィンドウ関数で乗算されます。パラメータnFFT_Length の値がnWindowLength より大きい場合、ウィンドウ付き時間信号は、必要なFFT入力長に達するよう最初と最後を同じ数のゼロで埋められます（ゼロパディング）。その後、実数値のFFTが適用され、結果として得られる複素数の絶対値が計算されます。パラメータbTransformToDecibel がTRUE の場合、値はデシベル値に変換されます。これらのデシベル値は、マグニチュードスペクトルとパワースペクトルにおいて同じです。つまり、マグニチュードスペクトルでは、デシベル値の計算において2乗の影響が考慮されており、その影響は2倍の係数として反映されています。さらに、マグニチュードスペクトルは、パラメータeScalingType を使ってスケールリングすることができます（[スペクトルのスケールリング \[▶ 25\]](#)を参照）。

FB_CMA_PowerSpectrum ファンクションブロックは、[FB_CMA_MagnitudeSpectrum \[▶ 161\]](#) と同様の動作をします。違いは、[FB_CMA_PowerSpectrum \[▶ 189\]](#)の結果は2乗になることです。

多くの場合、短期的なパワースペクトルは信号のスペクトルを統計的に推定するのに適していない。多くの場合、推定値の揺らぎは、複数の周波数や連続するスペクトルの平均化によって低減されるべきです。

スケーリング

加速度スペクトル密度などの結果値のスケーリングは、デフォルトでFFTの定義と一致しています。これは、ウィンドウ長とウィンドウ関数の影響が排除されることを意味します。
スペクトルスケーリングオプション [▶ 371] の項で説明されている表形式のパラメータは、絶対計測の算術スケーリングに使用できます。

メモリ特性

ウェルチ法を使用しているため、現在の入力データバッファと最後に転送されたバッファの両方が計算に使用されます。組み込まれるバッファの数は、選択したオーバーラップ(nOverlap)に依存します。
 周波数解析では、時系列データの段階的な変化が考慮されます。したがって、正しい結果を得るためには、使用する入力データバッファは、間隔やジャンプなしに連結していなければなりません。

NaN（非数）の発生

入力ベクトルに1つ以上のNaN（非数）値が含まれる場合、出力セクタ全体がNaNで満たされます。詳細については別項NaN値 [▶ 71]を参照してください。

● NaN値の取り扱い

i NaN値につながる上記の状況を除外できない、または安全に無視できない場合、アプリケーションプログラムはこれらのエラー値を適切に処理できるようにしてください。

マルチチャンネル入力データ処理時の動作

複数のチャンネル (nChannels > 1) を処理する場合、各チャンネルが異なる戻り値になる可能性があります。この場合、戻り値はファンクションブロックで個別に問い合わせできます。全てのチャンネルではなく、1つ以上のチャンネルの結果が許容されない場合、ファンクションブロックの値はeCM_InfRTTime_AmbiguousChannelResultsに対応します。全てのチャンネルの結果が許容されない場合、ファンクションブロック上の値はeCM_ErrRTTime_ErrornousChannelResults に対応します。

全チャンネルの戻り値のリストは、GetChannelErrors()のメソッドで照会できます。

入力と出力

入力バッファと出力バッファは、以下の定義（入力/出力形状）のいずれかに該当します。変数パラメータは、ファンクションブロック入力stInitPars の一部です。

バージョン	入力バッファ（マルチアレイ入力ストリーム） 要素タイプ、次元数、次元サイズ	出力バッファ（マルチアレイ出力ストリーム） 要素タイプ、次元数、次元サイズ
標準バージョン (nChannels = 1)	LREAL, 1, nWindowLength-nOverlap	LREAL, 1, nFFT_Length/2+1
マルチチャンネルバージョン (nChannels > 1)	LREAL, 2, nChannels x nWindowLength-nOverlap	LREAL, 2, nChannels x nFFT_Length/2+1

入力パラメータ

このファンクションブロックの入力パラメータは初期化パラメータを表し、FBインスタンスの宣言で割り当て済みである必要があります。(または、Init()メソッド) 一度しか割り当てできません。実行中の変更はできません。

```
VAR_INPUT
    stInitPars      : ST_CM_PowerSpectrum_InitPars;           // init parameter
    nOwnID          : UDINT;                                   // ID for this FB instance
    aDestIDs        : ARRAY[1..cCMA_MaxDest] OF UDINT;        // IDs of destinations for output
    nResultBuffers  : UDINT := 4;                             // number of MultiArrays which should be in
    itialized for results (0 for no initialization)
    tTransferTimeout : LTIME := LTIME#500US;                 // timeout checking off during access to in
    ter-task FIFOs
END_VAR
```

- stInitPars:ST_CM_PowerSpectrum_InitPars [▶ 295]型の初期化パラメータを持つファンクションブロック固有の構造体。パラメータは、上記の入出力バッファの定義と紐づいていなければなりません。

- **nOwnID**:一意のIDでファンクションブロックインスタンスを識別します。これは常に0より大きくなければなりません。この目的のためには、列挙を定義するアプローチが確立されています。
- **aDestIDs**:転送先のIDを指定することで、結果の転送先を定義します。出力バッファの定義（上記を参照）は、選択された各転送先の入力バッファの定義と紐づいていなければなりません。
- **nResultBuffers**:このファンクションブロックは、指定された数のマルチアレイバッファにより転送トレイストリームを初期化します。デフォルト値は4です。
- **tTransferTimeout**:内部マルチアレイ転送の同期タイムアウトの設定です。[並列処理 \[▶ 72\]](#)の項を参照してください。

出力パラメータ

```
VAR_OUTPUT
  bError      : BOOL;           // TRUE if an error occurs. Reset by next method call.
  hrErrorCode  : HRESULT;       // '< 0' = error; '> 0' = info; '0' = no error/info
  ipErrorMessage : I_TcMessage := fbErrorMessage; // Shows detailed information about occurred errors, warnings and more.
  nCntResults   : ULINT;        // Counts outgoing results (MultiArrays were calculated and sent to transfer tray).
END_VAR
```

- **bError**:エラーが発生した場合の出力は、TRUEです。
- **hrErrorCode**:エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。
- **ipErrorMessage**:現在の戻り値に関するより詳細な情報を含みます。[エラーの説明と情報 \[▶ 81\]](#)の項を参照してください。この特別なインターフェースポイントは、常に有効かつ割り当て済みであるよう、内部的に保護されています。

メソッド

```
METHOD Init : HRESULT
VAR_INPUT
  stInitPars      : ST_CM_PowerSpectrum_InitPars; // init parameter
  nOwnID          : UDINT;                       // ID for this FB instance
  aDestIDs        : ARRAY[1..cCMA_MaxDest] OF UDINT; // IDs of destinations for output
  nResultBuffers  : UDINT := 4;                 // number of MultiArrays which should be initialized for results (0 for no initialization)
END_VAR
```

Call():

このメソッドは、現在の入力データにアルゴリズムを適用するために、サイクルごとに呼び出されます。メソッドが新しい結果もエラーも示さない場合、ファンクションブロックは入力データを待機します。これは、解析チェーンのプロセスにおける規則的な動作です。

- **戻り値**: エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD Call : HRESULT
VAR_OUTPUT
  bNewResult      : BOOL;           // TRUE every time when outgoing MultiArray was calculated and sent to transfer tray.
  bError          : BOOL;           // TRUE if an error occurs.
  hrErrorCode      : HRESULT;       // '< 0' = error; '> 0' = info; '0' = no error/info
END_VAR
```

- **bError**:エラーが発生した場合の出力は、TRUEです。
- **hrErrorCode**:エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。この出力はメソッドの戻り値と同じです。



タイムアウトが発生した場合、または結果を格納するマルチアレイバッファがない場合、入力データも結果データも失われません。次の呼び出しに転送されます。

Init():

このメソッドは通常、状態監視アプリケーションでは必要ありません。これはファンクションブロック初期化の代替手段を提供するものです。Init() メソッドは、PLCの初期化フェーズ中にのみ呼び出すことができます。実行時には使用できません。メソッドFB_init または、属性'call_after_init'の使用を参照してください（TwinCAT PLCリファレンスを参照）。さらに、これはファンクションブロックのカプセル化を容易にします。

Init()メソッドを使用して初期化を行う場合、ファンクションブロックインスタンスの入力パラメータを宣言で代入することはできません。

- **戻り値** : エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。
- **stInitPars**: ST_CM_PowerSpectrum_InitPars [▶ 295]型の初期化パラメータを持つファンクションブロック固有の構造体。パラメータは、上記の入出力バッファの定義と紐づいていなければなりません。
- **nOwnID**: 一意のIDでファンクションブロックインスタンスを識別します。これは常に0より大きくなければなりません。この目的のためには、列挙を定義するアプローチが確立されています。
- **aDestIDs**: 転送先のIDを指定することで、結果の転送先を定義します。出力バッファの定義（上記を参照）は、選択された各転送先の入力バッファの定義と紐づいていなければなりません。
- **nResultBuffers**: このファンクションブロックは、指定された数のマルチアレイバッファにより転送ストリームを初期化します。デフォルト値は4です。

ResetData() :

このメソッドは、既に追加された全てのデータレコードを削除します。ファンクションブロックのメモリ特性を参照してください。ResetData() の後、Call() メソッドが再度呼び出された場合、有効な結果を計算するために内部メモリを補充する必要があります。

- **戻り値** : エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD ResetData : HRESULT
VAR_INPUT
END_VAR
```

PassInputs() :

API PLC リファレンス [▶ 79]で説明されているとおり、FB_CMA_Source インスタンスが呼び出され、信号データがターゲットブロックに転送される限り、解析チェーンの以降の全てのファンクションブロックを周期的に呼び出す必要があります。

アルゴリズムを一定時間実行しない方が有用な場合があります。例えば、いくつかのアルゴリズムは、事前に教示や設定をした後でのみ実行すべきです。ファンクションブロックは周期的に呼び出す必要がありますが、ファンクションブロックに到着したデータが通信リング [▶ 72]内で転送されれば十分です。これは、Call() メソッドの代わりにPassInputs() メソッドを使用して行われます。アルゴリズム自体はここでは呼び出されず、したがって結果も計算されず、出力バッファも生成されません。

- **戻り値** : エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD PassInputs : HRESULT
VAR_INPUT
END_VAR
```

GetChannelErrors() :

このメソッドは、複数のチャンネル（nChannels > 1）を処理する際に、チャンネル固有の戻り値のリストを照会できるようにします。呼び出しは、ファンクションブロックの戻り値が eCM_InfRTime_AmbiguousChannelResults または eCM_ErrRTime_ErrornousChannelResults のいずれかに該当する場合に有効です。

- **戻り値**: エラーコード一覧の読み込み処理に関する情報です。この値は、クエリが成功した場合は TRUE に、そうでない場合は FALSE に設定されます。

```
METHOD GetChannelErrors : BOOL
VAR_IN_OUT
    aChannelErrors : ARRAY[*] OF HRESULT;
END_VAR
```

- **aChannelErrors**: HRESULT 型の長さnChannelsのエラーリストです。

類似したファンクションブロック

FB_CMA_MagnitudeSpectrum [▶ 161] ファンクションブロックは、最後のステップで値の2乗を行わずにマグニチュードスペクトルを計算します。

FB_CMA_PowerCepstrum [▶ 185] ファンクションブロックは、高調波を強調する変換を行います。

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86, x64)	Tc3_CM, Tc3_CM_Base



CM 3.1 では一部の機能がすでに使用可能です。[互換性 \[▶ 61\]](#)の項を参照してください。

5.1.29 FB_CMA_Quantiles

シングルチャンネルおよびマルチチャンネルの実数値信号の入力値分布の分位数を計算します。

ファンクションブロックFB_CMA_Quantilesは、シングルチャンネルまたはマルチチャンネルの実数値入力データのp-分位点を計算します。各チャンネルは独立したチャンネルとして考慮されます。

このファンクションブロックは、初期状態では度数分布の計算に基づいています（FB_CMA_HistArray [▶ 145] を参照）。周波数分布の下限値、上限値、クラスの数（ビンとも呼ぶ）は、パラメータ設定のために転送されます。各クラスの制限値は、このように定義された全範囲にわたって同間隔で分布します（ヒストグラム [▶ 27] を参照）。次に累積度数分布が計算され、そこから設定された分位数が計算されます（統計解析 [▶ 27] を参照）。追加の設定パラメータは、各チャンネルに対して計算される分位の数です。

結果は実数値を持つ2次元配列です。最初のインデックスはチャンネル番号、2番目のインデックスはそれぞれの分位数です。

分類において下限を下回る値と上限を上回る値は、分位点の計算では無視されます。区間内では、分位値が補間されます。経験的累積度数分布がセクションごとに一定の場合、適切な最小値が使用されます。

メモリ特性

サンプル数Nは、新しいデータセットが入力されるたびに自動的に増加し、現在の結果を計算します。つまり、ファンクションブロックがインスタンス化されてからの全ての入力値を使用します。サンプル数のゼロリセット（FBの内部メモリの削除）は、ResetData() メソッド、またはCallEx() メソッドが使用される場合、変数bResetData によって提供されます。

設定

値を持つ2次元配列は、設定パラメータとしてファンクションブロックのConfigure() メソッドに転送されます。各値は、計算するチャンネルと分位の相対周波数を表します。次にファンクションブロックは、入力データに基づいて、各チャンネルの周波数分位数を計算します。設定された周波数は0.5で、これは中央値に相当します。あるいは、全てのチャンネルで使用される、計算された分位値の1次元配列を転送することもできます。

NaN（非数）の発生

チャンネルの結果がまだ利用可能でない場合、このチャンネルにはNaN（非数）が返されます。理由としては、入力データがまだ転送されていないか、これまでに転送されたデータが全てfMinBinned からfMaxBinned までの区間外であるか、または個々のチャンネルの入力値としてNaNだけが転送されたことが考えられます。

入力値のセットに特殊な定数 NaN が含まれる場合、この時間ステップのチャンネルの統計量に値は追加されない、つまり欠測の指標として扱われます。

● NaN値の取り扱い



NaN値につながる上記の状況を除外できない、または安全に無視できない場合、アプリケーションプログラムはこれらのエラー値を適切に処理できるようにしてください。

ルータメモリ

分位点計算はヒストグラムに基づく統計計算であり、多くのメモリを必要とします。メモリ使用量は、パラメータ `nChannels`、`nBins`、`nMaxQuantiles` に依存します。これらのパラメータはできるだけ小さくしておくことを推奨します。

マルチチャンネル入力データ処理時の動作

複数のチャンネル (`nChannels > 1`) を処理する場合、各チャンネルが異なる戻り値になる可能性があります。この場合、戻り値はファンクションブロックで個別に問い合わせできます。全てのチャンネルではなく、1つ以上のチャンネルの結果が許容されない場合、ファンクションブロックの値は `eCM_InfRTime_AmbiguousChannelResults` に対応します。全てのチャンネルの結果が許容されない場合、ファンクションブロック上の値は `eCM_ErrRTime_ErrornousChannelResults` に対応します。

全チャンネルの戻り値のリストは、`GetChannelErrors()` のメソッドで照会できます。

複数のサブチャンネル (`nSubChannels > 0`) を処理する場合、入出力データのフォーマットには特に注意が必要です。入力データが上流のファンクションブロックによるマルチチャンネルの結果で構成されている場合、`nChannels` の値を採用する必要があります。この場合、パラメータ `nSubChannels` を使用して追加の設定を行います。

例：マルチチャンネルスペクトル (`FB_CMA_MagnitudeSpectrum` など) の周波数チャンネルの統計的考察 (`FB_CMA_Quantiles` など) では、`nChannels` の値は、入力信号の数と同じである必要があります。サブチャンネルの数 `nSubChannels` は、スペクトルの長さに対応します。

サンプルの実装

実装用のサンプルは以下のリンクから入手できます：[周波数解析による状態監視 \[▶ 335\]](#)

入力と出力

入力バッファと出力バッファは、以下の定義（形状）に該当します。変数パラメータは、ファンクションブロック入力 `stInitPars` の一部です。

バージョン	入力バッファ（マルチアレイ入力ストリーム） 要素タイプ、次元数、次元サイズ	出力バッファ（マルチアレイ出力ストリーム） 要素タイプ、次元数、次元サイズ
標準バージョン (<code>nSubChannels = 0</code>)	LREAL, 1, <code>nChannels</code>	LREAL, 2, <code>nChannels x nMaxQuantiles</code>
複数データセット用標準バージョン (<code>nSubChannels = 0</code>)	LREAL, 2, <code>nChannelsx</code> 指定なし*	LREAL, 2, <code>nChannels x nMaxQuantiles</code>
マルチチャンネルバージョン (<code>nSubChannels > 0</code>)	LREAL, 2, <code>nChannels x nSubChannels</code>	LREAL, 3, <code>nChannels x nSubChannels x nMaxQuantiles</code>
複数データセット用マルチチャンネルバージョン (<code>nSubChannels > 0</code>)	LREAL, 3, <code>nChannels x nSubChannels x</code> 指定なし*	LREAL, 3, <code>nChannels x nSubChannels x nMaxQuantiles</code>

*:この次元の長さは任意に選択できるため、アプリケーションや先行のアルゴリズムの出力バッファに合わせる必要があります。

```
VAR_INPUT
    stInitPars      : ST_CM_Quantiles_InitPars;           // init parameter
    nOwnID          : UDINT;                             // ID for this FB instance
    aDestIDs        : ARRAY[1..cCMA_MaxDest] OF UDINT;   // IDs of destinations for output
    nResultBuffers  : UDINT := 4;                         // number of MultiArrays which should be in
    initialized for results (0 for no initialization)
```

```
tTransferTimeout : LTIME := LTIME#500US;           // timeout checking off during access to in
ter-task FIFOs
END_VAR
```

入力パラメータ

このファンクションブロックの入力パラメータは初期化パラメータを表し、FBインスタンスの宣言で割り当て済みである必要があります。(または、Init()メソッド) 一度しか割り当てできません。実行中の変更はできません。

- **stInitPars:** ST_CM_Quantiles_InitPars [▶ 296]型の初期化パラメータを持つファンクションブロック固有の構造体。パラメータは、上記の入出力バッファの定義と紐づいていなければなりません。
- **nOwnID:** 一意のIDでファンクションブロックインスタンスを識別します。これは常に0より大きくなければなりません。この目的のためには、列挙を定義するアプローチが確立されています。
- **aDestIDs:** 転送先のIDを指定することで、結果の転送先を定義します。出力バッファの定義（上記を参照）は、選択された各転送先の入力バッファの定義と紐づいていなければなりません。
- **nResultBuffers:** このファンクションブロックは、指定された数のマルチアレイバッファにより転送トレイストリームを初期化します。デフォルト値は4です。
- **tTransferTimeout:** 内部マルチアレイ転送の同期タイムアウトの設定です。[並列処理 \[▶ 72\]](#)の項を参照してください。

出力パラメータ

```
VAR_OUTPUT
    bError          : BOOL;                // TRUE if an error occurs. Reset by next metho
d call.
    hrErrorCode      : HRESULT;            // '< 0' = error; '> 0' = info; '0' = no error/
info
    ipErrorMessage  : I_TcMessage := fbErrorMessage; // Shows detailed information about occurred er
rors, warnings and more.
    nCntResults      : ULINT;              // Counts outgoing results (MultiArrays were ca
lculated and sent to transfer tray).
END_VAR
```

- **bError:** エラーが発生した場合の出力は、TRUEです。
- **hrErrorCode:** エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。
- **ipErrorMessage:** 現在の戻り値に関するより詳細な情報を含みます。[エラーの説明と情報 \[▶ 81\]](#)の項を参照してください。この特別なインターフェースポインタは、常に有効かつ割り当て済みであるよう、内部的に保護されています。

メソッド

Call():

このメソッドは、現在の入力データにアルゴリズムを適用するために、サイクルごとに呼び出されます。メソッドが新しい結果もエラーも示さない場合、ファンクションブロックは入力データを待機します。これは、解析チェーンのプロセスにおける規則的な動作です。

- **戻り値:** エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD Call : HRESULT
VAR_OUTPUT
    bNewResult      : BOOL;                // TRUE every time when outgoing MultiArray was calculated and sent t
o transfer tray.
    bError          : BOOL;                // TRUE if an error occurs.
    hrErrorCode      : HRESULT;            // '< 0' = error; '> 0' = info; '0' = no error/info
END_VAR
```

- **bError:** エラーが発生した場合の出力は、TRUEです。
- **hrErrorCode:** エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。この出力はメソッドの戻り値と同じです。



タイムアウトが発生した場合、または結果を格納するマルチアレイバッファがない場合、入力データも結果データも失われません。次の呼び出しに転送されます。

CallEx() :

このメソッドは、入力信号から分位数を計算するために各サイクルで呼び出されます。別の方法としてCall()があります。

分位値評価は一般的に、新しい入力値の登録よりも計算量が大幅に多くなります。したがって、Callex()の方法を使用すると、必要なときだけ統計結果を計算し、設定パラメータによっては実行時間をかなり短縮することができます。

メソッドが新しい結果もエラーも示さない場合、ファンクションブロックは入力データを待機します。これは、解析チェーンのプロセスにおける規則的な動作です。

- **戻り値** : エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD CallEx : HRESULT
VAR_INPUT
    nAppendData : UDINT;          // count of data buffers which are appended until calculation (1= calculate always)
    bResetData  : BOOL;           // automatic reset of dataset buffer after each calculation
END_VAR
VAR_OUTPUT
    bNewResult  : BOOL;           // TRUE every time when outgoing MultiArray was calculated and sent to transfer tray.
    bError      : BOOL;           // TRUE if an error occurs.
    hrErrorCode : HRESULT;        // '< 0' = error; '> 0' = info; '0' = no error/info
END_VAR
```

- **nAppendData**: 計算を実行する前に、いくつかの入力データバッファを収集するかを定義します。これは、正確な結果を得るために複数のデータブロックを追加することが望ましいためです。
- **bResetData**: これを設定した場合、内部データバッファは計算後に完全に削除されます。
- **bError**: エラーが発生した場合の出力は、TRUEです。
- **hrErrorCode**: エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。この出力はメソッドの戻り値と同じです。



タイムアウトが発生した場合、または結果を格納するマルチアレイバッファがない場合、入力データも結果データも失われません。次の呼び出しに転送されます。

Configure() :

分位数引数は、このメソッドの呼び出しの最初に設定しなければなりません。対応するPLC配列は以下のよう定義してください。Configure()メソッドは、異なる引数セットを持つ新しい設定にも使用できます。

- **戻り値** : エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧](#)に記載されています。

```
METHOD Configure : HRESULT
VAR_INPUT
    pArg : POINTER TO LREAL; // pointer to 2-dimensional array (LREAL) of arguments
    nArgSize : UDINT;         // size of arguments buffer in bytes
END_VAR
```

入力バッファは、以下の定義（入力形状）のいずれかに紐づいています。変数パラメータは、ファンクションブロック入力stInitParsの一部です。

バージョン	入力バッファ（マルチアレイ入カストリーム）要素タイプ、次元数、次元サイズ
全てのチャンネルおよびサブチャンネルで同一設定	LREAL, 1, nMaxQuantiles
チャンネル別の設定 (nSubChannels = 0)	LREAL, 2, nChannels x nMaxQuantiles
サブチャンネル別の設定 (nSubChannels > 0)	LREAL, 2, nSubChannels x nMaxQuantiles
チャンネルおよびサブチャンネル別の設定 (nSubChannels > 0)	LREAL, 3, nChannels x nSubChannels x nMaxQuantiles


```

METHOD Init : HRESULT
VAR_INPUT
    stInitPars      : ST_CM_Quantiles_InitPars; // init parameter
    nOwnID          : UDINT;                    // ID for this FB instance
    aDestIDs        : ARRAY[1..cCMA_MaxDest] OF UDINT; // IDs of destinations for output
    nResultBuffers  : UDINT := 4;              // number of MultiArrays which should be initialized for results (0 for no initialization)
END_VAR

```

Init():

このメソッドは通常、状態監視アプリケーションでは必要ありません。これはファンクションブロック初期化の代替手段を提供するものです。Init() メソッドは、PLCの初期化フェーズ中にのみ呼び出すことができます。実行時には使用できません。メソッドFB_init または、属性'call_after_init'の使用を参照してください（TwinCAT PLCリファレンスを参照）。さらに、これはファンクションブロックのカプセル化を容易にします。

Init()メソッドを使用して初期化を行う場合、ファンクションブロックインスタンスの入力パラメータを宣言で代入することはできません。

- **戻り値**：エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。
- **stInitPars**:ST_CM_Quantiles_InitPars [▶ 296]型の初期化パラメータを持つファンクションブロック固有の構造体。パラメータは、上記の入出力バッファの定義と紐づいていなければなりません。
- **nOwnID**:一意のIDでファンクションブロックインスタンスを識別します。これは常に0より大きくなければなりません。この目的のためには、列挙を定義するアプローチが確立されています。
- **aDestIDs**:転送先のIDを指定することで、結果の転送先を定義します。出力バッファの定義（上記を参照）は、選択された各転送先の入力バッファの定義と紐づいていなければなりません。
- **nResultBuffers**:このファンクションブロックは、指定された数のマルチアレイバッファにより転送ストリームを初期化します。デフォルト値は4です。

ResetData() :

このメソッドは、既に追加された全てのデータレコードを削除します。ファンクションブロックのメモリ特性を参照してください。ResetData() の後、Call() メソッドが再度呼び出された場合、有効な結果を計算するために内部メモリを補充する必要があります。

- **戻り値**：エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```

METHOD ResetData : HRESULT
VAR_INPUT
END_VAR

```

または、CallEx() の自動リセットを使用することもできます。

PassInputs() :

API PLC リファレンス [▶ 79]で説明されているとおり、FB_CMA_Source インスタンスが呼び出され、信号データがターゲットブロックに転送される限り、解析チェーンの以降の全てのファンクションブロックを周期的に呼び出す必要があります。

アルゴリズムを一定時間実行しない方が有用な場合があります。例えば、いくつかのアルゴリズムは、事前に教示や設定をした後でのみ実行すべきです。ファンクションブロックは周期的に呼び出す必要がありますが、ファンクションブロックに到着したデータが[通信リング \[▶ 72\]](#)内で転送されれば十分です。これは、Call() メソッドの代わりにPassInputs() メソッドを使用して行われます。アルゴリズム自体はここでは呼び出されず、したがって結果も計算されず、出力バッファも生成されません。

- **戻り値**：エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```

METHOD PassInputs : HRESULT
VAR_INPUT
END_VAR

```

GetChannelErrors() :

このメソッドは、複数のチャンネル（ $nChannels > 1$ ）を処理する際に、チャンネル固有の戻り値のリストを照会できるようにします。呼び出しは、ファンクションブロックの戻り値が `eCM_InfRTTime_AmbiguousChannelResults` または `eCM_ErrRTTime_ErrornousChannelResults` のいずれかに該当する場合に有効です。

- **戻り値:** エラーコード一覧の読み込み処理に関する情報です。この値は、クエリが成功した場合は TRUE に、そうでない場合は FALSE に設定されます。

```
METHOD GetChannelErrors : BOOL
VAR_IN_OUT
    aChannelErrors : ARRAY[*] OF HRESULT;
END_VAR
```

- `aChannelErrors:HRESULT` 型の長さ `nChannels` のエラーリストです。

類似したファンクションブロック

`FB_CMA_HistArray` [[▶ 145](#)] ファンクションブロックは、入力値分布のヒストグラムを計算します。

`FB_CMA_MomentCoefficients` [[▶ 172](#)] ファンクションブロックは、統計モーメント係数（平均値、標準偏差、歪度、尖度）を計算します。

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM, Tc3_CM_Base



CM 3.1 では一部の機能がすでに使用可能です。[互換性](#) [[▶ 61](#)]の項を参照してください。

5.1.30 FB_CMA_RainflowCounting

応力時間シーケンスで分類された反転点を計算します。

ファンクションブロック `FB_CMA_RainflowCounting` は、ASTM-E-1049-85に基づき、データストリームの処理に適した3点アルゴリズムを実装しています。カウントされたハーフサイクルは、2次元のマルチアレイ（平均応力と応力範囲）の形で返されます。

部品の疲労損傷は、ファンクションブロック `FB_CMA_MinersRule` [[▶ 168](#)] を使用して、カウントされたハーフサイクルに基づいて計算できます。損傷監視の場合、計数結果の平均応力補正は、ファンクションブロック `FB_CMA_MeanStressCorrection` [[▶ 165](#)] を使用して実行できます。

メモリ特性

サンプル数 N は、新しいデータセットが入力されるたびに自動的に増加し、現在の結果を計算します。つまり、ファンクションブロックがインスタンス化されてからの全ての入力値を使用します。サンプル数をゼロにリセットする（FBの内部メモリを削除する）には、`ResetData()` メソッドを使用します。

マルチチャンネル入力データ処理時の動作

複数のチャンネル（ $nChannels > 1$ ）を処理する場合、各チャンネルが異なる戻り値になる可能性があります。この場合、戻り値はファンクションブロックで個別に問い合わせできます。全てのチャンネルではなく、1つ以上のチャンネルの結果が許容されない場合、ファンクションブロックの値は `eCM_InfRTTime_AmbiguousChannelResults` に対応します。全てのチャンネルの結果が許容されない場合、ファンクションブロック上の値は `eCM_ErrRTTime_ErrornousChannelResults` に対応します。

全チャンネルの戻り値のリストは、`GetChannelErrors()` のメソッドで照会できます。

サンプルの実装

実装用のサンプルは以下のリンクから入手できます：[疲労分析](#) [[▶ 351](#)]

入力と出力

入力バッファと出力バッファは、以下の定義（入力/出力形状）のいずれかに該当します。変数パラメータは、ファンクションブロック入力stInitParsの一部です。

バージョン	入力バッファ（マルチアレイ入力ストリーム） 要素タイプ、次元数、次元サイズ	出力バッファ（マルチアレイ出力ストリーム） 要素タイプ、次元数、次元サイズ
標準バージョン (nChannels = 1)	LREAL, 1, 指定なし*	UDINT, 2, (nBins+2) x (nBinsMean+2)
マルチチャンネルバージョン (nChannels > 1)	LREAL, 2, nChannels x 指定なし*。	UDINT, 3, nChannels x (nBins+2) x (nBinsMean+2)

*:この次元の長さは任意に選択できるため、アプリケーションや先行のアルゴリズムの出力バッファに合わせるができます。

```
VAR_INPUT
    stInitPars      : ST_CM_RainflowCounting_InitPars; // init parameter
    nOwnID          : UDINT;                          // ID for this FB instance
    aDestIDs        : ARRAY[1..cCMA_MaxDest] OF UDINT; // IDs of destinations for output
    nResultBuffers  : UDINT := 4;                      // number of MultiArrays which should be in
    itialized for results (0 for no initialization)
    tTransferTimeout : LTIME := LTIME#500US;          // timeout checking off during access to in
    ter-task FIFOs
END_VAR
```

入力パラメータ

このファンクションブロックの入力パラメータは初期化パラメータを表し、FBインスタンスの宣言で割り当て済みである必要があります。（または、Init()メソッド）一度しか割り当てできません。実行中の変更はできません。

- stInitPars:ST_CM_RainflowCounting_InitPars [▶ 297] 型の初期化パラメータを持つファンクションブロック固有の構造体。パラメータは、上記の入出力バッファの定義と紐づいていなければなりません。
- nOwnID:一意のIDでファンクションブロックインスタンスを識別します。これは常に0より大きくなければなりません。この目的のためには、列挙を定義するアプローチが確立されています。
- aDestIDs:転送先のIDを指定することで、結果の転送先を定義します。出力バッファの定義（上記を参照）は、選択された各転送先の入力バッファの定義と紐づいていなければなりません。
- nResultBuffers:このファンクションブロックは、指定された数のマルチアレイバッファにより転送トレイストリームを初期化します。デフォルト値は4です。
- tTransferTimeout:内部マルチアレイ転送の同期タイムアウトの設定です。[並列処理](#) [▶ 72]の項を参照してください。

出力パラメータ

```
VAR_OUTPUT
    bError          : BOOL;                          // TRUE if an error occurs. Reset by next metho
    d call.
    hrErrorCode      : HRESULT;                      // '< 0' = error; '> 0' = info; '0' = no error/
    info
    ipErrorMessage  : I_TcMessage := fbErrorMessage; // Shows detailed information about occurred er
    rors, warnings and more.
    nCntResults      : ULINT;                        // Counts outgoing results (MultiArrays were ca
    lculated and sent to transfer tray).
END_VAR
```

- bError:エラーが発生した場合の出力は、TRUEです。
- hrErrorCode:エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧](#) [▶ 367]に記載されています。
- ipErrorMessage:現在の戻り値に関するより詳細な情報を含みます。[エラーの説明と情報](#) [▶ 81]の項を参照してください。この特別なインターフェイスポインタは、常に有効かつ割り当て済みであるよう、内部的に保護されています。

メソッド

```
METHOD Init : HRESULT
VAR_INPUT
    stInitPars      : ST_CM_RainflowCounting_InitPars; // init parameter
    nOwnID          : UDINT;                          // ID for this FB instance
    aDestIDs        : ARRAY[1..cCMA_MaxDest] OF UDINT; // IDs of destinations for output
    nResultBuffers  : UDINT := 4;                      // number of MultiArrays which should be initialized for results (0 for no initialization)
END_VAR
```

Call():

このメソッドは、現在の入力データにアルゴリズムを適用するために、サイクルごとに呼び出されます。メソッドが新しい結果もエラーも示さない場合、ファンクションブロックは入力データを待機します。これは、解析チェーンのプロセスにおける規則的な動作です。

- **戻り値** : エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD Call : HRESULT
VAR_OUTPUT
    bNewResult      : BOOL; // TRUE every time when outgoing MultiArray was calculated and sent to transfer tray.
    bError          : BOOL; // TRUE if an error occurs.
    hrErrorCode     : HRESULT; // '< 0' = error; '> 0' = info; '0' = no error/info
END_VAR
```

- **bError**: エラーが発生した場合の出力は、TRUE です。
- **hrErrorCode**: エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。この出力はメソッドの戻り値と同じです。



タイムアウトが発生した場合、または結果を格納するマルチアレイバッファがない場合、入力データも結果データも失われません。次の呼び出しに転送されます。

Init():

このメソッドは通常、状態監視アプリケーションでは必要ありません。これはファンクションブロック初期化の代替手段を提供するものです。Init() メソッドは、PLC の初期化フェーズ中にのみ呼び出すことができます。実行時には使用できません。メソッド FB_init または、属性 'call_after_init' の使用を参照してください (TwinCAT PLC リファレンスを参照)。さらに、これはファンクションブロックのカプセル化を容易にします。

Init() メソッドを使用して初期化を行う場合、ファンクションブロックインスタンスの入力パラメータを宣言で代入することはできません。

- **戻り値** : エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。
- **stInitPars**: [ST_CM_RainflowCounting_InitPars \[▶ 297\]](#) 型の初期化パラメータを持つファンクションブロック固有の構造体。パラメータは、上記の入出力バッファの定義と紐づいていなければなりません。
- **nOwnID**: 一意の ID でファンクションブロックインスタンスを識別します。これは常に 0 より大きくなければなりません。この目的のためには、列挙を定義するアプローチが確立されています。
- **aDestIDs**: 転送先の ID を指定することで、結果の転送先を定義します。出力バッファの定義 (上記を参照) は、選択された各転送先の入力バッファの定義と紐づいていなければなりません。
- **nResultBuffers**: このファンクションブロックは、指定された数のマルチアレイバッファにより転送トレイストリームを初期化します。デフォルト値は 4 です。

ResetData() :

このメソッドは、既に追加された全てのデータレコードを削除します。ファンクションブロックのメモリ特性を参照してください。ResetData() の後、Call() メソッドが再度呼び出された場合、有効な結果を計算するために内部メモリを補充する必要があります。

- **戻り値** : エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD ResetData : HRESULT
VAR_INPUT
END_VAR
```

PassInputs() :

API PLC リファレンス [▶ 79]で説明されているとおり、FB_CMA_Source インスタンスが呼び出され、信号データがターゲットブロックに転送される限り、解析チェーンの以降の全てのファンクションブロックを周期的に呼び出す必要があります。

アルゴリズムを一定時間実行しない方が有用な場合があります。例えば、いくつかのアルゴリズムは、事前に教示や設定をした後でのみ実行すべきです。ファンクションブロックは周期的に呼び出す必要がありますが、ファンクションブロックに到着したデータが通信リング [▶ 72]内で転送されれば十分です。これは、Call() メソッドの代わりにPassInputs() メソッドを使用して行われます。アルゴリズム自体はここでは呼び出されず、したがって結果も計算されず、出力バッファも生成されません。

- **戻り値** : エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD PassInputs : HRESULT
VAR_INPUT
END_VAR
```

GetChannelErrors() :

このメソッドは、複数のチャンネル（nChannels > 1）を処理する際に、チャンネル固有の戻り値のリストを照会できるようにします。呼び出しは、ファンクションブロックの戻り値が eCM_InfRTTime_AmbiguousChannelResults または eCM_ErrRTTime_ErrornousChannelResults のいずれかに該当する場合に有用です。

- **戻り値** : エラーコード一覧の読み込み処理に関する情報です。この値は、クエリが成功した場合は TRUE に、そうでない場合は FALSE に設定されます。

```
METHOD GetChannelErrors : BOOL
VAR_IN_OUT
    aChannelErrors : ARRAY[*] OF HRESULT;
END_VAR
```

- aChannelErrors: HRESULT 型の長さnChannelsのエラーリストです。

類似したファンクションブロック

ファンクションブロックFB_CMA_MeanStressCorrection [▶ 165] により、監視対象部品の引張強度に関する平均応力補正を実行できます。

ファンクションブロックFB_CMA_MinersRule [▶ 168] のアルゴリズムは、カウントされたハーフサイクルに基づいてコンポーネントの疲労損傷を計算するために使用できます。

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM, Tc3_CM_Base

5.1.31 FB_CMA_ReadCsvFile

CSVファイルを読み込み、PLCバッファ（配列）にデータを順次出力します。

データは、TwinCAT 3 Scopeで記録、表示、保存できます。保存形式の1つとしてCSVが考えられます。これにより、信号データをCSVファイルに保存し、後で FB_CMA_ReadCsvFile を使用して再度読み込むことができます。読み込まれた信号データは、FB_CMA_Source [▶ 230]のインスタンスに転送することができます。その結果、入力データの代替ソースとして状態監視解析チェーンに転送できます。

ファンクションブロックの使用例については、[疲労分析 \[▶ 351\]](#) をご覧ください。

本ページの一番下には別の例を記載しています。

入力と出力

入力パラメータ

```

VAR_INPUT
    bExecute      : BOOL := TRUE;           // starts execution (on rising edge)
    bAbort        : BOOL;                   // aborts execution (after finishing current ADS communica
tion)
    sNetId        : T_AmsNetId := '';       // TwinCAT system network address (could be kept empty for
local host)
    tAdsTimeout   : TIME := T#1S;          // ADS timeout (only 1 second as condition for a fast data
transmission)
    sFileName     : T_MaxString := '';     // CSV source file path and name (located on the system wi
th the specified Net Id)
    sSeparator    : STRING(2) := '$T';     // CSV field separator [e.g. tab '$T' or comma ',' or semi
colon ';' or colon ':' or blank ' ']
    nHeaderLines  : UDINT := 0;            // Number of lines in the CSV file which belongs to the he
ader. Data start after header lines.
    bColumn       : BOOL := TRUE;          // select whether the data is placed as column (e.g. TcSco
pe CSV file) or as line
END_VAR
VAR_IN_OUT
    aBuffer       : ARRAY[*] OF LREAL;     // buffer with individual length (do not switch the buffer
during one execution)
END_VAR

```

- **bExecute**: ファンクションブロックがまだ有効でない場合、bExecute入力の正エッジによってファンクションブロックが呼び出されます。
- **bAbort**: この入力設定されている場合、実行は中止されます。このファンクションブロックは、既に内部で開始されているADS通信を終了できるように、出力bBusy=FALSE に達するまで呼び出ししなければなりません。
- **sNetId**: ローカルデバイス上で操作を実行する場合は、この入力変数を指定する必要はありません。あるいは、空の文字列を指定することもできます。クエリを別のコンピュータに向けるには、そのコンピュータのAMS Net ID (T_AmsNetId 型) を指定します。
- **tAdsTimeout**: ADS内部通信の最大実行時間を指定します。
- **sFileName**: 読み込むCSVファイルのファイルパスを指定します。
- **sSeparator**: CSVファイルで使用するCSVフィールド区切り文字を指定します。こうして、個々の値は互いに分離されます。
- **nHeaderLines**: ヘッダ情報のみを含む行数を指定します。これらは読み出し時にスキップされます。TwinCAT 3 Scope Viewで保存したCSVファイルには、異なる数のヘッダ情報を含めることができます。
- **bColumn**: データがCSVの行 (bColumn=FALSE) と列 (bColumn=TRUE) のどちらに配置されているかを示します。後者は、TwinCAT 3 Scope Viewで保存されたCSVファイルの場合です。
- **aBuffer**: 読み込んだ値を書き込む出力データバッファを指定します。これは、任意の長さのLREAL型のPLC配列である必要があります。ハードウェアセンサ信号の代替としてFB_CMA_ReadCsvFileを使用する場合、長さは通常オーバーサンプリング係数に対応します。指定されたバッファは、操作の間、維持しなければなりません。動作中、aBuffer バッファは新しいデータレコードで複数回満たされ、これはbBufferFull 出力変数を通じて通知されます。

出力パラメータ

```

VAR_OUTPUT
    bBufferFull   : BOOL;                   // signals that aBuffer is fully filled
    nReadRecords  : UDINT;                 // shows the number of records saved to aBuffer
    bBusy         : BOOL;                 // TRUE if execution is active
    bError        : BOOL;                 // TRUE if an error occurred
    hrErrorCode    : HRESULT;             // '< 0' = error; '> 0' = info; '0' = no error/info
    ipErrorMessage : I_TcMessage := fbErrorMessage; // shows detailed information about occurred
errors, warnings and more
END_VAR

```

- **bBufferFull**: aBuffer バッファが完全に満たされた場合、出力はTRUE になります。これは、CSVファイルが完全に読み出されるまでの間に、しばしば発生します。aBuffer バッファに新しいデータレコードがある場合は、ファンクションブロックが再度呼び出される前に直ちに処理する必要があります。通常、ファンクションブロックが呼び出されるたびに、新しいデータがバッファに格納されます。とはいえ、bBufferFull の出力をチェックし、正しいデータストリームであることを確認する必要があります。

- nReadRecords: 出力は読み取りデータセット全体にいくつの値が含まれているかを示します。
- bBusy: ファンクションブロックが有効な間は、出力はTRUEです。CSVファイルが完全に読み込まれたか、エラーが発生したときの出力はFALSEです。
- bError: エラーが発生した場合の出力は、TRUEです。
- hrErrorCode: エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。
- ipErrorMessage: 現在の戻り値に関するより詳細な情報を含みます。[エラーの説明と情報 \[▶ 81\]](#)の項を参照してください。この特別なインターフェースポイントは、常に有効かつ割り当て済みであるよう、内部的に保護されています。

例

この例では、FB_CMA_ReadCsvFile の使用方法を示します。aBufferに指定されたバッファには、大きなバッファ長が想定されており、別のタスクへの委譲が推奨されます。バッファは、プロセスイメージ経由で実際のPLCタスクに転送され、オーバーサンプリングターミナルに類似したプロセスイメージ入力として使用できます。

サンプルのダウンロード : https://infosys.beckhoff.com/content/1033/TF3600_TC3_ConditionMonitoring/Resources/17670679435.zip

5.1.32 FB_CMA_RealFFT

実数値入力信号の高速フーリエ変換（FFT）を計算します。

ファンクションブロックFB_CMA_RealFFTは、実数値入力信号 $x[n]$ のフーリエ変換を計算します。高性能なFFTアルゴリズムが使用されています。順変換と逆変換を計算できます。設定は、入力stInitPars で設定できます（入力と出力を参照）。

DFT表記による順フーリエ変換の定義：

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-i2\pi nk/N}$$

DFT表記による逆フーリエ変換の定義：

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] e^{i2\pi nk/N}$$

入力信号成分の最高周波数は、エイリアシング効果を避けるために、入力信号のサンプリングレートの最大でも半分程度にするべきです。

FFTは、周期的に継続する信号の変換として定義されます。これは、周期的に継続する信号が正確に連続していない、つまり最初と最後が一致しない場合、即座にジャンプと判定される可能性があります。ファンクションブロックFB_CMA_PowerSpectrum [▶ 189]およびFB_CMA_MagnitudeSpectrum [▶ 161] は、ウィンドウ関数を乗算したオーバーラップセクションの解析により、これらの困難を回避します。

スケーリング

計測によるスペクトルの定量的な評価のために、計算されたスペクトルは、直流成分、すなわち出力の最初の配列要素については $1/nFFT\_Length$ 、他の全ての要素については $2/nFFT\_Length$ 、重み付けされます。ファンクションブロックは、順変換と逆変換の間でスケーリングを行い、連続した順変換と逆変換の場合に、元の入力信号が出力で再び直接得られるようにします。

NaN（非数）の発生

入力ベクトルに1つ以上のNaN（非数）値が含まれる場合、出力セクタ全体がNaNで満たされます。詳細については別項NaN値 [▶ 71]を参照してください。

● NaN値の取り扱い

NaN値につながる上記の状況を除外できない、または安全に無視できない場合、アプリケーションプログラムはこれらのエラー値を適切に処理できるようにしてください。

マルチチャンネル入力データ処理時の動作

複数のチャンネル ($nChannels > 1$) を処理する場合、各チャンネルが異なる戻り値になる可能性があります。この場合、戻り値はファンクションブロックで個別に問い合わせできます。全てのチャンネルではなく、1つ以上のチャンネルの結果が許容されない場合、ファンクションブロックの値は `eCM_InfRTTime_AmbiguousChannelResults` に対応します。全てのチャンネルの結果が許容されない場合、ファンクションブロック上の値は `eCM_ErrRTTime_ErrornousChannelResults` に対応します。

全チャンネルの戻り値のリストは、`GetChannelErrors()` のメソッドで照会できます。

サンプルの実装

実装用のサンプルは以下のリンクから入手できます：[実数値入力信号によるFFT \[▶ 307\]](#)。

入力と出力

入力バッファと出力バッファは、以下の定義（入力/出力形状）のいずれかに該当します。変数パラメータは、ファンクションブロック入力 `stInitPars` の一部です。

バージョン	入力バッファ（マルチアレイ入力ストリーム） 要素タイプ、次元数、次元サイズ	出力バッファ（マルチアレイ出力ストリーム） 要素タイプ、次元数、次元サイズ
標準バージョン ($nChannels = 1$)	LREAL, 1, <code>nFFT_Length</code>	LCOMPLEX, 1, <code>nFFT_Length/2+1</code>
フルスペクトル搭載バージョン* ($nChannels = 1$)	LREAL, 1, <code>nFFT_Length</code>	LCOMPLEX, 1, <code>nFFT_Length</code>
マルチチャンネルバージョン ($nChannels > 1$)	LREAL, 2, <code>nChannels x nFFT_Length</code>	LCOMPLEX, 2, <code>nChannels x nFFT_Length/2+1</code>
フルスペクトル搭載マルチチャンネル・バージョン* ($nChannels > 1$)	LREAL, 2, <code>nChannels x nFFT_Length</code>	LCOMPLEX, 2, <code>nChannels x nFFT_Length</code>

*:パラメータ `bHalfSpec` (`:=FALSE`) を反転させるとフルスペクトルの出力バッファを出力できます。

```
VAR_INPUT
    stInitPars      : ST_CM_RealFFT_InitPars;           // init parameter
    nOwnID          : UDINT;                           // ID for this FB instance
    aDestIDs        : ARRAY[1..cCMA_MaxDest] OF UDINT; // IDs of destinations for output
    nResultBuffers  : UDINT := 4;                      // number of MultiArrays which should be in
    itialized for results (0 for no initialization)
    tTransferTimeout : LTIME := LTIME#500US;           // timeout checking off during access to in
    ter-task FIFOs
END_VAR
```

入力パラメータ

このファンクションブロックの入力パラメータは初期化パラメータを表し、FBインスタンスの宣言で割り当て済みである必要があります。（または、`Init()`メソッド）一度しか割り当てできません。実行中の変更はできません。

- `stInitPars`: `ST_CM_RealFFT_InitPars` [▶ 297] 型の初期化パラメータを持つファンクションブロック構造体。パラメータは、上記の入出力バッファの定義と紐づいていなければなりません。
- `nOwnID`: 一意のIDでファンクションブロックインスタンスを識別します。これは常に0より大きくなければなりません。この目的のためには、列挙を定義するアプローチが確立されています。
- `aDestIDs`: 転送先のIDを指定することで、結果の転送先を定義します。出力バッファの定義（上記を参照）は、選択された各転送先の入力バッファの定義と紐づいていなければなりません。
- `nResultBuffers`: このファンクションブロックは、指定された数のマルチアレイバッファにより転送トレイストリームを初期化します。デフォルト値は4です。

- **tTransferTimeout**:内部マルチアレイ転送の同期タイムアウトの設定です。[並列処理 \[▶ 72\]](#)の項を参照してください。

出力パラメータ

```
VAR_OUTPUT
    bError      : BOOL;           // TRUE if an error occurs. Reset by next method call.
    hrErrorCode  : HRESULT;       // '< 0' = error; '> 0' = info; '0' = no error/info
    ipErrorMessage : I_TcMessage := fbErrorMessage; // Shows detailed information about occurred errors, warnings and more.
    nCntResults   : ULINT;        // Counts outgoing results (MultiArrays were calculated and sent to transfer tray).
END_VAR
```

- **bError**:エラーが発生した場合の出力は、TRUEです。
- **hrErrorCode**:エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。
- **ipErrorMessage**:現在の戻り値に関するより詳細な情報を含みます。[エラーの説明と情報 \[▶ 81\]](#)の項を参照してください。この特別なインターフェースポイントは、常に有効かつ割り当て済みであるよう、内部的に保護されています。

メソッド

```
METHOD Init : HRESULT
VAR_INPUT
    stInitPars      : ST_CM_RealFFT_InitPars;           // init parameter
    nOwnID           : UDINT;                           // ID for this FB instance
    aDestIDs         : ARRAY[1..cCMA_MaxDest] OF UDINT; // IDs of destinations for output
    nResultBuffers   : UDINT := 4;                     // number of MultiArrays which should be initialized for results (0 for no initialization)
END_VAR
```

Call():

このメソッドは、現在の入力データにアルゴリズムを適用するために、サイクルごとに呼び出されます。メソッドが新しい結果もエラーも示さない場合、ファンクションブロックは入力データを待機します。これは、解析チェーンのプロセスにおける規則的な動作です。

- **戻り値**: エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD Call : HRESULT
VAR_OUTPUT
    bNewResult      : BOOL;           // TRUE every time when outgoing MultiArray was calculated and sent to transfer tray.
    bError          : BOOL;           // TRUE if an error occurs.
    hrErrorCode      : HRESULT;       // '< 0' = error; '> 0' = info; '0' = no error/info
END_VAR
```

- **bError**:エラーが発生した場合の出力は、TRUEです。
- **hrErrorCode**:エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。この出力はメソッドの戻り値と同じです。



タイムアウトが発生した場合、または結果を格納するマルチアレイバッファがない場合、入力データも結果データも失われません。次の呼び出しに転送されます。

Init():

このメソッドは通常、状態監視アプリケーションでは必要ありません。これはファンクションブロック初期化の代替手段を提供するものです。Init() メソッドは、PLCの初期化フェーズ中にのみ呼び出すことができます。実行時には使用できません。メソッドFB_init または、属性'call_after_init'の使用を参照してください（TwinCAT PLCリファレンスを参照）。さらに、これはファンクションブロックのカプセル化を容易にします。

Init()メソッドを使用して初期化を行う場合、ファンクションブロックインスタンスの入力パラメータを宣言で代入することはできません。

- **戻り値**：エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。
- **stInitPars**: ST_CM_RealFFT_InitPars [▶ 297]型の初期化パラメータを持つファンクションブロック構造体。パラメータは、上記の入出力バッファの定義と紐づいていなければなりません。
- **nOwnID**: 一意のIDでファンクションブロックインスタンスを識別します。これは常に0より大きくなければなりません。この目的のためには、列挙を定義するアプローチが確立されています。
- **aDestIDs**: 転送先のIDを指定することで、結果の転送先を定義します。出力バッファの定義（上記を参照）は、選択された各転送先の入力バッファの定義と紐づいていなければなりません。
- **nResultBuffers**: このファンクションブロックは、指定された数のマルチアレイバッファにより転送ストリームを初期化します。デフォルト値は4です。

PassInputs() :

API PLC リファレンス [▶ 79]で説明されているとおり、FB_CMA_Source インスタンスが呼び出され、信号データがターゲットブロックに転送される限り、解析チェーンの以降の全てのファンクションブロックを周期的に呼び出す必要があります。

アルゴリズムを一定時間実行しない方が有用な場合があります。例えば、いくつかのアルゴリズムは、事前に教示や設定をした後でのみ実行すべきです。ファンクションブロックは周期的に呼び出す必要がありますが、ファンクションブロックに到着したデータが[通信リング \[▶ 72\]](#)内で転送されれば十分です。これは、Call() メソッドの代わりにPassInputs() メソッドを使用して行われます。アルゴリズム自体はここでは呼び出されず、したがって結果も計算されず、出力バッファも生成されません。

- **戻り値**：エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD PassInputs : HRESULT
VAR_INPUT
END_VAR
```

GetChannelErrors() :

このメソッドは、複数のチャンネル（nChannels > 1）を処理する際に、チャンネル固有の戻り値のリストを照会できるようにします。呼び出しは、ファンクションブロックの戻り値がeCM_InfRTime_AmbiguousChannelResults またはeCM_ErrRTime_ErrornousChannelResults のいずれかに該当する場合に有効です。

- **戻り値**: エラーコード一覧の読み込み処理に関する情報です。この値は、クエリが成功した場合はTRUE に、そうでない場合はFALSE に設定されます。

```
METHOD GetChannelErrors : BOOL
VAR_IN_OUT
aChannelErrors : ARRAY[*] OF HRESULT;
END_VAR
```

- **aChannelErrors**: HRESULT 型の長さnChannelsのエラーリストです。

類似したファンクションブロック

ファンクションブロックFB_CMA_ComplexFFT [▶ 104]は、複素数値の入力信号に対しフーリエ変換を計算します。

ファンクションブロックFB_CMA_PowerSpectrum [▶ 189]は、実数値入力信号のパワースペクトルを計算します。

ファンクションブロックFB_CMA_MagnitudeSpectrum [▶ 161]は、実数値入力信号のマグニチュードスペクトルを計算します。

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM, Tc3_CM_Base



CM 3.1 では一部の機能がすでに使用可能です。[互換性 \[▶ 61\]](#)の項を参照してください。

5.1.33 FB_CMA_RMS

シングルチャンネルおよびマルチチャンネルの実数値信号の時間的RMS値を計算します。

このファンクションブロックは、1つまたは複数の入力チャンネルの時間的RMS値を計算します。RMS値は内部バッファ長Mに対してブロック単位で計算されます。

$$x_{\text{RMS}} = \sqrt{\frac{1}{M} \sum_{m=0}^{M-1} x^2[m]}$$

この内部バッファが一杯になると、最も古い値が最新の値で置き換えられ、新しい結果が出力されます。置換される入力値の数は、ソースファンクションブロック ([FB_CMA_Source \[▶ 230\]](#)) で設定したマルチアレイサイズに依存します。

メモリ特性

RMS値の計算のために、nBufferLength の時系列の値がチャンネル/サブチャンネルごとに内部に保存されます。入力バッファサイズが小さい場合の呼び出しでは、転送できる値の数は少なくなります。この場合、バッファの内容はシフトされ、信号長は新たに転送された値の対応する数で満たされます。入力バッファが内部バッファより大きい場合、内部バッファは計算のために最新の値で満たされます。

NaN（非数）の発生

入力ベクトルに1つ以上のNaN（非数）値が含まれる場合、出力セクタ全体がNaNで満たされます。詳細については別項[NaN値 \[▶ 71\]](#)を参照してください。

内部バッファが新しい有効値で完全に満たされるまで、出力ベクトルはNaNで満たされます。



NaN値の取り扱い

NaN値につながる上記の状況を除外できない、または安全に無視できない場合、アプリケーションプログラムはこれらのエラー値を適切に処理できるようにしてください。

マルチチャンネル入力データ処理時の動作

複数のチャンネル（nChannels > 1）を処理する場合、各チャンネルが異なる戻り値になる可能性があります。この場合、戻り値はファンクションブロックで個別に問い合わせできます。全てのチャンネルではなく、1つ以上のチャンネルの結果が許容されない場合、ファンクションブロックの値はeCM_InfRTime_AmbiguousChannelResultsに対応します。全てのチャンネルの結果が許容されない場合、ファンクションブロック上の値はeCM_ErrRTime_ErrornousChannelResults に対応します。

全チャンネルの戻り値のリストは、GetChannelErrors()のメソッドで照会できます。

複数のサブチャンネル(nSubChannels > 0)を処理する場合、入出力データのフォーマットには特に注意が必要です。入力データが上流のファンクションブロックによるマルチチャンネルの結果で構成されている場合、nChannels の値を採用する必要があります。この場合、パラメータnSubChannels を使用して追加の設定を行います。

例：マルチチャンネルスペクトル（FB_CMA_MagnitudeSpectrumなど）の周波数チャンネルの統計的考察（FB_CMA_Quantilesなど）では、nChannels の値は、入力信号の数と同じである必要があります。サブチャンネルの数nSubChannels は、スペクトルの長さに対応します。

サンプルの実装

実装用のサンプルは以下のリンクから入手できます：[時間ベースのRMS値 \[▶ 321\]](#)。

入力と出力

入力バッファと出力バッファは、以下の定義（入力/出力形状）のいずれかに該当します。変数パラメータは、ファンクションブロック入力stInitParsの一部です。

バージョン	入力バッファ（マルチアレイ入力ストリーム） 要素タイプ、次元数、次元サイズ	出力バッファ（マルチアレイ出力ストリーム） 要素タイプ、次元数、次元サイズ
複数データセット用標準バージョン (nSubChannels = 0)	LREAL, 2, nChannelsx 指定なし*	LREAL, 1, nChannels
複数データセット用マルチチャネルバージョン (nSubChannels > 0)	LREAL, 3, nChannels x nSubChannels x 指定なし*	LREAL, 2, nChannels x nSubChannels

*:この次元の長さは任意に選択できるため、アプリケーションや先行のアルゴリズムの出力バッファに合わせる必要があります。

```
VAR_INPUT
    stInitPars      : ST_CM_RMS_InitPars;           // init parameter
    nOwnID          : UDINT;                       // ID for this FB instance
    aDestIDs        : ARRAY[1..cCMA_MaxDest] OF UDINT; // IDs of destinations for output
    nResultBuffers  : UDINT := 4;                 // number of MultiArrays which should be in
    itialized for results (0 for no initialization)
    tTransferTimeout : LTIME := LTIME#500US;       // timeout checking off during access to in
    ter-task FIFOs
END_VAR
```

入力パラメータ

このファンクションブロックの入力パラメータは初期化パラメータを表し、FBインスタンスの宣言で割り当て済みである必要があります。(または、Init()メソッド) 一度しか割り当てできません。実行中の変更はできません。

- **stInitPars:** ST_CM_RMS_InitPars [▶ 298]型の初期化パラメータを持つファンクションブロック固有の構造体。パラメータは、上記の入出力バッファの定義と紐づいていなければなりません。
- **nOwnID:**一意のIDでファンクションブロックインスタンスを識別します。これは常に0より大きくなければなりません。この目的のためには、列挙を定義するアプローチが確立されています。
- **aDestIDs:**転送先のIDを指定することで、結果の転送先を定義します。出力バッファの定義（上記を参照）は、選択された各転送先の入力バッファの定義と紐づいていなければなりません。
- **nResultBuffers:**このファンクションブロックは、指定された数のマルチアレイバッファにより転送トレイストリームを初期化します。デフォルト値は4です。
- **tTransferTimeout:**内部マルチアレイ転送の同期タイムアウトの設定です。[並列処理 \[▶ 72\]](#)の項を参照してください。

出力パラメータ

```
VAR_OUTPUT
    bError          : BOOL;                       // TRUE if an error occurs. Reset by next metho
    d call.
    hrErrorCode      : HRESULT;                   // '< 0' = error; '> 0' = info; '0' = no error/
    info
    ipErrorMessage  : I_TcMessage := fbErrorMessage; // Shows detailed information about occurred er
    rors, warnings and more.
    nCntResults      : ULINT;                     // Counts outgoing results (MultiArrays were ca
    lculated and sent to transfer tray).
END_VAR
```

- **bError:**エラーが発生した場合の出力は、TRUEです。
- **hrErrorCode:**エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。
- **ipErrorMessage:**現在の戻り値に関するより詳細な情報を含みます。[エラーの説明と情報 \[▶ 81\]](#)の項を参照してください。この特別なインターフェイスポインタは、常に有効かつ割り当て済みであるよう、内部的に保護されています。

メソッド

```
METHOD Init : HRESULT
VAR_INPUT
    stInitPars      : ST_CM_RMS_InitPars; // init parameter
    nOwnID          : UDINT;              // ID for this FB instance
    aDestIDs        : ARRAY[1..CMA_MaxDest] OF UDINT; // IDs of destinations for output
    nResultBuffers  : UDINT := 4; // number of MultiArrays which should be initialized for results (
0 for no initialization)
END_VAR
```

Call():

このメソッドは、現在の入力データにアルゴリズムを適用するために、サイクルごとに呼び出されます。メソッドが新しい結果もエラーも示さない場合、ファンクションブロックは入力データを待機します。これは、解析チェーンのプロセスにおける規則的な動作です。

- **戻り値** : エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD Call : HRESULT
VAR_OUTPUT
    bNewResult      : BOOL; // TRUE every time when outgoing MultiArray was calculated and sent to transfer tray.
    bError          : BOOL; // TRUE if an error occurs.
    hrErrorCode      : HRESULT; // '< 0' = error; '> 0' = info; '0' = no error/info
END_VAR
```

- **bError**: エラーが発生した場合の出力は、TRUE です。
- **hrErrorCode**: エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。この出力はメソッドの戻り値と同じです。



タイムアウトが発生した場合、または結果を格納するマルチアレイバッファがない場合、入力データも結果データも失われません。次の呼び出しに転送されます。

Init():

このメソッドは通常、状態監視アプリケーションでは必要ありません。これはファンクションブロック初期化の代替手段を提供するものです。Init() メソッドは、PLCの初期化フェーズ中にのみ呼び出すことができます。実行時には使用できません。メソッドFB_init または、属性'call_after_init'の使用を参照してください (TwinCAT PLCリファレンスを参照)。さらに、これはファンクションブロックのカプセル化を容易にします。

Init()メソッドを使用して初期化を行う場合、ファンクションブロックインスタンスの入力パラメータを宣言で代入することはできません。

- **戻り値** : エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。
- **stInitPars**: [ST_CM_RMS_InitPars \[▶ 298\]](#)型の初期化パラメータを持つファンクションブロック固有の構造体。パラメータは、上記の入出力バッファの定義と紐づいていなければなりません。
- **nOwnID**: 一意のIDでファンクションブロックインスタンスを識別します。これは常に0より大きくなければなりません。この目的のためには、列挙を定義するアプローチが確立されています。
- **aDestIDs**: 転送先のIDを指定することで、結果の転送先を定義します。出力バッファの定義（上記を参照）は、選択された各転送先の入力バッファの定義と紐づいていなければなりません。
- **nResultBuffers**: このファンクションブロックは、指定された数のマルチアレイバッファにより転送トレイストリームを初期化します。デフォルト値は4です。

ResetData() :

このメソッドは、既に追加された全てのデータレコードを削除します。ファンクションブロックのメモリ特性を参照してください。ResetData() の後、Call() メソッドが再度呼び出された場合、有効な結果を計算するために内部メモリを補充する必要があります。

- **戻り値** : エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD ResetData : HRESULT
VAR_INPUT
END_VAR
```

PassInputs() :

API PLC リファレンス [▶ 79]で説明されているとおり、FB_CMA_Source インスタンスが呼び出され、信号データがターゲットブロックに転送される限り、解析チェーンの以降の全てのファンクションブロックを周期的に呼び出す必要があります。

アルゴリズムを一定時間実行しない方が有用な場合があります。例えば、いくつかのアルゴリズムは、事前に教示や設定をした後でのみ実行すべきです。ファンクションブロックは周期的に呼び出す必要がありますが、ファンクションブロックに到着したデータが通信リング [▶ 72]内で転送されれば十分です。これは、Call() メソッドの代わりにPassInputs() メソッドを使用して行われます。アルゴリズム自体はここでは呼び出されず、したがって結果も計算されず、出力バッファも生成されません。

- **戻り値** : エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD PassInputs : HRESULT
VAR_INPUT
END_VAR
```

GetChannelErrors() :

このメソッドは、複数のチャンネル（nChannels > 1）を処理する際に、チャンネル固有の戻り値のリストを照会できるようにします。呼び出しは、ファンクションブロックの戻り値が eCM_InfRTTime_AmbiguousChannelResults または eCM_ErrRTTime_ErrornousChannelResults のいずれかに該当する場合に有用です。

- **戻り値**: エラーコード一覧の読み込み処理に関する情報です。この値は、クエリが成功した場合は TRUE に、そうでない場合は FALSE に設定されます。

```
METHOD GetChannelErrors : BOOL
VAR_IN_OUT
    aChannelErrors : ARRAY[*] OF HRESULT;
END_VAR
```

- aChannelErrors: HRESULT 型の長さnChannelsのエラーリストです。

類似したファンクションブロック

ファンクションブロックFB_CMA_CrestFactor [▶ 93]は、各チャンネルの時系列データに対して、シングルチャンネルおよびマルチチャンネルのクレストファクタを計算します。

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM, Tc3_CM_Base



CM 3.1 では一部の機能がすでに使用可能です。[互換性 \[▶ 61\]](#)の項を参照してください。

5.1.34 FB_CMA_SlidingDFT

実数値入力信号のスペクトル値をオンライン計算します。

ファンクションブロックFB_CMA_SlidingDFTは、設定可能な数の個々のスペクトル値を提供し、ファンクションブロックFB_CMA_RealFFT [▶ 203]に適合しています。DFT係数は、Goertzelアルゴリズムを使用して

$$X_n(k) = \sum_{m=0}^{N-1} x(q+m)e^{-i2\pi km/N}, \quad q = n - N + 1$$

減衰パラメータ r ($0 < r < 1$) を用いて、再帰規則によって近似的に算出されます。

$$\tilde{X}_n(k) = \sum_{m=0}^{N-1} x(q+m)r^{N-m}e^{-i2\pi km/N} = re^{i2\pi k/N} \left(\tilde{X}_{n-1}(k) - r^N x(n-N) + x(n) \right)$$

計算にはスライディングウィンドウが使用され、新しいサンプルごとにスペクトル値が更新されます。FFTと比較した場合、選択したスペクトル値のオンライン計算のメリットは、プロセッサの使用率が一定であること（ブロック処理が不要）と、ウィンドウの長さが2の累乗に制約されないことです。入力信号成分の最高周波数は、エイリアシング効果を避けるために、入力信号のサンプリングレートの最大でも半分程度にするべきです。

メモリ特性

スペクトル値の計算に直近の `nWindowLength(N)` サンプルを使用するため、時系列にジャンプが生じると、周波数解析においてスペクトル領域で誤った結果を引き起こす可能性があります。入力バッファサイズが小さい場合の呼び出しでは、転送できる値の数は少なくなります。この場合、設定したスペクトル値は、新たに転送された全ての値に対して更新されます。

NaN（非数）の発生

入力ベクトルに1つ以上のNaN（非数）値が含まれる場合、出力セクタ全体がNaNで満たされます。詳細については別項NaN値 [▶ 71]を参照してください。

● NaN値の取り扱い

1 NaN値につながる上記の状況を除外できない、または安全に無視できない場合、アプリケーションプログラムはこれらのエラー値を適切に処理できるようにしてください。

マルチチャンネル入力データ処理時の動作

複数のチャンネル (`nChannels > 1`) を処理する場合、各チャンネルが異なる戻り値になる可能性があります。この場合、戻り値はファンクションブロックで個別に問い合わせできます。全てのチャンネルではなく、1つ以上のチャンネルの結果が許容されない場合、ファンクションブロックの値は `eCM_InfRTime_AmbiguousChannelResults` に対応します。全てのチャンネルの結果が許容されない場合、ファンクションブロック上の値は `eCM_ErrRTime_ErrornousChannelResults` に対応します。

全チャンネルの戻り値のリストは、`GetChannelErrors()`のメソッドで照会できます。

サンプルの実装

ファンクションブロックの使用方法和設定を示す実装サンプルは、以下のリンクから入手できます：[スライディングDFT \[▶ 359\]](#)

入力と出力

入力バッファと出力バッファは、以下の定義（入力/出力形状）のいずれかに該当します。変数パラメータは、ファンクションブロック入力 `stInitPars` の一部です。

バージョン	入力バッファ（マルチアレイ 入力ストリーム） 要素タイプ、次元数、次元サイズ	出力バッファ（マルチアレイ 出力ストリーム） 要素タイプ、次元数、次元サイズ
標準バージョン (<code>nChannels = 1</code>)	LREAL, 1, 指定なし*	LCOMPLEX, 1, <code>nBins</code>
マルチチャンネルバージョン (<code>nChannels > 1</code>)	LREAL, 2, <code>nChannels</code> x 指定なし*	LCOMPLEX, 2, <code>nChannels</code> x <code>nBins</code>

```
VAR_INPUT
    stInitPars      : ST_CM_SlidingDFT_InitPars;           // init parameter
    nOwnID          : UDINT;                               // ID for this FB instance
    aDestIDs        : ARRAY[1..cCMA_MaxDest] OF UDINT;    // IDs of destinations for output
    nResultBuffers  : UDINT := 4;                          // number of MultiArrays which should be in
    ialized for results (0 for no initialization)
    tTransferTimeout : LTIME := LTIME#500US;              // timeout checking off during access to i
    nter-task FIFOs
END_VAR
```

入力パラメータ

このファンクションブロックの入力パラメータは初期化パラメータを表し、FBインスタンスの宣言で割り当て済みである必要があります。(または、Init()メソッド) 一度しか割り当てできません。実行中の変更はできません。

- **stInitPars:ST CM SlidingDFT_InitPars [▶ 298]** 型の初期化パラメータを持つファンクションブロック固有の構造体。パラメータは、上記の入出力バッファの定義と紐づいていなければなりません。
- **nOwnID:**一意のIDでファンクションブロックインスタンスを識別します。これは常に0より大きくなければなりません。この目的のためには、列挙を定義するアプローチが確立されています。
- **aDestIDs:**転送先のIDを指定することで、結果の転送先を定義します。出力バッファの定義（上記を参照）は、選択された各転送先の入力バッファの定義と紐づいていなければなりません。
- **nResultBuffers:**このファンクションブロックは、指定された数のマルチアレイバッファにより転送トレイストリームを初期化します。デフォルト値は4です。
- **tTransferTimeout:**内部マルチアレイ転送の同期タイムアウトの設定です。[並列処理 \[▶ 72\]](#)の項を参照してください。

出力パラメータ

```
VAR_OUTPUT
    bError      : BOOL;                // TRUE if an error occurs. Reset by next method call.
    hrErrorCode  : HRESULT;            // '< 0' = error; '> 0' = info; '0' = no error/info
    ipErrorMessage : I_TcMessage := fbErrorMessage; // Shows detailed information about occurred errors, warnings and more.
    nCntResults   : ULINT;             // Counts outgoing results (MultiArrays were calculated and sent to transfer tray).
END_VAR
```

- **bError:**エラーが発生した場合の出力は、TRUEです。
- **hrErrorCode:**エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。
- **ipErrorMessage:**現在の戻り値に関するより詳細な情報を含みます。[エラーの説明と情報 \[▶ 81\]](#)の項を参照してください。この特別なインターフェースポインタは、常に有効かつ割り当て済みであるよう、内部的に保護されています。

メソッド

Call():

このメソッドは、現在の入力データにアルゴリズムを適用するために、サイクルごと呼び出されます。メソッドが新しい結果もエラーも示さない場合、ファンクションブロックは入力データを待機します。これは、解析チェーンのプロセスにおける規則的な動作です。

- **戻り値:** エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD Call : HRESULT
VAR_OUTPUT
    bNewResult   : BOOL;                // TRUE every time when outgoing MultiArray was calculated and sent to transfer tray.
    bError       : BOOL;                // TRUE if an error occurs.
    hrErrorCode   : HRESULT;            // '< 0' = error; '> 0' = info; '0' = no error/info
END_VAR
```

- **bError:**エラーが発生した場合の出力は、TRUEです。
- **hrErrorCode:**エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。この出力はメソッドの戻り値と同じです。



タイムアウトが発生した場合、または結果を格納するマルチアレイバッファがない場合、入力データも結果データも失われません。次の呼び出しに転送されます。

Configure() :

このメソッドを呼び出す際には、最初にスペクトル値を設定しておく必要があります。DFTインデックスは、周波数 f から $k = f * fSampleRate / nWindowLength$ に対して計算されます。 f が周波数分解能 $fSampleRate / nWindowLength$ の整数倍でない場合、スペクトル部分は連続する2つのスペクトル値に分配されます。

対応するPLC配列は以下のように定義してください。Configure()メソッドは、異なる引数セットを持つ新しい設定にも使用できます。

- **戻り値**：エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD Configure : HRESULT
VAR_INPUT
    pArg      : PVOID;      // pointer to array of arguments
    nArgSize   : UDINT;      // size of arguments buffer in bytes
END_VAR
```

入力バッファは、以下の定義（入力形状）のいずれかに紐づいています。変数パラメータは、ファンクションブロック入力stInitPars の一部です。

バージョン	入力バッファ（マルチアレイ 入力ストリーム） 要素タイプ、次元数、次元サイズ
全てのチャンネルで同一設定	UDINT, 1, nBins
チャンネル別の設定 (nChannels > 1)	UDINT, 2, nChannels x nBins

```
METHOD Init : HRESULT
VAR_INPUT
    stInitPars      : ST_CM_SlidingDFT_InitPars;      // init parameter
    nOwnID           : UDINT;                          // ID for this FB instance
    aDestIDs         : ARRAY[1..cCMA_MaxDest] OF UDINT; // IDs of destinations for output
    nResultBuffers   : UDINT := 4;                     // number of MultiArrays which should be initialized for results (0 for no initialization)
END_VAR
```

Init():

このメソッドは通常、状態監視アプリケーションでは必要ありません。これはファンクションブロック初期化の代替手段を提供するものです。Init() メソッドは、PLCの初期化フェーズ中にのみ呼び出すことができます。実行時には使用できません。メソッドFB_init または、属性'call_after_init'の使用を参照してください（TwinCAT PLCリファレンスを参照）。さらに、これはファンクションブロックのカプセル化を容易にします。

Init()メソッドを使用して初期化を行う場合、ファンクションブロックインスタンスの入力パラメータを宣言で代入することはできません。

- **戻り値**：エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。
- **stInitPars**:ST_CM_SlidingDFT_InitPars [▶ 298] 型の初期化パラメータを持つファンクションブロック固有の構造体。パラメータは、上記の入出力バッファの定義と紐づいていなければなりません。
- **nOwnID**:一意のIDでファンクションブロックインスタンスを識別します。これは常に0より大きくなければなりません。この目的のためには、列挙を定義するアプローチが確立されています。
- **aDestIDs**:転送先のIDを指定することで、結果の転送先を定義します。出力バッファの定義（上記を参照）は、選択された各転送先の入力バッファの定義と紐づいていなければなりません。
- **nResultBuffers**:このファンクションブロックは、指定された数のマルチアレイバッファにより転送トレイストリームを初期化します。デフォルト値は4です。

ResetData() :

このメソッドは、既に追加された全てのデータレコードを削除します。ファンクションブロックのメモリ特性を参照してください。ResetData() の後、Call() メソッドが再度呼び出された場合、有効な結果を計算するために内部メモリを補充する必要があります。

- **戻り値**：エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD ResetData : HRESULT
VAR_INPUT
END_VAR
```

PassInputs() :

API PLC リファレンス [▶ 79]で説明されているとおり、FB_CMA_Source インスタンスが呼び出され、信号データがターゲットブロックに転送される限り、解析チェーンの以降の全てのファンクションブロックを周期的に呼び出す必要があります。

アルゴリズムを一定時間実行しない方が有用な場合があります。例えば、いくつかのアルゴリズムは、事前に教示や設定をした後でのみ実行すべきです。ファンクションブロックは周期的に呼び出す必要がありますが、ファンクションブロックに到着したデータが通信リング [▶ 72]内で転送されれば十分です。これは、Call() メソッドの代わりにPassInputs() メソッドを使用して行われます。アルゴリズム自体はここでは呼び出されず、したがって結果も計算されず、出力バッファも生成されません。

- **戻り値:** エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD PassInputs : HRESULT
VAR_INPUT
END_VAR
```

GetChannelErrors() :

このメソッドは、複数のチャンネル（nChannels > 1）を処理する際に、チャンネル固有の戻り値のリストを照会できるようにします。呼び出しは、ファンクションブロックの戻り値が eCM_InfRTTime_AmbiguousChannelResults または eCM_ErrRTTime_ErrornousChannelResults のいずれかに該当する場合に有用です。

- **戻り値:** エラーコード一覧の読み込み処理に関する情報です。この値は、クエリが成功した場合は TRUE に、そうでない場合は FALSE に設定されます。

```
METHOD GetChannelErrors : BOOL
VAR_IN_OUT
    aChannelErrors : ARRAY[*] OF HRESULT;
END_VAR
```

- aChannelErrors: HRESULT 型の長さnChannelsのエラーリストです。

類似したファンクションブロック

ファンクションブロックFB_CMA_RealFFT [▶ 203]は、実数値入力信号に対して高速フーリエ変換を計算します。

ファンクションブロックST_CM_SparseSpectrum_InitPars [▶ 299] は、個別に設定可能なマグニチュードとスペクトル値を計算します。

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM, Tc3_CM_Base

5.1.35 FB_CMA_SparseSpectrum

実数値入力信号に対する個々のスペクトル値を計算します。

ファンクションブロックFB_CMA_SparseSpectrum は、設定した数の各スペクトル値を提供し、ファンクションブロック、FB_CMA_RealFFT [▶ 203]、FB_CMA_MagnitudeSpectrum [▶ 161] または FB_CMA_PowerSpectrum [▶ 189] にしたがってスケーリングされます。DFT係数は、Goertzelアルゴリズムを使用して

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-i2\pi nk/N}$$

計算します。これは、ウィンドウ長 L で計算される係数の数 M に対して、次の条件が当てはまる場合、Radix-2 FFT (FFTの手法の一つ) よりも効率的です：

$$M \leq \frac{5N}{6L} \log_2(N), \quad N = 2^m$$

ここで、 N は、ウィンドウ長 L 以上の最小の2の累乗です。必要なスペクトル値が少数または個別の場合、必要に応じて（高速）サンプリングタスク内で直接、ファンクションブロックを用いて計算できるため、スペクトルの急な変化に素早く対応できます。

入力信号成分の最高周波数は、エイリアシング効果を避けるために、入力信号のサンプリングレートの最大でも半分程度にするべきです。

ファンクションブロックは複数の機能を実行します。[データストリームの解析](#) [▶ 16] と [周波数解析](#) [▶ 34] を参照してください。

入力データバッファは、まず直前のバッファとオーバーラップした後、ウィンドウ関数で乗算されます。その後、DFT係数はGoertzelアルゴリズムを用いて計算されます。このパラメータeSpectrumTypeは、得られた複素数値から絶対値を計算するか、その2乗を計算するかを指定します。パラメータbTransformToDecibelがTRUEの場合、マグニチュード値またはパワー値が計算されていれば、値はデシベル値に変換されます。これらのデシベル値は、両方のスペクトル値に対して同じです。つまり、パワー値における2乗の影響は、マグニチュード値に対して2倍することでデシベル値の計算に反映されています。さらに、パラメータeScalingTypeを使用することで、結果のスケールリングが可能です。[スペクトルのスケールリング](#) [▶ 25] を参照してください。

スケールリング

加速度スペクトル密度などの結果値のスケールリングは、デフォルトでFFTの定義と一致しています。これは、ウィンドウ長とウィンドウ関数の影響が排除されることを意味します。[スペクトルスケールリングオプション](#) [▶ 371] の項で説明されている表形式のパラメータは、絶対計測の算術スケールリングに使用できます。

メモリ特性

ウェルチ法を使用しているため、現在の入力データバッファと最後に転送されたバッファの両方が計算に使用されます。組み込まれるバッファの数は、選択したオーバーラップ(nOverlap)に依存します。周波数解析では、時系列データの段階的な変化が考慮されます。したがって、正しい結果を得るためには、使用する入力データバッファは、間隔やジャンプなしに連結していなければなりません。

NaN（非数）の発生

入力ベクトルに1つ以上のNaN（非数）値が含まれる場合、出力セクタ全体がNaNで満たされます。詳細については別項[NaN値](#) [▶ 71]を参照してください。

● NaN値の取り扱い

i NaN値につながる上記の状況を除外できない、または安全に無視できない場合、アプリケーションプログラムはこれらのエラー値を適切に処理できるようにしてください。

マルチチャンネル入力データ処理時の動作

複数のチャンネル（nChannels > 1）を処理する場合、各チャンネルが異なる戻り値になる可能性があります。この場合、戻り値はファンクションブロックで個別に問い合わせできます。全てのチャンネルではなく、1つ以上のチャンネルの結果が許容されない場合、ファンクションブロックの値はeCM_InfRTime_AmbiguousChannelResultsに対応します。全てのチャンネルの結果が許容されない場合、ファンクションブロック上の値はeCM_ErrRTime_ErrornousChannelResultsに対応します。

全チャンネルの戻り値のリストは、GetChannelErrors()のメソッドで照会できます。

サンプルの実装

ファンクションブロックの使用方法和設定を示す実装サンプルは、以下のリンクから入手できます：[個々のスペクトル値の計算](#) [▶ 315]

入力と出力

入力バッファと出力バッファは、以下の定義（入力/出力形状）のいずれかに該当します。変数パラメータは、ファンクションブロック入力stInitParsの一部です。

バージョン	入力バッファ（マルチアレイ入力ストリーム） 要素タイプ、次元数、次元サイズ	出力バッファ（マルチアレイ出力ストリーム） 要素タイプ、次元数、次元サイズ
標準バージョン DFT (nChannels = 1, eSpectrumType = eCM_DFT)	LREAL, 1, nWindowLength-nOverlap	LCOMPLEX, 1, nBins
標準バージョン スペクトル (nChannels = 1, eSpectrumType <> eCM_DFT)	LREAL, 1, nWindowLength-nOverlap	LREAL, 1, nBins
マルチチャンネルバージョン DFT (nChannels > 1, eSpectrumType = eCM_DFT)	LREAL, 2, nChannels x nWindowLength-nOverlap	LCOMPLEX, 2, nChannels x nBins
マルチチャンネルバージョン スペクトル (nChannels > 1, eSpectrumType <> eCM_DFT)	LREAL, 2, nChannels x nWindowLength-nOverlap	LREAL, 2, nChannels x nBins

```
VAR_INPUT
    stInitPars      : ST_CM_SparseSpectrum_InitPars;      // init parameter
    nOwnID          : UDINT;                               // ID for this FB instance
    aDestIDs        : ARRAY[1..cCMA_MaxDest] OF UDINT;    // IDs of destinations for output
    nResultBuffers  : UDINT := 4;                         // number of MultiArrays which should be initialized for results (0 for no initialization)
    tTransferTimeout : LTIME := LTIME#500US;              // timeout checking off during access to inter-task FIFOs
END_VAR
```

入力パラメータ

このファンクションブロックの入力パラメータは初期化パラメータを表し、FBインスタンスの宣言で割り当て済みである必要があります。（または、Init()メソッド）一度しか割り当てできません。実行中の変更はできません。

- stInitPars:ST_CM_SparseSpectrum_InitPars [▶ 299] 型の初期化パラメータを持つファンクションブロック固有の構造体。パラメータは、上記の入出力バッファの定義と紐づいていなければなりません。
- nOwnID:一意のIDでファンクションブロックインスタンスを識別します。これは常に0より大きくなければなりません。この目的のためには、列挙を定義するアプローチが確立されています。
- aDestIDs:転送先のIDを指定することで、結果の転送先を定義します。出力バッファの定義（上記を参照）は、選択された各転送先の入力バッファの定義と紐づいていなければなりません。
- nResultBuffers:このファンクションブロックは、指定された数のマルチアレイバッファにより転送トレイストリームを初期化します。デフォルト値は4です。
- tTransferTimeout:内部マルチアレイ転送の同期タイムアウトの設定です。[並列処理 \[▶ 72\]](#)の項を参照してください。

出力パラメータ

```
VAR_OUTPUT
    bError          : BOOL;                               // TRUE if an error occurs. Reset by next method call.
    hrErrorCode      : HRESULT;                           // '< 0' = error; '> 0' = info; '0' = no error/info
    ipErrorMessage  : I_TcMessage := fbErrorMessage;    // Shows detailed information about occurred errors, warnings and more.
    nCntResults      : ULINT;                             // Counts outgoing results (MultiArrays were calculated and sent to transfer tray).
END_VAR
```

- bError:エラーが発生した場合の出力は、TRUEです。

- **hrErrorCode**:エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。
- **ipErrorMessage**:現在の戻り値に関するより詳細な情報を含みます。[エラーの説明と情報 \[▶ 81\]](#)の項を参照してください。この特別なインターフェースポインタは、常に有効かつ割り当て済みであるよう、内部的に保護されています。

メソッド

Call():

このメソッドは、現在の入力データにアルゴリズムを適用するために、サイクルごとに呼び出されます。メソッドが新しい結果もエラーも示さない場合、ファンクションブロックは入力データを待機します。これは、解析チェーンのプロセスにおける規則的な動作です。

- **戻り値**:エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD Call : HRESULT
VAR_OUTPUT
    bNewResult    : BOOL;          // TRUE every time when outgoing MultiArray was calculated and sent to transfer tray.
    bError        : BOOL;          // TRUE if an error occurs.
    hrErrorCode    : HRESULT;      // '< 0' = error; '> 0' = info; '0' = no error/info
END_VAR
```

- **bError**:エラーが発生した場合の出力は、TRUEです。
- **hrErrorCode**:エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。この出力はメソッドの戻り値と同じです。



タイムアウトが発生した場合、または結果を格納するマルチアレイバッファがない場合、入力データも結果データも失われません。次の呼び出しに転送されます。

Configure() :

このメソッドを呼び出す際には、最初にスペクトル値を設定しておく必要があります。DFTインデックスは、周波数 f から $k = f * fSampleRate / nWindowLength$ に対して計算されます。 f が周波数分解能 $fSampleRate / nWindowLength$ の整数倍でない場合、スペクトル部分は連続する2つのスペクトル値に分配されます。

対応するPLC配列は以下のように定義してください。Configure()メソッドは、異なる引数セットを持つ新しい設定にも使用できます。

- **戻り値**:エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD Configure : HRESULT
VAR_INPUT
    pArg      : PVOID;          // pointer to array of arguments
    nArgSize  : UDINT;          // size of arguments buffer in bytes
END_VAR
```

入力バッファは、以下の定義（入力形状）のいずれかに紐づいています。変数パラメータは、ファンクションブロック入力stInitPars の一部です。

バージョン	入力バッファ（マルチアレイ入カストリーム） 要素タイプ、次元数、次元サイズ
全てのチャンネルで同一設定	UDINT, 1, nBins
チャンネル別の設定 (nChannels > 1)	UDINT, 2, nChannels x nBins

```
METHOD Init : HRESULT
VAR_INPUT
    stInitPars : ST_CM_SparseSpectrum_InitPars; // init parameter
    nOwnID     : UDINT;                          // ID for this FB instance
    aDestIDs   : ARRAY[1..cCMA_MaxDest] OF UDINT; // IDs of destinations for output
```

```
nResultBuffers : UDINT := 4; // number of MultiArrays which should be initialized for results (0 for no initialization)
END_VAR
```

Init():

このメソッドは通常、状態監視アプリケーションでは必要ありません。これはファンクションブロック初期化の代替手段を提供するものです。Init() メソッドは、PLCの初期化フェーズ中にのみ呼び出すことができます。実行時には使用できません。メソッドFB_init または、属性'call_after_init'の使用を参照してください（TwinCAT PLCリファレンスを参照）。さらに、これはファンクションブロックのカプセル化を容易にします。

Init()メソッドを使用して初期化を行う場合、ファンクションブロックインスタンスの入力パラメータを宣言で代入することはできません。

- **戻り値**：エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。
- **stInitPars:ST_CM_SparseSpectrum_InitPars [▶ 299]** 型の初期化パラメータを持つファンクションブロック固有の構造体。パラメータは、上記の入出力バッファの定義と紐づいていなければなりません。
- **nOwnID**:一意のIDでファンクションブロックインスタンスを識別します。これは常に0より大きくなければなりません。この目的のためには、列挙を定義するアプローチが確立されています。
- **aDestIDs**:転送先のIDを指定することで、結果の転送先を定義します。出力バッファの定義（上記を参照）は、選択された各転送先の入力バッファの定義と紐づいていなければなりません。
- **nResultBuffers**:このファンクションブロックは、指定された数のマルチアレイバッファにより転送ストリームを初期化します。デフォルト値は4です。

ResetData() :

このメソッドは、既に追加された全てのデータレコードを削除します。ファンクションブロックのメモリ特性を参照してください。ResetData() の後、Call() メソッドが再度呼び出された場合、有効な結果を計算するために内部メモリを補充する必要があります。

- **戻り値**：エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD ResetData : HRESULT
VAR_INPUT
END_VAR
```

PassInputs() :

API PLC リファレンス [▶ 79]で説明されているとおり、FB_CMA_Source インスタンスが呼び出され、信号データがターゲットブロックに転送される限り、解析チェーンの以降の全てのファンクションブロックを周期的に呼び出す必要があります。

アルゴリズムを一定時間実行しない方が有用な場合があります。例えば、いくつかのアルゴリズムは、事前に教示や設定をした後でのみ実行すべきです。ファンクションブロックは周期的に呼び出す必要がありますが、ファンクションブロックに到着したデータが[通信リング \[▶ 72\]](#)内で転送されれば十分です。これは、Call() メソッドの代わりにPassInputs() メソッドを使用して行われます。アルゴリズム自体はここでは呼び出されず、したがって結果も計算されず、出力バッファも生成されません。

- **戻り値**：エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD PassInputs : HRESULT
VAR_INPUT
END_VAR
```

GetChannelErrors() :

このメソッドは、複数のチャンネル（nChannels > 1）を処理する際に、チャンネル固有の戻り値のリストを照会できるようにします。呼び出しは、ファンクションブロックの戻り値がeCM_InfRTTime_AmbiguousChannelResults またはeCM_ErrRTTime_ErrornousChannelResults のいずれかに該当する場合に有効です。

- **戻り値**: エラーコード一覧の読み込み処理に関する情報です。この値は、クエリが成功した場合はTRUE に、そうでない場合はFALSE に設定されます。

```

METHOD GetChannelErrors : BOOL
VAR_IN_OUT
    aChannelErrors : ARRAY[*] OF HRESULT;
END_VAR

```

- aChannelErrors:HRESULT 型の長さnChannelsのエラーリストです。

類似したファンクションブロック

ファンクションブロックFB_CMA_RealFFT [[▶ 203](#)]は、実数値入力信号に対して高速フーリエ変換を計算します。

ファンクションブロックFB_CMA_MagnitudeSpectrum [[▶ 161](#)] は、実数値入力信号のマグニチュードスペクトルを計算します。

ファンクションブロックFB_CMA_PowerSpectrum [[▶ 189](#)]は、最後のステップで得られた値を2乗することでパワースペクトルを計算します。

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM, Tc3_CM_Base

5.1.36 FB_CMA_SpikeEnergySpectrum

高周波信号成分のピークエネルギーを解析します。

ファンクションブロックFB_CMA_SpikeEnergySpectrumは、機械要素の振動解析に広く使用されている手法を実装しています。このメソッドは高周波信号成分、特に加速度センサの線形伝送範囲を超える周波数を解析するため、損傷パターンの早期検出において非常に感度が高い指標となります。

この手法は加速度センサ信号の評価に基づいており、加速度センサの共振周波数を越えた信号成分が考慮されます。したがって、高周波のパルス状の外乱は、例えばローラーベアリングの軽微な損傷を通過する際に発生し、搬送波信号の変調として現れます。キャリア信号は、取り付けられた加速度センサの共振周波数によって決定されます。スパイクエネルギー法は、搬送波信号をフィルタリングし、高周波のパルス状外乱の上に包絡線を形成します。そのため、例えばローラーベアリングのような部品の損傷周波数を、スペクトル内でより明確に認識できます。

この方法は、組み立て方法や、場所、センサの種類に強く依存するため、トレンド監視にのみ適しており、他の計測点への閾値の転送は不可能です。

ファンクションブロックFB_CMA_SpikeEnergySpectrumの入力データバッファは、まず直前のバッファとオーバーラップして結合され、ウィンドウ関数で乗算されます。その後、DFT係数はGoertzelアルゴリズムを用いて計算されます。このパラメータeSpectrumTypeは、得られた複素数値から絶対値を計算するか、その2乗を計算するかを指定します。パラメータbTransformToDecibel がTRUE の場合、値はデシベル値に変換されます。これらのデシベル値は、両方のスペクトル値（マグニチュードまたはその振幅）に対して同じです。つまり、パワー値における二乗の影響は、マグニチュード値に対して 2 倍することでデシベル値の計算に反映されています。さらに、パラメータeScalingTypeを使用することで、結果のスケールリングが可能です。[スペクトルのスケールリング](#) [[▶ 25](#)] を参照してください。

スケールリング

加速度スペクトル密度などの結果値のスケールリングは、デフォルトでFFTの定義と一致しています。これは、ウィンドウ長とウィンドウ関数の影響が排除されることを意味します。[スペクトルスケールリングオプション](#) [[▶ 371](#)] の項で説明されている表形式のパラメータは、絶対計測の算術スケールリングに使用できます。

メモリ特性

ウェルチ法を使用しているため、現在の入力データバッファと最後に転送されたバッファの両方が計算に使用されます。組み込まれるバッファの数は、選択したオーバーラップ(nOverlap)に依存します。周波数解析では、時系列データの段階的な変化が考慮されます。したがって、正しい結果を得るためには、使用する入力データバッファは、間隔やジャンプなしに連結していなければなりません。

NaN（非数）の発生

入力ベクトルに1つ以上のNaN（非数）値が含まれる場合、出力セクタ全体がNaNで満たされます。詳細については別項NaN値 [▶ 71]を参照してください。

● NaN値の取り扱い

i NaN値につながる上記の状況を除外できない、または安全に無視できない場合、アプリケーションプログラムはこれらのエラー値を適切に処理できるようにしてください。

マルチチャンネル入力データ処理時の動作

複数のチャンネル（ $nChannels > 1$ ）を処理する場合、各チャンネルが異なる戻り値になる可能性があります。この場合、戻り値はファンクションブロックで個別に問い合わせできます。全てのチャンネルではなく、1つ以上のチャンネルの結果が許容されない場合、ファンクションブロックの値はeCM_InfRTime_AmbiguousChannelResultsに対応します。全てのチャンネルの結果が許容されない場合、ファンクションブロック上の値はeCM_ErrRTime_ErrornousChannelResults に対応します。

全チャンネルの戻り値のリストは、GetChannelErrors()のメソッドで照会できます。

サンプルの実装

ファンクションブロックの使用方法和設定を示す実装サンプルは、以下のリンクから入手できます：[スパイクエネルギースペクトル \[▶ 354\]](#)

入力と出力

入力バッファと出力バッファは、以下の定義（入力/出力形状）のいずれかに該当します。変数パラメータは、ファンクションブロック入力stInitPars の一部です。

バージョン	入力バッファ（マルチアレイ 入力ストリーム） 要素タイプ、次元数、次元サイズ	出力バッファ（マルチアレイ 出力ストリーム） 要素タイプ、次元数、次元サイズ
標準バージョン ($nChannels = 1$)	LREAL, 1, nWindowLength-nOverlap	LREAL, 1, nFFT_Length/2+1
マルチチャンネルバージョン ($nChannels > 1$)	LREAL, 2, $nChannels \times nWindowLength - nOverlap$	LREAL, 2, $nChannels \times nFFT\_Length/2+1$

```
VAR_INPUT
    stInitPars      : ST_CM_SpikeEnergySpectrum_InitPars; // init parameter
    nOwnID          : UDINT;                             // ID for this FB instance
    aDestIDs        : ARRAY[1..cCMA_MaxDest] OF UDINT;    // IDs of destinations for output
    nResultBuffers  : UDINT := 4;                         // number of MultiArrays which should be
    initialized for results (0 for no initialization)
    tTransferTimeout : LTIME := LTIME#500US;             // timeout checking off during access to
    inter-task FIFOs
END_VAR
```

入力パラメータ

このファンクションブロックの入力パラメータは初期化パラメータを表し、FBインスタンスの宣言で割り当て済みである必要があります。（または、Init()メソッド）一度しか割り当てできません。実行中の変更はできません。

- stInitPars:ST_CM_SpikeEnergySpectrum_InitPars [▶ 300] 型の初期化パラメータを持つファンクションブロック固有の構造体。パラメータは、上記の入出力バッファの定義と紐づいていなければなりません。
- nOwnID:一意のIDでファンクションブロックインスタンスを識別します。これは常に0より大きくなければなりません。この目的のためには、列挙を定義するアプローチが確立されています。
- aDestIDs:転送先のIDを指定することで、結果の転送先を定義します。出力バッファの定義（上記を参照）は、選択された各転送先の入力バッファの定義と紐づいていなければなりません。
- nResultBuffers:このファンクションブロックは、指定された数のマルチアレイバッファにより転送トレイストリームを初期化します。デフォルト値は4です。

- **tTransferTimeout**: 内部マルチアレイ転送の同期タイムアウトの設定です。並列処理 [▶ 72] の項を参照してください。

出力パラメータ

```
VAR_OUTPUT
    bError      : BOOL;           // TRUE if an error occurs. Reset by next method call.
    hrErrorCode  : HRESULT;       // '< 0' = error; '> 0' = info; '0' = no error/info
    ipErrorMessage : I_TcMessage := fbErrorMessage; // Shows detailed information about occurred errors, warnings and more.
    nCntResults   : ULINT;        // Counts outgoing results (MultiArrays were calculated and sent to transfer tray).
END_VAR
```

- **bError**: エラーが発生した場合の出力は、TRUE です。
- **hrErrorCode**: エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#) に記載されています。
- **ipErrorMessage**: 現在の戻り値に関するより詳細な情報を含みます。[エラーの説明と情報 \[▶ 81\]](#) の項を参照してください。この特別なインターフェースポインタは、常に有効かつ割り当て済みであるよう、内部的に保護されています。

メソッド

Call():

このメソッドは、現在の入力データにアルゴリズムを適用するために、サイクルごとに呼び出されます。メソッドが新しい結果もエラーも示さない場合、ファンクションブロックは入力データを待機します。これは、解析チェーンのプロセスにおける規則的な動作です。

- **戻り値**: エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#) に記載されています。

```
METHOD Call : HRESULT
VAR_OUTPUT
    bNewResult   : BOOL;           // TRUE every time when outgoing MultiArray was calculated and sent to transfer tray.
    bError       : BOOL;           // TRUE if an error occurs.
    hrErrorCode   : HRESULT;       // '< 0' = error; '> 0' = info; '0' = no error/info
END_VAR
```

- **bError**: エラーが発生した場合の出力は、TRUE です。
- **hrErrorCode**: エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#) に記載されています。この出力はメソッドの戻り値と同じです。



タイムアウトが発生した場合、または結果を格納するマルチアレイバッファがない場合、入力データも結果データも失われません。次の呼び出しに転送されます。

Configure() :

このメソッドを呼び出すことで、減衰時間(fDecayTime) と、考慮する周波数帯域 (fLowerFrequencyLimit, fUpperFrequencyLimit) 経由) を実行時に調整できます。減衰時間は、ピークエネルギーが完全に減衰するように、最適なケースで選択する必要があります (fDecayTime > 1/f_fault など)。

対応する PLC 配列は以下のように定義してください :

[fDecayTime, fLowerFrequencyLimit, fUpperFrequencyLimit] Configure() メソッドは、異なる引数セットを持つ新しい設定にも使用できます。

- **戻り値**: エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は [エラーコードの概要 \[▶ 367\]](#) に記載されています。

```
METHOD Configure : HRESULT
VAR_INPUT
    pArg      : PVOID;           // pointer to array of arguments
    nArgSize  : UDINT;           // size of arguments buffer in bytes
END_VAR
```


入力バッファは、以下の定義（入力形状）のいずれかに紐づいています。変数パラメータは、ファンクションブロック入力stInitPars の一部です。

バージョン	入力バッファ（マルチアレイ 入力ストリーム） 要素タイプ、次元数、次元サイズ
全てのチャンネルで同一設定	LREAL, 1, 3
チャンネル別の設定 (nChannels > 1)	LREAL, 2, nChannels x 3

```

METHOD Init : HRESULT
VAR_INPUT
    stInitPars      : ST_CM_SpikeEnergySpectrum_InitPars; // init parameter
    nOwnID          : UDINT; // ID for this FB instance
    aDestIDs        : ARRAY[1..CMA_MaxDest] OF UDINT; // IDs of destinations for output
    nResultBuffers  : UDINT := 4; // number of MultiArrays which should be i
initialized for results (0 for no initialization)
END_VAR

```

Init():

このメソッドは通常、状態監視アプリケーションでは必要ありません。これはファンクションブロック初期化の代替手段を提供するものです。Init() メソッドは、PLCの初期化フェーズ中にのみ呼び出すことができます。実行時には使用できません。メソッドFB_init または、属性'call_after_init' の使用を参照してください（TwinCAT PLCリファレンスを参照）。さらに、これはファンクションブロックのカプセル化を容易にします。

Init()メソッドを使用して初期化を行う場合、ファンクションブロックインスタンスの入力パラメータを宣言で代入することはできません。

- **戻り値**：エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。
- **stInitPars:ST_CM_SpikeEnergySpectrum_InitPars [▶ 300]** 型の初期化パラメータを持つファンクションブロック固有の構造体。パラメータは、上記の入出力バッファの定義と紐づいていなければなりません。
- **nOwnID**:一意のIDでファンクションブロックインスタンスを識別します。これは常に0より大きくなければなりません。この目的のためには、列挙を定義するアプローチが確立されています。
- **aDestIDs**:転送先のIDを指定することで、結果の転送先を定義します。出力バッファの定義（上記を参照）は、選択された各転送先の入力バッファの定義と紐づいていなければなりません。
- **nResultBuffers**:このファンクションブロックは、指定された数のマルチアレイバッファにより転送トレイストリームを初期化します。デフォルト値は4です。

ResetData() :

このメソッドは、既に追加された全てのデータレコードを削除します。ファンクションブロックのメモリ特性を参照してください。ResetData() の後、Call() メソッドが再度呼び出された場合、有効な結果を計算するために内部メモリを補充する必要があります。

- **戻り値**：エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```

METHOD ResetData : HRESULT
VAR_INPUT
END_VAR

```

PassInputs() :

API PLC リファレンス [▶ 79]で説明されているとおり、FB_CMA_Source インスタンスが呼び出され、信号データがターゲットブロックに転送される限り、解析チェーンの以降の全てのファンクションブロックを周期的に呼び出す必要があります。

アルゴリズムを一定時間実行しない方が有用な場合があります。例えば、いくつかのアルゴリズムは、事前に教示や設定をした後でのみ実行すべきです。ファンクションブロックは周期的に呼び出す必要がありますが、ファンクションブロックに到着したデータが[通信リング \[▶ 72\]](#)内で転送されれば十分です。これは、Call() メソッドの代わりにPassInputs() メソッドを使用して行われます。アルゴリズム自体はここでは呼び出されず、したがって結果も計算されず、出力バッファも生成されません。

- **戻り値**：エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD PassInputs : HRESULT
VAR_INPUT
END_VAR
```

GetChannelErrors() :

このメソッドは、複数のチャンネル（nChannels > 1）を処理する際に、チャンネル固有の戻り値のリストを照会できるようにします。呼び出しは、ファンクションブロックの戻り値が eCM_InfRTTime_AmbiguousChannelResults または eCM_ErrRTTime_ErrornousChannelResults のいずれかに該当する場合に有効です。

- **戻り値**：エラーコード一覧の読み込み処理に関する情報です。この値は、クエリが成功した場合は TRUE に、そうでない場合は FALSE に設定されます。

```
METHOD GetChannelErrors : BOOL
VAR_IN_OUT
aChannelErrors : ARRAY[*] OF HRESULT;
END_VAR
```

- aChannelErrors: HRESULT 型の長さ nChannels のエラーリストです。

類似したファンクションブロック

ファンクションブロック [FB_CMA_PowerCepstrum \[▶ 185\]](#) は、実数値入力信号のパワーケプストラムを計算します。

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM, Tc3_CM_Base

5.1.37 FB_CMA_Sink

このファンクションブロックは、マルチアレイバッファからPLCデータバッファにデータを書き込みます。

指定したAnalysis IDに転送される全てのマルチアレイが含まれます。解析チェーンによっては、出力結果にNaN値が含まれることがあります。

注記

例外

NaN（非数）との比較は、実行停止につながる例外を引き起こす可能性があり、機械損傷の原因となる可能性があります。NaNの処理を行う前に、早急に結果を確認することを強く推奨します。あるいは、アプリケーションでNaNを処理する場合は、このタスクにおいて浮動小数点例外を無効にする必要があります。

入力と出力

```
VAR_INPUT
nOwnID : UDINT; // ID for this FB instance
tTransferTimeout : LTIME := LTIME#40US; // timeout checking off during access to inter-
task FIFOs
END_VAR
```

入力パラメータ

このファンクションブロックの入力パラメータは初期化パラメータを表し、FBインスタンスの宣言で割り当て済みである必要があります。（または、Init()メソッド）一度しか割り当てできません。実行中の変更はできません。

- nOwnID: 一意のIDでファンクションブロックインスタンスを識別します。これは常に0より大きくなければなりません。この目的のためには、列挙を定義するアプローチが確立されています。

- **tTransferTimeout**:内部マルチアレイ転送の同期タイムアウトの設定です。[並列処理 \[▶ 72\]](#)の項を参照してください。

出力パラメータ

```
VAR_OUTPUT
    bError      : BOOL;                // TRUE if an error occurs. Reset by next method call.
    hrErrorCode  : HRESULT;            // '< 0' = error; '> 0' = info; '0' = no error/info
    ipErrorMessage : I_TcMessage := fbErrorMessage; // Shows detailed information about occurred errors, warnings and more.
    nCntResults   : ULINT;             // Counts outgoing results (MultiArrays were calculated and sent to transfer tray).
END_VAR
```

- **bError**:エラーが発生した場合の出力は、TRUEです。
- **hrErrorCode**:エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。
- **ipErrorMessage**:現在の戻り値に関するより詳細な情報を含みます。[エラーの説明と情報 \[▶ 81\]](#)の項を参照してください。この特別なインターフェースポインタは、常に有効かつ割り当て済みであるよう、内部的に保護されています。

プロパティ

以下のプロパティは、最後に出力された結果に関連する解析チェーンの入力値についての情報を提供します。[FB_CMA_Source \[▶ 230\]](#) で可能なパラメータ設定も参照してください。

名前	タイプ	アクセス	説明
nSourceTimestampNewest	ULINT [T_DCTIME64; 1ns steps since 1.1.2000]	Get	最後の出力結果に関連して、解析チェーンの最も新しい入力値のタイムスタンプを出力。 これは時間範囲の上限（除外）となる。
nSourceTimestampOldest	ULINT [T_DCTIME64; 1ns steps since 1.1.2000]	Get	最後の出力結果に関連して、解析チェーンの最も古い入力値のタイムスタンプを出力。 これは時間範囲の下限（包含）となる。
nSourceValues	ULINT	Get	最後の出力結果に関連して、解析チェーンの入力値（信号値）の数を示す。
tSourceTimespan	LTIME	Get	最後の出力結果に関連して、解析チェーンの入力値（信号値）の時間範囲を出力。 tSourceTimespan := nSourceValues * tSamplePeriod;

メソッド

Output1D():

マルチアレイから外部の1次元データバッファにデータを書き込みます。メソッドが新しい結果もエラーも示さない場合、ファンクションブロックは入力データを待機します。これは、解析チェーンのプロセスにおける規則的な動作です。

- **戻り値** : エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧](#)に記載されています。

```

METHOD Output1D : HRESULT
VAR_INPUT
    pDataOut      : POINTER TO BYTE;      // address of data buffer
    nDataOutSize  : UDINT;                // size of data buffer in bytes
    eElementType  : E_MA_ElementTypeCode;
    nWorkDim      : UDINT:=0;             // It designates the dimension in the MultiArray being pro
cessed.
    nElements     : UDINT:=0;             // optional: default:0->full
    copy; It designates the number of elements to be copied out of the MultiArray.
    pStartIndex   : POINTER TO UDINT;     (* optional: default:0->internally handled as [0,0,...];
                                           It designates the index of the first element to be copied
out of the MultiArray.
                                           If allocated it must point to a onedimensional array of UD
INT with so many elements as dimensions of the MultiArray. *)
    nOptionPars   : DWORD;               // option mask
END_VAR
VAR_OUTPUT
    bNewResult    : BOOL;                // TRUE every time when data was written from MultiArray t
o data buffer.
    bError        : BOOL;                // TRUE if an error occurs.
    hrErrorCode    : HRESULT;            // '< 0' = error; '> 0' = info; '0' = no error/info
END_VAR

```

- **eElementType**:この入力はE_MA_ElementTypeCode型です。指定したマルチアレイバッファの要素型は、指定した外部データバッファの要素型と一致する必要があります。
- **nWorkDim**:処理するマルチアレイの次元。これらのデータは、指定された外部データバッファにコピーされます。一般的に、マルチアレイも1次元で、nWorkDim:=0 です。しかし、マルチアレイは追加で次元を持つこともできます。
- **nElements**:マルチアレイからコピーする要素数を指定します。全てをコピーするには、nElements:=0 を設定しなければなりません。しかし、結果のうち特定の帯域幅のみ必要な場合は、データ全体をコピーする必要はありません。これにより、外部データバッファの必要サイズも削減します。
- **pStartIndex**:これは、マルチアレイが複数の次元を持つ場合や、全ての要素をコピーしない場合に役立つオプションのパラメータです。マルチアレイからコピーする最初の要素のインデックスを指定します。代入する場合は、マルチアレイの次元数と同じ数の要素を持つUDINT の1次元配列を指す必要があります。
- **bNewResult**:この出力は、新しい結果がデータバッファに正常に書き込まれるたびにTRUEになります。
- **bError**:エラーが発生した場合、この出力はTRUEになります。
- **hrErrorCode**:エラーが発生した場合は、HRESULT のエラーコードが表示されます。可能性のある値は、エラーコード一覧に記載されています。この出力はメソッドの戻り値と同じです。
ヒント：タイムアウトが発生しても、入力データは失われません。次の呼び出しで処理されます。

Output1DStd() :

マルチアレイから外部の1次元データバッファにデータを書き込みます。メソッドが新しい結果もエラーも示さない場合、ファンクションブロックは入力データを待機します。これは、解析チェーンのプロセスにおける規則的な動作です。

このメソッドでは、パラメータnWorkDim:=0、nElements:=0、pStartIndex:=0 にデフォルト値を使用します。

- **戻り値**：エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、エラーコード一覧に記載されています。

```

METHOD Output1DStd : HRESULT
VAR_INPUT
    pDataOut      : POINTER TO BYTE;      // address of data buffer
    nDataOutSize  : UDINT;                // size of data buffer in bytes
    eElementType  : E_MA_ElementTypeCode;
    nOptionPars   : DWORD;               // option mask
END_VAR
VAR_OUTPUT
    bNewResult    : BOOL;                // TRUE every time when data was written from MultiArray t
o data buffer.
    bError        : BOOL;                // TRUE if an error occurs.
    hrErrorCode    : HRESULT;            // '< 0' = error; '> 0' = info; '0' = no error/info
END_VAR

```

- **eElementType**:この入力はE_MA_ElementTypeCode型です。指定したマルチアレイバッファの要素型は、指定した外部データバッファの要素型と一致する必要があります。

- **bNewResult**: この出力は、新しい結果がデータバッファに正常に書き込まれるたびにTRUEになります。
- **bError**: エラーが発生した場合、この出力はTRUEになります。
- **hrErrorCode**: エラーが発生した場合は、HRESULT のエラーコードが表示されます。可能性のある値は、エラーコード一覧に記載されています。この出力はメソッドの戻り値と同じです。
ヒント: タイムアウトが発生しても、入力データは失われません。次の呼び出しで処理されます。

Output2D():

マルチアレイから外部の2次元データバッファにデータを書き込みます。メソッドが新しい結果もエラーも示さない場合、オブジェクトは入力データを待機します。これは解析チェーンにおける標準的な動作です。

- **戻り値**: エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧](#) [▶ 367]に記載されています。

```
METHOD Output2D : HRESULT
VAR_INPUT
    pDataOut      : POINTER TO BYTE; // address of data buffer
    nDataOutSize  : UDINT;           // size of data buffer in bytes
    eElementType  : E_MA_ElementTypeCode;
    nWorkDim0     : UDINT:=0;        // It designates the first dimension in the MultiArray being p
    rocessed.
    nWorkDim1     : UDINT:=1;        // It designates the second dimension in the MultiArray being
    processed.
    nElementsDim0 : UDINT:=0;        // optional: default:0-
    >full copy; It designates the number of elements to be copied out of WorkDim0 of the MultiArray.
    nElementsDim1 : UDINT:=0;        // optional: default:0-
    >full copy; It designates the number of elements to be copied out of WorkDim1 of the MultiArray.
    pStartIndex   : POINTER TO UDINT; (* optional: default:0->internally handled as [0,0,...];
    It designates the index of the first element to be copied out
    of the MultiArray.
    If allocated it must point to a onedimensional array of UDINT
    with so many elements as dimensions of the MultiArray. *)
    nOptionPars   : DWORD;           // option mask
END_VAR
VAR_OUTPUT
    bNewResult    : BOOL;            // TRUE every time when data was written from MultiArray to dat
    a buffer.
    bError        : BOOL;            // TRUE if an error occurs.
    hrErrorCode    : HRESULT;        // '< 0' = error; '> 0' = info; '0' = no error/info
END_VAR
```

- **pDataOut**: 外部2次元データバッファのアドレス。
- **eElementType**: この入力には [E_MA_ElementTypeCode](#) [▶ 274]型です。指定したマルチアレイバッファの要素型は、指定した外部データバッファの要素型と一致する必要があります。
- **nWorkDim0**: 処理するマルチアレイの最初の次元を指定します。これらのデータは、指定された外部データバッファの最初の次元にコピーされます。一般的に、マルチアレイも2次元で、nWorkDim0:=0です。しかし、マルチアレイは追加で次元を持つこともできます。ただし、その場合、追加された次元はコピーされない可能性があります。
例: 2番目の次元を対象のデータバッファの最初のインデックスにコピーする場合、nWorkDim0:=1を設定します。
- **nWorkDim1**: 処理対象のマルチアレイの2次元目を指定します。これらのデータは、指定した外部データバッファの2次元目にコピーされます。一般的に、マルチアレイも2次元で、nWorkDim1:=1です。マルチアレイは追加で次元を持つこともできます。ただし、その場合、追加された次元はコピーされない可能性があります。
例: 最初の次元を対象のデータバッファの2番目のインデックスにコピーする場合、nWorkDim1:=0を設定します。
- **nElementsDim0**: マルチアレイのnWorkDim0から、コピーする要素数を指定します。全てをコピーするには、nElementsDim0:=0を設定しなければなりません。しかし、結果のうち特定の帯域幅のみ必要な場合は、データ全体をコピーする必要はありません。これにより、外部データバッファの必要サイズも削減します。
- **nElementsDim1**: マルチアレイのnWorkDim0から、コピーする要素数を指定します。全てをコピーするには、nElementsDim1:=0を設定しなければなりません。しかし、結果のうち特定の帯域幅のみ必要な場合は、データ全体をコピーする必要はありません。これにより、外部データバッファの必要サイズも削減します。

- **pStartIndex:**これは、マルチアレイが2次元以上ある場合や、全ての要素をコピーするわけではない場合に役立つオプションのパラメータです。マルチアレイからコピーする最初の要素のインデックスを指定します。代入する場合は、マルチアレイの次元数と同じ数の要素を持つUDINT の1次元配列を指す必要があります。
- **bNewResult:**この出力は、新しい結果がデータバッファに正常に書き込まれるたびにTRUEになります。
- **bError:**エラーが発生した場合、この出力はTRUEになります。
- **hrErrorCode:**エラーが発生した場合は、HRESULT のエラーコードが表示されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。この出力はメソッドの戻り値と同じです。
ヒント：タイムアウトが発生しても、入力データは失われません。次の呼び出しで処理されます。

Output2DStd() :

マルチアレイから外部の2次元データバッファにデータを書き込みます。メソッドが新しい結果もエラーも示さない場合、オブジェクトは入力データを待機します。これは解析チェーンにおける標準的な動作です。

このメソッドでは、パラメータnWorkDim0:=0、nWorkDim1:=1、nElementsDim0:=0、nElementsDim1:=0、pStartIndex:=0 にデフォルト値を使用します。

- **戻り値：**エラーが発生した場合は、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD Output2DStd : HRESULT
VAR_INPUT
    pDataOut      : POINTER TO BYTE; // address of data buffer
    nDataOutSize  : UDINT;           // size of data buffer in bytes
    eElementType  : E_MA_ElementTypeCode;
    nOptionMask   : DWORD;           // option mask
END_VAR
VAR_OUTPUT
    bNewResult    : BOOL;            // TRUE every time when data was written from MultiArray to data buffer.
    bError        : BOOL;            // TRUE if an error occurs.
    hrErrorCode    : HRESULT;        // '< 0' = error; '> 0' = info; '0' = no error/info
END_VAR
```

- **pDataOut:**外部2次元データバッファのアドレス。
- **eElementType:**この入力値はE_MA_ElementTypeCode [▶ 274]型です。指定したマルチアレイバッファの要素型は、指定した外部データバッファの要素型と一致する必要があります。
- **bNewResult:**この出力は、新しい結果がデータバッファに正常に書き込まれるたびにTRUEになります。
- **bError:**エラーが発生した場合、この出力はTRUEになります。
- **hrErrorCode:**エラーが発生した場合は、HRESULT のエラーコードが表示されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。この出力はメソッドの戻り値と同じです。
ヒント：タイムアウトが発生しても、入力データは失われません。次の呼び出しで処理されます。

Output3D():

マルチアレイから外部の3次元データバッファにデータを書き込みます。メソッドが新しい結果もエラーも示さない場合、オブジェクトは入力データを待機します。これは解析チェーンにおける標準的な動作です。

- **戻り値：**エラーが発生した場合は、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD Output3D : HRESULT
VAR_INPUT
    pDataOut      : POINTER TO BYTE; // address of data buffer
    nDataOutSize  : UDINT;           // size of data buffer in bytes
    eElementType  : E_MA_ElementTypeCode;
    nWorkDim0     : UDINT:=0;        // It designates the first dimension in the MultiArray being processed.
    nWorkDim1     : UDINT:=1;        // It designates the second dimension in the MultiArray being processed.
    nWorkDim2     : UDINT:=1;        // It designates the third dimension in the MultiArray being processed.
    nElementsDim0 : UDINT:=0;        // optional: default:0->full copy; It designates the number of elements to be copied out of WorkDim0 of the MultiArray.
    nElementsDim1 : UDINT:=0;        // optional: default:0->full copy; It designates the number of elements to be copied out of WorkDim1 of the MultiArray.
```

```

nElementsDim2 : UDINT:=0;          // optional: default:0-
>full copy; It designates the number of elements to be copied out of WorkDim2 of the MultiArray.
pStartIndex    : POINTER TO UDINT; (* optional: default:0->internally handled as [0,0,...];
                                   It designates the index of the first element to be copied out
of the MultiArray.
                                   If allocated it must point to a onedimensional array of UDINT
with so many elements as dimensions of the MultiArray. *)
nOptionParas   : DWORD;            // option mask
END_VAR
VAR_OUTPUT
bNewResult      : BOOL;             // TRUE every time when data was written from MultiArray to dat
a buffer.
bError          : BOOL;             // TRUE if an error occurs.
hrErrorcode     : HRESULT;          // '< 0' = error; '> 0' = info; '0' = no error/info
END_VAR

```

- **pDataOut:**外部2次元データバッファのアドレス。
- **eElementType:**この入力にはE_MA_ElementTypeCode [▶ 274]型です。指定したマルチアレイバッファの要素型は、指定した外部データバッファの要素型と一致する必要があります。
- **nWorkDim0:**処理するマルチアレイの最初の次元を指定します。これらのデータは、指定された外部データバッファの最初の次元にコピーされます。一般的に、マルチアレイも2次元で、nWorkDim0:=0です。しかし、マルチアレイは追加で次元を持つこともできます。ただし、その場合、追加された次元はコピーされない可能性があります。
例：2番目の次元を対象のデータバッファの最初のインデックスにコピーする場合、nWorkDim0:=1を設定します。
- **nWorkDim1:**処理対象のマルチアレイの2次元目を指定します。これらのデータは、指定した外部データバッファの2次元目にコピーされます。一般的に、マルチアレイも2次元で、nWorkDim1:=1です。マルチアレイは追加で次元を持つこともできます。ただし、その場合、追加された次元はコピーされない可能性があります。
例：最初の次元を対象のデータバッファの2番目のインデックスにコピーする場合、nWorkDim1:=0を設定します。
- **nWorkDim2:**処理対象のマルチアレイの3次元目を指定します。これらのデータは、指定した外部データバッファの3次元目にコピーされます。一般的に、マルチアレイも3次元で、nWorkDim2:=2です。しかし、マルチアレイは追加で次元を持つこともできます。ただし、その場合、追加された次元はコピーされない可能性があります。
例：最初の次元を対象のデータバッファの2番目のインデックスにコピーする場合、nWorkDim2:=0を設定します。
- **nElementsDim0:**マルチアレイのnWorkDim0から、コピーする要素数を指定します。全てをコピーするには、nElementsDim0:=0を設定しなければなりません。しかし、結果のうち特定の帯域幅のみ必要な場合は、データ全体をコピーする必要はありません。これにより、外部データバッファの必要サイズも削減します。
- **nElementsDim1:**マルチアレイのnWorkDim1から、コピーする要素数を指定します。全てをコピーするには、nElementsDim1:=0を設定しなければなりません。しかし、結果のうち特定の帯域幅のみ必要な場合は、データ全体をコピーする必要はありません。これにより、外部データバッファの必要サイズも削減します。
- **nElementsDim2:**マルチアレイのnWorkDim2から、コピーする要素数を指定します。全てをコピーするには、nElementsDim2:=0を設定しなければなりません。しかし、結果のうち特定の帯域幅のみ必要な場合は、データ全体をコピーする必要はありません。これにより、外部データバッファの必要サイズも削減します。
- **pStartIndex:**これは、マルチアレイが2次元以上ある場合や、全ての要素をコピーするわけではない場合に役立つオプションのパラメータです。マルチアレイからコピーする最初の要素のインデックスを指定します。代入する場合は、マルチアレイの次元数と同じ数の要素を持つUDINTの1次元配列を指す必要があります。
- **bNewResult:**この出力は、新しい結果がデータバッファに正常に書き込まれるたびにTRUEになります。
- **bError:**エラーが発生した場合、この出力はTRUEになります。
- **hrErrorcode:**エラーが発生した場合は、HRESULTのエラーコードが表示されます。可能性のある値は、エラーコード一覧 [▶ 367]に記載されています。この出力はメソッドの戻り値と同じです。
ヒント：タイムアウトが発生しても、入力データは失われません。次の呼び出しで処理されます。

Output3DStd() :

マルチアレイから外部の3次元データバッファにデータを書き込みます。メソッドが新しい結果もエラーも示さない場合、オブジェクトは入力データを待機します。これは解析チェーンにおける標準的な動作です。

このメソッドでは、パラメータ `nWorkDim0:=0`、`nWorkDim1:=1`、`nWorkDim2:=2`、`nElementsDim0:=0`、`nElementsDim1:=0`、`nElementsDim2:=0`、`pStartIndex:=0` のデフォルト値を使用します。

- **戻り値**：エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD Output3DStd : HRESULT
VAR_INPUT
    pDataOut      : POINTER TO BYTE; // address of data buffer
    nDataOutSize  : UDINT;           // size of data buffer in bytes
    eElementType  : E_MA_ElementTypeCode;
    nOptionPars   : DWORD;           // option mask
END_VAR
VAR_OUTPUT
    bNewResult    : BOOL;            // TRUE every time when data was written from MultiArray to data buffer.
    bError        : BOOL;            // TRUE if an error occurs.
    hrErrorCode   : HRESULT;         // '< 0' = error; '> 0' = info; '0' = no error/info
END_VAR
```

- `pDataOut`: 外部2次元データバッファのアドレス。
- `eElementType`: この入力は `E_MA_ElementTypeCode` [▶ 274] 型です。指定したマルチアレイバッファの要素型は、指定した外部データバッファの要素型と一致する必要があります。
- `bNewResult`: この出力は、新しい結果がデータバッファに正常に書き込まれるたびに TRUE になります。
- `bError`: エラーが発生した場合、この出力は TRUE になります。
- `hrErrorCode`: エラーが発生した場合は、HRESULT のエラーコードが表示されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。この出力はメソッドの戻り値と同じです。
ヒント：タイムアウトが発生しても、入力データは失われません。次の呼び出しで処理されます。

```
METHOD Init : HRESULT
VAR_INPUT
    nOwnID: UDINT; // ID for this FB instance
END_VAR
```

Init():

このメソッドは通常、状態監視アプリケーションでは必要ありません。これはファンクションブロック初期化の代替手段を提供するものです。Init() メソッドは、PLCの初期化フェーズ中にのみ呼び出すことができます。実行時には使用できません。メソッド `FB_init` または、属性 `'call_after_init'` の使用を参照してください（TwinCAT PLCリファレンスを参照）。さらに、これはファンクションブロックのカプセル化を容易にします。

Init()メソッドを使用して初期化を行う場合、ファンクションブロックインスタンスの入力パラメータを宣言で代入することはできません。

- **戻り値**：エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。
- `nOwnID`: 一意のIDでファンクションブロックインスタンスを識別します。これは常に0より大きくなければなりません。この目的のためには、列挙を定義するアプローチが確立されています。

PassInputs() :

API PLC リファレンス [▶ 79]で説明されているとおり、`FB_CMA_Source` インスタンスが呼び出され、信号データがターゲットブロックに転送される限り、解析チェーンの以降の全てのファンクションブロックを周期的に呼び出す必要があります。

アルゴリズムを一定時間実行しない方が有用な場合があります。例えば、いくつかのアルゴリズムは、事前に教示や設定をした後でのみ実行すべきです。ファンクションブロックは周期的に呼び出す必要がありますが、ファンクションブロックに到着したデータが[通信リング \[▶ 72\]](#)内で転送されれば十分です。これは、`Call()` メソッドの代わりに `PassInputs()` メソッドを使用して行われます。アルゴリズム自体はここでは呼び出されず、したがって結果も計算されず、出力バッファも生成されません。

- **戻り値**：エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD PassInputs : HRESULT
VAR_INPUT
END_VAR
```

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM, Tc3_CM_Base, Tc3_MultiArray

5.1.38 FB_CMA_Source

このファンクションブロックは、PLCデータバッファからマルチアレイバッファにデータを書き込みます。

マルチアレイの最大サイズに達するまで、連続的に入力データを累積します。マルチアレイが一杯になると、対象のAnalysis IDに転送されます。

FB_CMA_Source のインスタンスは、他の解析ファンクションブロックの対象として使用してはいけません。ソース機能のみを提供します。

エラー発生時には、時系列データの収集が中断される可能性があります。信号データが失われると、アルゴリズムの設定によっては、解析チェーンが予期せぬ結果をもたらす可能性があります。

入力と出力

出力バッファは、以下の定義（形状）に該当します。変数パラメータは、ファンクションブロック入力 stInitPars の一部です。

マルチアレイの所在	要素の型	次元	次元変数
出カストリーム	eTypeCode	nDims	aDimSizes

```
VAR_INPUT
    stInitPars      : ST_MA_MultiArray_InitPars;      // init parameter
    nOwnID          : UDINT;                          // ID for this FB instance
    aDestIDs        : ARRAY[1..cCMA_MaxDest] OF UDINT; // IDs of destinations for output
    nResultBuffers  : UDINT := 4;                     // number of MultiArrays which should be ini
    tialized for results (0 for no initialization)
    tTransferTimeout : LTIME := LTIME#40US;           // timeout checking off during access to int
    er-task FIFOs
END_VAR
```

入力パラメータ

このファンクションブロックの入力パラメータは初期化パラメータを表し、FBインスタンスの宣言で割り当て済みである必要があります。(または、Init()メソッド) 一度しか割り当てできません。実行中の変更はできません。

- stInitPars:ST_MA_MultiArray_InitPars [▶ 303]型の初期化パラメータのファンクションブロック固有の構造体。結果バッファにはマルチアレイバッファが指定されます。これらのパラメータは、上記の出力バッファの定義と紐づいていなければなりません。
- nOwnID:一意のIDでファンクションブロックインスタンスを識別します。これは常に0より大きくなければなりません。この目的のためには、列挙を定義するアプローチが確立されています。
- aDestIDs:転送先のIDを指定することで、結果の転送先を定義します。出力バッファの定義（上記を参照）は、選択された各転送先の入力バッファの定義と紐づいていなければなりません。
- nResultBuffers:このファンクションブロックは、指定された数のマルチアレイバッファにより転送トレイストリームを初期化します。デフォルト値は4です。
- tTransferTimeout:内部マルチアレイ転送の同期タイムアウトの設定です。[並列処理 \[▶ 72\]](#)の項を参照してください。

出力パラメータ

```
VAR_OUTPUT
    bError          : BOOL;                          // TRUE if an error occurs. Reset by next metho
    d call.
```

```

    hrErrorCode      : HRESULT;                // '< 0' = error; '> 0' = info; '0' = no error/
info
    ipErrorMessage  : I_TcMessage := fbErrorMessage; // Shows detailed information about occurred er
rors, warnings and more.
    nCntResults      : ULINT;                // Counts outgoing results (MultiArrays were ca
lculated and sent to transfer tray).
END_VAR

```

- **bError**:エラーが発生した場合の出力は、TRUEです。
- **hrErrorCode**:エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。
- **ipErrorMessage**:現在の戻り値に関するより詳細な情報を含みます。[エラーの説明と情報 \[▶ 81\]](#)の項を参照してください。この特別なインターフェースポインタは、常に有効かつ割り当て済みであるよう、内部的に保護されています。

プロパティ

次の入力呼び出しに関連する入力値のタイムスタンプを指定するため、オプションで以下のプロパティを使用できます。プロパティが設定されていない場合、現在のタイムスタンプと、2つの入力値間の標準的な時間差が内部的に自動決定されます。

[FB_CMA_Sink \[▶ 223\]](#)では、解析チェーンの入力値のタイムスパンとタイムスタンプが解析結果に割り当てられます。

名前	タイプ	アクセス	説明
nTimestamp	ULINT [T_DCTIME64; 1ns steps since 1.1.2000]	Set	次の入力呼び出しで転送される、最も古い値に関連するタイムスタンプの指定。 (例: StartTimeNextLatch - CycleTime)
tSamplePeriod	LTIME	Set	2つの入力値間の時間差の指定。 (例: CycleTime / cOversamples)。 この値は定数のため、指定は一度のみ。

メソッド

Input1D():

外部の1次元データバッファからマルチアレイにデータを書き込みます。このメソッドは、通常は新しい入力サンプルが利用可能になるたびに周期的に呼び出されなければなりません。

- **戻り値**: エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```

METHOD Input1D : HRESULT
VAR_INPUT
    pDataIn          : POINTER TO BYTE;        // address of data buffer (e.g. oversampling data) as one-
dimensional array
    nDataInSize       : UDINT;                // size of data buffer in bytes
    eElementType      : E_MA_ElementTypeCode;
    nWorkDim          : UDINT;                // It designates the dimension in the multi array being pr
ocessed.
    pStartIndex       : POINTER TO UDINT;      (* optional: default:0-
>internally handled; It designates the index of the first MultiArray element to be copied.
        If allocated it must point to a onedimensional array of
        UDINT with so many elements as dimensions of the MultiArray.
        Upon successful completion of the copy, corresponding S
tartIndex is incremented by the number of copied elements. *)
    nOptionPars       : DWORD;                // option mask
END_VAR
VAR_OUTPUT
    bNewResult        : BOOL;                // TRUE every time when outgoing MultiArray was calculated
and sent to transfer tray.

```



```

bError      : BOOL;           // TRUE if an error occurs.
hrErrorCode  : HRESULT;       // '< 0' = error; '> 0' = info; '0' = no error/info
END_VAR

```

- **pDataIn:** データバッファには全チャンネルのデータが含まれている必要があります。
- **eElementType:** この入力には **E_MA_ElementTypeCode** [▶ 274] 型です。指定されたマルチアレイバッファの要素型（初期化パラメータ）は、指定した外部データバッファの要素型と一致する必要があります。
- **nWorkDim:** データを累積する次元を定義します。一般的に、マルチアレイも1次元で、**nWorkDim := 0** です。しかし、マルチアレイは追加で次元を持つこともできます。ただし、その場合、追加された次元は処理されない可能性があります。
- **pStartIndex:** これは、マルチアレイが複数の次元を持つ場合に役立つオプションのパラメータです。コピーする最初のマルチアレイ要素のインデックスを指定します。代入する場合は、マルチアレイの次元数と同じ数の要素を持つ UDINT の1次元配列を指す必要があります。コピー処理が成功すると、対応する開始インデックスがコピーされた要素数分インクリメントされます。
- **bError:** エラーが発生した場合、この出力は TRUE になります。
- **hrErrorCode:** エラーが発生した場合は、HRESULT 型の詳細なエラーコードが表示されます。可能性のある値は、エラーコード一覧 [▶ 367] に記載されています。この出力はメソッドの戻り値と同じです。

Input1DStd() :

外部の1次元データバッファからマルチアレイにデータを書き込みます。このメソッドは、通常は新しい入力サンプルが利用可能になるたびに周期的に呼び出されなければなりません。

このメソッドでは、パラメータ **nWorkDim := 0** と **pStartIndex := 0** にデフォルト値を使用します。

- **戻り値:** エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、エラーコード一覧 [▶ 367] に記載されています。

```

METHOD Input1DStd : HRESULT
VAR_INPUT
    pDataIn      : POINTER TO BYTE;           // address of data buffer (e.g. oversampling data) as one-
dimensional array
    nDataInSize  : UDINT;                     // size of data buffer in bytes
    eElementType : E_MA_ElementTypeCode;
    nOptionPars  : DWORD;                     // option mask
END_VAR
VAR_OUTPUT
    bNewResult   : BOOL;                       // TRUE every time when outgoing MultiArray was calculated
    and sent to transfer tray.
    bError       : BOOL;                       // TRUE if an error occurs.
    hrErrorCode  : HRESULT;                     // '< 0' = error; '> 0' = info; '0' = no error/info
END_VAR

```

- **pDataIn:** データバッファには全チャンネルのデータが含まれている必要があります。
- **eElementType:** この入力には **E_MA_ElementTypeCode** [▶ 274] 型です。指定されたマルチアレイバッファの要素型（初期化パラメータ）は、指定した外部データバッファの要素型と一致する必要があります。
- **bError:** エラーが発生した場合、この出力は TRUE になります。
- **hrErrorCode:** エラーが発生した場合は、HRESULT 型の詳細なエラーコードが表示されます。可能性のある値は、エラーコード一覧 [▶ 367] に記載されています。この出力はメソッドの戻り値と同じです。

Input2D():

外部の2次元データバッファからマルチアレイにデータを書き込みます。このメソッドは、通常は新しい入力サンプルが利用可能になるたびに周期的に呼び出されなければなりません。

- **戻り値:** エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、エラーコード一覧 [▶ 367] に記載されています。

```

METHOD Input2D : HRESULT
VAR_INPUT
    pDataIn      : POINTER TO BYTE;           // address of data buffer (e.g. oversampling data) as twod
imensional array (e.g. [1..channels, 1..oversamples] )
    nDataInSize  : UDINT;                     // size of data buffer in bytes
    eElementType : E_MA_ElementTypeCode;

```

```

    nWorkDim0      : UDINT:=0;          // It designates the first dimension in the MultiArray bei
ng processed. (e.g. 1..channels)
    nWorkDim1      : UDINT:=1;          // It designates the second dimension in the MultiArray be
ing processed.
    pStartIndex    : POINTER TO UDINT;  (* optional: default:0->
internally handled; It designates the index of the first MultiArray element to be copied.
                                         If allocated it must point to a onedimensional array of
UDINT with so many elements as dimensions of the MultiArray.
                                         Upon successful completion of the copy, corresponding S
tartIndex is incremented by the number of copied elements. *)
    nOptionPars    : DWORD;            // option mask
END_VAR
VAR_OUTPUT
    bNewResult     : BOOL;              // TRUE every time when outgoing MultiArray was calculated
and sent to transfer tray.
    bError         : BOOL;              // TRUE if an error occurs.
    hrErrorCode    : HRESULT;           // '< 0' = error; '> 0' = info; '0' = no error/info
END_VAR

```

- **pDataIn:** データバッファには全チャンネルのデータが含まれている必要があります。
- **eElementType:** この入力には E_MA_ElementTypeCode [▶ 274] 型です。指定されたマルチアレイバッファの要素型（初期化パラメータ）は、指定した外部データバッファの要素型と一致する必要があります。
- **nWorkDim0:** チャンネル数と一致する次元を定義します。一般的に、マルチアレイも2次元で、 $nWorkDim0 := 0$ です。しかし、マルチアレイは追加で次元を持つこともできます。ただし、その場合、追加された次元は処理されない可能性があります。
例：指定されたデータバッファの最初のインデックスがチャンネルを表し、マルチアレイの2番目の次元がチャンネル数をカウントする場合は、 $nWorkDim0 := 1$ を設定してください。
- **nWorkDim1:** データを累積する次元を定義します。一般的にマルチアレイも2次元で、 $nWorkDim1 := 1$ です。マルチアレイは追加で次元を持つこともできます。ただし、その場合、追加された次元は処理されない可能性があります。
例：指定されたデータバッファの2番目のインデックスが累積されたデータを表し、マルチアレイの最初の次元がデータを収集する場合は、 $nWorkDim1 := 0$ を設定してください。
- **pStartIndex:** これは、マルチアレイが2次元以上である場合に役立つオプションのパラメータです。コピーする最初のマルチアレイ要素のインデックスを指定します。代入する場合は、マルチアレイの次元数と同じ数の要素を持つ UDINT の1次元配列を指す必要があります。コピー処理が成功すると、対応する開始インデックスがコピーされた要素数分インクリメントされます。
- **bError:** エラーが発生した場合、この出力は TRUE になります。
- **hrErrorCode:** エラーが発生した場合は、HRESULT のエラーコードが表示されます。可能性のある値は、エラーコード一覧 [▶ 367] に記載されています。この出力はメソッドの戻り値と同じです。

Input2DStd() :

外部の2次元データバッファからマルチアレイにデータを書き込みます。このメソッドは、通常は新しい入力サンプルが利用可能になるたびに周期的に呼び出されなければなりません。

このメソッドでは、パラメータ $nWorkDim0 := 0$ 、 $nWorkDim1 := 1$ 、 $pStartIndex := 0$ にデフォルト値を使用します。

- **戻り値:** エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、エラーコード一覧 [▶ 367] に記載されています。

```

METHOD Input2DStd : HRESULT
VAR_INPUT
    pDataIn        : POINTER TO BYTE;    // address of data buffer (e.g. oversampling data) as twod
imensional array (e.g. [1..channels, 1..oversamples] )
    nDataInSize    : UDINT;              // size of data buffer in bytes
    eElementType   : E_MA_ElementTypeCode;
    nOptionPars    : DWORD;              // option mask
END_VAR
VAR_OUTPUT
    bNewResult     : BOOL;              // TRUE every time when outgoing MultiArray was calculated
and sent to transfer tray.
    bError         : BOOL;              // TRUE if an error occurs.
    hrErrorCode    : HRESULT;           // '< 0' = error; '> 0' = info; '0' = no error/info
END_VAR

```

- **pDataIn:** データバッファには全チャンネルのデータが含まれている必要があります。

- **eElementType**: この入力には E_MA_ElementTypeCode [▶ 274] 型です。指定されたマルチアレイバッファの要素型（初期化パラメータ）は、指定した外部データバッファの要素型と一致する必要があります。
- **bError**: エラーが発生した場合、この出力は TRUE になります。
- **hrErrorCode**: エラーが発生した場合は、HRESULT のエラーコードが表示されます。可能性のある値は、エラーコード一覧 [▶ 367] に記載されています。この出力はメソッドの戻り値と同じです。

Input3D():

外部の3次元データバッファからマルチアレイにデータを書き込みます。このメソッドは、通常は新しい入力サンプルが利用可能になるたびに周期的に呼び出されなければなりません。

- **戻り値**: エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、エラーコード一覧 [▶ 367] に記載されています。

```
METHOD Input2D : HRESULT
VAR_INPUT
    pDataIn      : POINTER TO BYTE;          // address of data buffer (e.g. oversampling data) as twod
dimensional array (e.g. [1..channels, 1..oversamples] )
    nDataInSize  : UDINT;                    // size of data buffer in bytes
    eElementType : E_MA_ElementTypeCode;
    nWorkDim0    : UDINT:=0;                 // It designates the first dimension in the MultiArray bei
ng processed. (e.g. 1..channels)
    nWorkDim1    : UDINT:=1;                 // It designates the second dimension in the MultiArray be
ing processed.
    nWorkDim2    : UDINT:=2;                 // It designates the third dimension in the MultiArray bei
ng processed.
    pStartIndex  : POINTER TO UDINT;         (* optional: default:0->
internally handled; It designates the index of the first MultiArray element to be copied.
If allocated it must point to a onedimensional array of
UDINT with so many elements as dimensions of the MultiArray.
Upon successful completion of the copy, corresponding S
tartIndex is incremented by the number of copied elements. *)
    nOptionPars  : DWORD;                    // option mask
END_VAR
VAR_OUTPUT
    bNewResult   : BOOL;                     // TRUE every time when outgoing MultiArray was calculated
and sent to transfer tray.
    bError       : BOOL;                     // TRUE if an error occurs.
    hrErrorCode  : HRESULT;                  // '< 0' = error; '> 0' = info; '0' = no error/info
END_VAR
```

- **pDataIn**: データバッファには全チャンネルのデータが含まれている必要があります。
- **eElementType**: この入力には E_MA_ElementTypeCode [▶ 274] 型です。指定されたマルチアレイバッファの要素型（初期化パラメータ）は、指定した外部データバッファの要素型と一致する必要があります。
- **nWorkDim0**: チャンネル数と一致する次元を定義します。一般的に、マルチアレイも2次元で、 $nWorkDim0 := 0$ です。しかし、マルチアレイは追加で次元を持つこともできます。ただし、その場合、追加された次元は処理されない可能性があります。
例：指定されたデータバッファの最初のインデックスがチャンネルを表し、マルチアレイの2番目の次元がチャンネル数をカウントする場合は、 $nWorkDim0 := 1$ を設定してください。
- **nWorkDim1**: サブチャンネル数に相当する次元を定義します。一般的に、マルチアレイも3次元であり、 $nWorkDim0 := 0$ および $nWorkDim1 := 1$ です。しかし、マルチアレイは追加で次元を持つことができます。ただし、その場合、追加された次元は処理されない可能性があります。
例：指定されたデータバッファの2番目のインデックスがサブチャンネルを表し、マルチアレイの3番目の次元がサブチャンネルをカウントする場合は、 $nWorkDim1 := 2$ を設定してください。
- **nWorkDim2**: データを累積する次元を定義します。一般的にマルチアレイも2次元で、 $nWorkDim1 := 1$ です。マルチアレイは追加で次元を持つこともできます。ただし、その場合、追加された次元は処理されない可能性があります。
例：指定されたデータバッファの2番目のインデックスが累積されたデータを表し、マルチアレイの最初の次元がデータを収集する場合は、 $nWorkDim1 := 0$ を設定してください。
- **pStartIndex**: これは、マルチアレイが2次元以上である場合に役立つオプションのパラメータです。コピーする最初のマルチアレイ要素のインデックスを指定します。代入する場合は、マルチアレイの次元数と同じ数の要素を持つ UDINT の1次元配列を指す必要があります。コピー処理が成功すると、対応する開始インデックスがコピーされた要素数分インクリメントされます。
- **bError**: エラーが発生した場合、この出力は TRUE になります。

hrErrorCode:エラーが発生した場合は、HRESULT のエラーコードが表示されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。この出力はメソッドの戻り値と同じです。

Input3DStd() :

外部の3次元データバッファからマルチアレイにデータを書き込みます。このメソッドは、通常は新しい入力サンプルが利用可能になるたびに周期的に呼び出されなければなりません。

このメソッドでは、パラメータnWorkDim0 := 0、nWorkDim1 := 1、nWorkDim2 := 2、pStartIndex := 0 (にデフォルト値を使用します)。

- **戻り値** : エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD Input2DStd : HRESULT
VAR_INPUT
    pDataIn      : POINTER TO BYTE;          // address of data buffer (e.g. oversampling data) as two dimensional array (e.g. [1..channels, 1..oversamples] )
    nDataInSize  : UDINT;                    // size of data buffer in bytes
    eElementType : E_MA_ElementTypeCode;
    nOptionPars  : DWORD;                    // option mask
END_VAR
VAR_OUTPUT
    bNewResult   : BOOL;                     // TRUE every time when outgoing MultiArray was calculated and sent to transfer tray.
    bError       : BOOL;                     // TRUE if an error occurs.
    hrErrorCode  : HRESULT;                  // '< 0' = error; '> 0' = info; '0' = no error/info
END_VAR
```

- **pDataIn**:データバッファは、全てのチャンネルとサブチャンネルのデータを含んでいなければなりません。
- **eElementType**:この入力はE_MA_ElementTypeCode [[▶ 274](#)]型です。指定されたマルチアレイバッファの要素型（初期化パラメータ）は、指定した外部データバッファの要素型と一致する必要があります。
- **bError**:エラーが発生した場合、この出力はTRUEになります。

hrErrorCode:エラーが発生した場合は、HRESULT のエラーコードが表示されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。この出力はメソッドの戻り値と同じです。

```
METHOD Init : HRESULT
VAR_INPUT
    stInitPars   : ST_MA_MultiArray_InitPars; // init parameter
    nOwnID       : UDINT;                      // ID for this FB instance
    aDestIDs     : ARRAY[1..CMA_MaxDest] OF UDINT; // IDs of destinations for output
    nResultBuffers : UDINT := 4;               // number of MultiArrays which should be initialized for results (0 for no initialization)
END_VAR
```

Init():

このメソッドは通常、状態監視アプリケーションでは必要ありません。これはファンクションブロック初期化の代替手段を提供するものです。Init() メソッドは、PLCの初期化フェーズ中にのみ呼び出すことができます。実行時には使用できません。メソッドFB_init または、属性'call_after_init'の使用を参照してください（TwinCAT PLCリファレンスを参照）。さらに、これはファンクションブロックのカプセル化を容易にします。

Init()メソッドを使用して初期化を行う場合、ファンクションブロックインスタンスの入力パラメータを宣言で代入することはできません。

- **戻り値** : エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。
- **stInitPars**:ST_MA_MultiArray_InitPars [[▶ 303](#)]型の初期化パラメータのファンクションブロック固有の構造体。結果バッファにはマルチアレイバッファが指定されます。これらのパラメータは、上記の出力バッファの定義と紐づいていなければなりません。
- **nOwnID**:一意のIDでファンクションブロックインスタンスを識別します。これは常に0より大きくなければなりません。この目的のためには、列挙を定義するアプローチが確立されています。
- **aDestIDs**:転送先のIDを指定することで、結果の転送先を定義します。出力バッファの定義（上記を参照）は、選択された各転送先の入力バッファの定義と紐づいていなければなりません。

- **nResultBuffers**: このファンクションブロックは、指定された数のマルチアレイバッファにより転送ストリームを初期化します。デフォルト値は4です。

ResetData() :

このメソッドは、現在の出力バッファ（マルチアレイ）を最初から再び埋められるように、すでに追加されたすべてのデータセットを削除します。充填時に外部インデックス（pStartIndex パラメータ）を使用する場合、明示的にリセットする必要があります。

- **戻り値** : エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、エラーコード一覧に記載されています。

```
METHOD ResetData : HRESULT
VAR_INPUT
END_VAR
```

ResetAnalysisChain() :

このメソッドを呼び出すと、解析チェーンの全アルゴリズムが自動的にリセットされます。内部的には、新しいデータセットを受け入れる前に、毎回ResetData()が実行されます。その結果、FB_CMA_Sourceに対して ResetData() を実行することが含まれます。

解析チェーンが特定の期間のみ有効である場合、この方法では次回実行前に全てのアルゴリズムをリセットするオプションを提供します。

入力メソッドの呼び出し時にエラーが発生し、時系列データの収集が中断される可能性があります。解析チェーン内の以下のアルゴリズムがスペクトルを計算する場合、入力メソッドを呼び出す際にエラーが発生した場合、ResetAnalysisChain() メソッドを呼び出すことができます。時系列データが断片化しているため、正しいスペクトルを計算することはできません。

- **戻り値** : エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、エラーコード一覧に記載されています。

```
METHOD ResetAnalysisChain : HRESULT
VAR_INPUT
END_VAR
```

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM, Tc3_CM_Base, Tc3_MultiArray

5.1.39 FB_CMA_SourcePaired

このファンクションブロックは、PLCデータバッファから1対のマルチアレイバッファにデータを書き込みます。

マルチアレイの最大サイズに達するまで、連続的に入力データを累積します。これらが完全に埋まると、対象のAnalysis IDに転送されます。

● データの同期

i 最初のストリームから2番目のストリームへのデータの割り当ては、直接的に行われます。すなわち、ストリームAの最初の値は、ストリームBの最初の値に対応しています。位置や振動値などの計測データは、事前に同期しておく必要があります。

FB_CMA_Source のインスタンスは、他の解析ファンクションブロックの対象として使用してはいけません。ソース機能のみを提供します。

エラー発生時には、時系列データの収集が中断される可能性があります。信号データが失われると、アルゴリズムの設定によっては、解析チェーンが予期せぬ結果をもたらす可能性があります。

入力と出力

出力バッファは、以下の定義（形状）に該当します。変数パラメータは、ファンクションブロック入力 `stInitPars` の一部です。

マルチアレイの所在	要素の型	次元	次元変数
出カストリームA	eTypeCode	nDims	aDimSizes
出カストリームB	eTypeCode	nDims	aDimSizes

```

VAR_INPUT
    stInitParsA      : ST_MA_MultiArray_InitPars;      // init parameters for first MultiArray
    stInitParsB      : ST_MA_MultiArray_InitPars;      // init parameters for second MultiArray
    nOwnID            : UDINT;                          // ID for this FB instance
    aDestIDs          : ARRAY[1..cCMA_MaxDest] OF UDINT; // IDs of destinations for output
    nResultBuffers    : UDINT := 4;                    // number of MultiArrays which should be ini
tialized for results (0 for no initialization)
    tTransferTimeout  : LTIME := LTIME#40US;           // timeout checking off during access to int
er-task FIFOs
END_VAR

```

入力パラメータ

このファンクションブロックの入力パラメータは初期化パラメータを表し、FBインスタンスの宣言で割り当て済みである必要があります。(または、Init()メソッド) 一度しか割り当てできません。実行中の変更はできません。

- `stInitParsA`: `ST_MA_MultiArray_InitPars` [▶ 303]型の初期化パラメータのファンクションブロック固有の構造体。結果バッファにはマルチアレイバッファが指定されます。これらのパラメータは、上記の出力バッファの定義と紐づいていなければなりません。
- `stInitParsB`: `ST_MA_MultiArray_InitPars` [▶ 303]型の初期化パラメータのファンクションブロック固有の構造体。結果バッファにはマルチアレイバッファが指定されます。これらのパラメータは、上記の出力バッファの定義と紐づいていなければなりません。
- `nOwnID`: 一意のIDでファンクションブロックインスタンスを識別します。これは常に0より大きくなければなりません。この目的のためには、列挙を定義するアプローチが確立されています。
- `aDestIDs`: 転送先のIDを指定することで、結果の転送先を定義します。出力バッファの定義（上記を参照）は、選択された各転送先の入力バッファの定義と紐づいていなければなりません。
- `nResultBuffers`: このファンクションブロックは、指定された数のマルチアレイバッファにより転送トレイストリームを初期化します。デフォルト値は4です。
- `tTransferTimeout`: 内部マルチアレイ転送の同期タイムアウトの設定です。並列処理 [▶ 72]の項を参照してください。

出力パラメータ

```

VAR_OUTPUT
    bError            : BOOL;                          // TRUE if an error occurs. Reset by next metho
d call.
    hrErrorCode       : HRESULT;                       // '< 0' = error; '> 0' = info; '0' = no error/
info
    ipErrorMessage    : I_TcMessage := fbErrorMessage; // Shows detailed information about occurred er
rors, warnings and more.
    nCntResults       : ULINT;                         // Counts outgoing results (MultiArrays were ca
lculated and sent to transfer tray).
END_VAR

```

- `bError`: エラーが発生した場合の出力は、TRUEです。
- `hrErrorCode`: エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧](#) [▶ 367]に記載されています。
- `ipErrorMessage`: 現在の戻り値に関するより詳細な情報を含みます。[エラーの説明と情報](#) [▶ 81]の項を参照してください。この特別なインターフェースポイントは、常に有効かつ割り当て済みであるよう、内部的に保護されています。

プロパティ

次の入力呼び出しに関連する入力値のタイムスタンプを指定するため、オプションで以下のプロパティを使用できます。プロパティが設定されていない場合、現在のタイムスタンプと、2つの入力値間の標準的な時間差が内部的に自動決定されます。

FB_CMA_Sink [▶ 223]では、解析チェーンの入力値のタイムスパンとタイムスタンプが解析結果に割り当てられます。

名前	タイプ	アクセス	説明
nTimestamp	ULINT [T_DCTIME64; 1ns steps since 1.1.2000]	Set	次の入力呼び出しで転送される、最も古い値に関連するタイムスタンプの指定。 (例 : StartTimeNextLatch - CycleTime)
tSamplePeriod	LTIME	Set	2つの入力値間の時間差の指定。 (例 : CycleTime / cOversamples)。 この値は定数のため、指定は一度のみ。

メソッド

Input1D():

外部の1次元データバッファから2つのマルチアレイにデータを書き込みます。このメソッドは、通常は新しい入力サンプルが利用可能になるたびに周期的に呼び出されなければなりません。

- **戻り値** : エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD Input1D : HRESULT
VAR_INPUT
    pDataInA      : POINTER TO BYTE;           // address of data buffer (e.g. oversampling data) as one-
dimensional array
    nDataInSizeA  : UDINT;                     // size of data buffer in bytes
    pDataInB      : POINTER TO BYTE;           // address of data buffer (e.g. oversampling data) as one-
dimensional array
    nDataInSizeB  : UDINT;                     // size of data buffer in bytes
    eElementTypeA : E_MA_ElementTypeCode;      // valid types: LREAL, INT32, UINT64, LCOMPLEX
    eElementTypeB : E_MA_ElementTypeCode;      // valid types: LREAL, INT32, UINT64, LCOMPLEX
    nWorkDimA     : UDINT;                     // It designates the dimension in the first multi array be
ing processed.
    nWorkDimB     : UDINT;                     // It designates the dimension in the second multi array b
eing processed.
    pStartIndexA  : POINTER TO UDINT;          (* optional: default:0-
>internally handled; It designates the index of the first MultiArray element to be copied.
    If allocated it must point to a onedimensional array of UDINT with so many
elements as dimensions of the MultiArray.
    Upon successful completion of the copy, corresponding StartIndex is increme
nted by the number of copied elements. *)
    pStartIndexB  : POINTER TO UDINT;          // see pStartIndexA
    nOptionPars   : DWORD;                     // option mask
END_VAR
VAR_OUTPUT
    bNewResult    : BOOL;                      // TRUE every time when outgoing MultiArray was calculated
and sent to transfer tray.
    bError        : BOOL;                      // TRUE if an error occurs.
    hrErrorCode   : HRESULT;                   // '< 0' = error; '> 0' = info; '0' = no error/info
END_VAR
```

- **pDataInA pDataInB** : データバッファには全チャンネルのデータが含まれている必要があります。
- **eElementTypeA eElementTypeB** : この入力は **E_MA_ElementTypeCode [▶ 274]** 型です。指定されたマルチアレイバッファの要素型 (初期化パラメータ) は、指定した外部データバッファの要素型と一致する必要があります。

- **nWorkDimA nWorkDimB:** データを累積する次元を定義します。一般的に、マルチアレイも1次元で、**nWorkDimA := 0** または **nWorkDimB := 0** です。しかし、マルチアレイは追加で次元を持つこともできます。ただし、その場合、追加された次元は処理されない可能性があります。
- **pStartIndexA pStartIndexB:** これは、マルチアレイが複数の次元を持つ場合に役立つオプションのパラメータです。コピーする最初のマルチアレイ要素のインデックスを指定します。代入する場合は、マルチアレイの次元数と同じ数の要素を持つ UDINT の1次元配列を指す必要があります。コピー処理が成功すると、対応する開始インデックスがコピーされた要素数分インクリメントされます。
- **bError:** エラーが発生した場合、この出力は TRUE になります。
- **hrErrorCode:** エラーが発生した場合は、HRESULT 型の詳細なエラーコードが表示されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#) で定義されています。この出力はメソッドの戻り値と同じです。

Input1DStd() :

外部の1次元データバッファから2つのマルチアレイにデータを書き込みます。このメソッドは、通常は新しい入力サンプルが利用可能になるたびに周期的に呼び出されなければなりません。

このメソッドでは、パラメータ **nWorkDimA := 0, nWorkDimB := 0** および **pStartIndexA := 0, pStartIndexB := 0** にデフォルト値を使用します。

- **戻り値:** エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#) に記載されています。

```
METHOD Input1DStd : HRESULT
VAR_INPUT
    pDataInA      : POINTER TO BYTE;          // address of data buffer (e.g. oversampling data) as one-
dimensional array
    nDataInSizeA  : UDINT;                    // size of data buffer in bytes
    pDataInB      : POINTER TO BYTE;          // address of data buffer (e.g. oversampling data) as one-
dimensional array
    nDataInSizeB  : UDINT;                    // size of data buffer in bytes
    eElementTypeA : E_MA_ElementTypeCode;    // valid types: LREAL, INT32, UINT64, LCOMPLEX
    eElementTypeB : E_MA_ElementTypeCode;    // valid types: LREAL, INT32, UINT64, LCOMPLEX
    nOptionPars   : DWORD;                    // option mask
END_VAR
VAR_OUTPUT
    bNewResult    : BOOL;                     // TRUE every time when outgoing MultiArray was calculated
and sent to transfer tray.
    bError        : BOOL;                     // TRUE if an error occurs.
    hrErrorCode   : HRESULT;                  // '< 0' = error; '> 0' = info; '0' = no error/info
END_VAR
```

- **pDataInA pDataInB:** データバッファには全チャンネルのデータが含まれている必要があります。
- **eElementTypeA eElementTypeB:** この入力には **E_MA_ElementTypeCode [▶ 274]** 型です。指定されたマルチアレイバッファの要素型（初期化パラメータ）は、指定した外部データバッファの要素型と一致する必要があります。
- **bError:** エラーが発生した場合、この出力は TRUE になります。
- **hrErrorCode:** エラーが発生した場合は、HRESULT 型の詳細なエラーコードが表示されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#) で定義されています。この出力はメソッドの戻り値と同じです。

Input2D():

外部の2次元データバッファから1対のマルチアレイにデータを書き込みます。このメソッドは、通常は新しい入力サンプルが利用可能になるたびに周期的に呼び出されなければなりません。

- **戻り値:** エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#) に記載されています。

```
METHOD Input2D : HRESULT
VAR_INPUT
    pDataInA      : POINTER TO BYTE;          // address of data buffer (e.g. oversampling data) as one-
dimensional array
    nDataInSizeA  : UDINT;                    // size of data buffer in bytes
    pDataInB      : POINTER TO BYTE;          // address of data buffer (e.g. oversampling data) as one-
dimensional array
    nDataInSizeB  : UDINT;                    // size of data buffer in bytes
    eElementTypeA : E_MA_ElementTypeCode;    // valid types: LREAL, INT32, UINT64, LCOMPLEX
    eElementTypeB : E_MA_ElementTypeCode;    // valid types: LREAL, INT32, UINT64, LCOMPLEX
    nWorkDimA0    : UDINT;                    // It designates the first dimension in the first multi ar
```

```

ray being processed.
  nWorkDimA1      : UDINT;                // It designates the second dimension in the first multi a
rray being processed.
  nWorkDimB0      : UDINT;                // It designates the first dimension in the second multi a
rray being processed.
  nWorkDimB1      : UDINT;                // It designates the second dimension in the second multi
array being processed.
  pStartIndexA    : POINTER TO UDINT;    (* optional: default:0-
>internally handled; It designates the index of the first MultiArray element to be copied.
    If allocated it must point to a onedimensional array of UDINT with so many
elements as dimensions of the MultiArray.
    Upon successful completion of the copy, corresponding StartIndex is increme
nted by the number of copied elements. *)
  pStartIndexB    : POINTER TO UDINT;    // see pStartIndexA
  nOptionPars     : DWORD;               // option mask
END_VAR
VAR_OUTPUT
  bNewResult      : BOOL;                // TRUE every time when outgoing MultiArray was calculated
and sent to transfer tray.
  bError          : BOOL;                // TRUE if an error occurs.
  hrErrorCode     : HRESULT;             // '< 0' = error; '> 0' = info; '0' = no error/info
END_VAR

```

- **pDataInA pDataInB:** データバッファには全チャンネルのデータが含まれている必要があります。
- **eElementTypeA eElementTypeB:** この入力には **E_MA_ElementTypeCode** [▶ 274] 型です。指定されたマルチアレイバッファの要素型（初期化パラメータ）は、指定した外部データバッファの要素型と一致する必要があります。
- **nWorkDimA0 nWorkDimB0:** チャンネル数に一致する次元を定義します。一般的に、マルチアレイも2次元で、 $nWorkDimA0 := 0$ または $nWorkDimB0 := 0$ です。しかし、マルチアレイは追加で次元を持つこともできます。ただし、その場合は、追加された次元は処理されない可能性があります。
例：指定されたデータバッファの最初のインデックスがチャンネルを表し、マルチアレイの2番目の次元がチャンネルを列挙する場合、 $nWorkDimA0 := 1$ または $nWorkDimB0 := 1$ を設定してください。
- **nWorkDimA1 nWorkDimB1:** データを累積する次元を定義します。一般的に、マルチアレイも2次元で、 $nWorkDimA1 := 1$ または $nWorkDimB1 := 1$ です。しかし、マルチアレイは追加の次元を持つこともできます。ただし、その場合は追加された次元は処理されない可能性があります。
例：指定されたデータバッファの2番目のインデックスが収集されたデータを表すが、マルチアレイの最初の次元がデータ収集する場合、 $nWorkDimB1 := 0$ または $nWorkDimB1 := 0$ を設定してください。
- **pStartIndexA pStartIndexB:** これは、マルチアレイが複数の次元を持つ場合に役立つオプションのパラメータです。コピーする最初のマルチアレイ要素のインデックスを指定します。代入する場合は、マルチアレイの次元数と同じ数の要素を持つ UDINT の1次元配列を指す必要があります。コピー処理が成功すると、対応する開始インデックスがコピーされた要素数分インクリメントされます。
- **bError:** エラーが発生した場合、この出力は TRUE になります。
- **hrErrorCode:** エラーが発生した場合は、HRESULT 型の詳細なエラーコードが表示されます。可能性のある値は、**エラーコード一覧** [▶ 367] で定義されています。この出力はメソッドの戻り値と同じです。

Input2DStd() :

外部の2次元データバッファから1対のマルチアレイにデータを書き込みます。このメソッドは、通常は新しい入力サンプルが利用可能になるたびに周期的に呼び出されなければなりません。

このメソッドでは、パラメータ $nWorkDimA0 := 0, nWorkDimA1 := 1, nWorkDimB0 := 0, nWorkDimB1 := 1$ および $pStartIndexA := 0, pStartIndexB := 0$ にデフォルト値を使用します。

- **戻り値:** エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、**エラーコード一覧** [▶ 367] に記載されています。

```

METHOD Input2D : HRESULT
VAR_INPUT
  pDataInA        : POINTER TO BYTE;    // address of data buffer (e.g. oversampling data) as one-
dimensional array
  nDataInSizeA    : UDINT;               // size of data buffer in bytes
  pDataInB        : POINTER TO BYTE;    // address of data buffer (e.g. oversampling data) as one-
dimensional array
  nDataInSizeB    : UDINT;               // size of data buffer in bytes
  eElementTypeA   : E_MA_ElementTypeCode; // valid types: LREAL, INT32, UINT64, LCOMPLEX
  eElementTypeB   : E_MA_ElementTypeCode; // valid types: LREAL, INT32, UINT64, LCOMPLEX
  nOptionPars     : DWORD;               // option mask
END_VAR

```

```

VAR_OUTPUT
  bNewResult      : BOOL;                // TRUE every time when outgoing MultiArray was calculated
and sent to transfer tray.
  bError          : BOOL;                // TRUE if an error occurs.
  hrErrorCode     : HRESULT;             // '< 0' = error; '> 0' = info; '0' = no error/info
END_VAR

```

- **pDataInA pDataInB:** データバッファには全チャンネルのデータが含まれている必要があります。
- **eElementTypeA eElementTypeB:** この入力には **E_MA_ElementTypeCode** [▶ 274] 型です。指定されたマルチアレイバッファの要素型（初期化パラメータ）は、指定した外部データバッファの要素型と一致する必要があります。
- **bError:** エラーが発生した場合、この出力はTRUEになります。
- **hrErrorCode:** エラーが発生した場合は、HRESULT型の詳細なエラーコードが表示されます。可能性のある値は、エラーコード一覧 [▶ 367] で定義されています。この出力はメソッドの戻り値と同じです。

```

METHOD Init : HRESULT
VAR_INPUT
  stInitParsA      : ST_MA_MultiArray_InitPars;    // init parameters for first MultiArray
  stInitParsB      : ST_MA_MultiArray_InitPars;    // init parameters for second MultiArray
  nOwnID           : UDINT;                        // ID for this FB instance
  aDestIDs         : ARRAY[1..cCMA_MaxDest] OF UDINT; // IDs of destinations for output
  nResultBuffers   : UDINT := 4;                  // number of MultiArrays which should be initialized for results (0 for no initialization)
END_VAR

```

Init():

このメソッドは通常、状態監視アプリケーションでは必要ありません。これはファンクションブロック初期化の代替手段を提供するものです。Init() メソッドは、PLCの初期化フェーズ中にのみ呼び出すことができます。実行時には使用できません。メソッドFB_init または、属性'call_after_init'の使用を参照してください（TwinCAT PLCリファレンスを参照）。さらに、これはファンクションブロックのカプセル化を容易にします。

Init()メソッドを使用して初期化を行う場合、ファンクションブロックインスタンスの入力パラメータを宣言で代入することはできません。

- **戻り値:** エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、エラーコード一覧 [▶ 367] に記載されています。
- **stInitParsA:** ST_MA_MultiArray_InitPars [▶ 303] 型の初期化パラメータのファンクションブロック固有の構造体。結果バッファにはマルチアレイバッファが指定されます。これらのパラメータは、上記の出力バッファの定義と紐づいていなければなりません。
- **stInitParsB:** ST_MA_MultiArray_InitPars [▶ 303] 型の初期化パラメータのファンクションブロック固有の構造体。結果バッファにはマルチアレイバッファが指定されます。これらのパラメータは、上記の出力バッファの定義と紐づいていなければなりません。
- **nOwnID:** 一意のIDでファンクションブロックインスタンスを識別します。これは常に0より大きくなければなりません。この目的のためには、列挙を定義するアプローチが確立されています。
- **aDestIDs:** 転送先のIDを指定することで、結果の転送先を定義します。出力バッファの定義（上記を参照）は、選択された各転送先の入力バッファの定義と紐づいていなければなりません。
- **nResultBuffers:** このファンクションブロックは、指定された数のマルチアレイバッファにより転送ストリームを初期化します。デフォルト値は4です。

ResetData() :

このメソッドは、現在の出力バッファ（マルチアレイ）を最初から再び埋められるように、すでに追加されたすべてのデータセットを削除します。充填時に外部インデックス（pStartIndex パラメータ）を使用する場合、明示的にリセットする必要があります。

- **戻り値:** エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、エラーコード一覧 に記載されています。

```

METHOD ResetData : HRESULT
VAR_INPUT
END_VAR

```


要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM, Tc3_CM_Base, Tc3_MultiArray

5.1.40 FB_CMA_VibrationAssessment

実数値入力信号の振動を評価します。

ファンクションブロックFB_CMA_VibrationAssessmentは、ISO 10816-3に基づき実数値入力信号の振動評価を行います。これについては、アプリケーションの概念（振動評価 [▶ 33]）の項で詳しく説明しています。このファンクションブロックは、設定可能な周波数帯域における積分RMS値の計算と、シングルチャンネルおよびマルチチャンネルの入力データに対する分類を組み合わせています。

その結果、各周波数帯域について、計算された最高分類（範囲1..nMaxClasses）、関連する積分次数（範囲0..nOrder）、チャンネル（範囲1..nChannels）の3つの値を含む1次元配列が得られます。シークの分類結果は、aResult : ARRAY[1..nMaxBands] OF ARRAY [1..3] OF DINT の形式で2次元配列に保存することを推奨します。その後、各帯域のデータは以下のように利用可能です：[[class],[order],[channel]]。

このデータは、振動計測値に基づいて機械の状態に関する声明/評価を行うために使用することができます。ISO10816-3を例にとれば、機械の様々な速度に対する条件を同時に決定することができます。

● ウィンドウの長さに注意

I nWindowLength の値はnFFT_Length の値以下になるようにしてください。FFTの長さは、必要な周波数分解能に合わせるすることができます。一般的には、FFT長の約4/5の値がウィンドウ長としてよく使用されます。

nFFT_Length がnWindowLengthよりも大きい場合、FFTの周波数分解能が（したがって、戻り値ベクトルの長さも）増加します。長さの差は、フーリエ変換の前にゼロで満たされます。これは、より高い周波数分解能を得るためや、時間領域での逆変換による計算で円形エイリアシングを避けるために有用です。ただし、より高い周波数分解能にもかかわらず、結果にはそれ以上の情報は含まれていません。

メモリ特性

ウェルチ法を使用しているため、現在の入力データバッファと最後に転送されたバッファの両方が計算に使用されます。組み込まれるバッファの数は、選択したオーバーラップ(nOverlap)に依存します。周波数解析では、時系列データの段階的な変化が考慮されます。したがって、正しい結果を得るためには、使用する入力データバッファは、間隔やジャンプなしに連結していなければなりません。

マルチチャンネル入力データ処理時の動作

複数のチャンネル（nChannels > 1）を処理する場合、各チャンネルが異なる戻り値になる可能性があります。この場合、戻り値はファンクションブロックで個別に問い合わせできます。全てのチャンネルではなく、1つ以上のチャンネルの結果が許容されない場合、ファンクションブロックの値はeCM_InfRTIME_AmbiguousChannelResultsに対応します。全てのチャンネルの結果が許容されない場合、ファンクションブロック上の値はeCM_ErrRTIME_ErrornousChannelResults に対応します。

全チャンネルの戻り値のリストは、GetChannelErrors()のメソッドで照会できます。

サンプルの実装

実装用のサンプルは以下のリンクから入手できます：[ISO 10816-3に準拠した振動評価（コンパクト版）\[▶ 331\]](#)。

入力と出力

入力バッファと出力バッファは、以下の定義（入力/出力形状）のいずれかに該当します。変数パラメータは、ファンクションブロック入力stInitPars の一部です。

バージョン	入力バッファ（マルチアレイ入力ストリーム） 要素タイプ、次元数、次元サイズ	出力バッファ（マルチアレイ出力ストリーム） 要素タイプ、次元数、次元サイズ
標準バージョン (nChannels = 1)	LREAL, 1, nWindowLength-nOverlap	DINT, 1, nMaxBands*3
マルチチャンネルバージョン (nChannels > 1)	LREAL, 2, nChannels x nWindowLength-nOverlap	DINT, 1, nMaxBands*3

```

VAR_INPUT
    stInitPars      : ST_CM_VibrationAssessment_InitPars; // init parameter
    nOwnID          : UDINT;                             // ID for this FB instance
    aDestIDs        : ARRAY[1..cCMA_MaxDest] OF UDINT;    // IDs of destinations for output
    nResultBuffers  : UDINT := 4;                         // number of MultiArrays which should be
    initialized for results (0 for no initialization)
    tTransferTimeout : LTIME := LTIME#500US;              // timeout checking off during access to
    inter-task FIFOs
END_VAR

```

入力パラメータ

このファンクションブロックの入力パラメータは初期化パラメータを表し、FBインスタンスの宣言で割り当て済みである必要があります。(または、Init()メソッド) 一度しか割り当てできません。実行中の変更はできません。

- **stInitPars:** ST_CM_VibrationAssessment_InitPars [▶ 301] 型の初期化パラメータを持つ、ファンクションブロック固有の構造体。パラメータは、上記の入出力バッファの定義と紐づいていなければなりません。
- **nOwnID:** 一意のIDでファンクションブロックインスタンスを識別します。これは常に0より大きくなければなりません。この目的のためには、列挙を定義するアプローチが確立されています。
- **aDestIDs:** 転送先のIDを指定することで、結果の転送先を定義します。出力バッファの定義（上記を参照）は、選択された各転送先の入力バッファの定義と紐づいていなければなりません。
- **nResultBuffers:** このファンクションブロックは、指定された数のマルチアレイバッファにより転送トレイストリームを初期化します。デフォルト値は4です。
- **tTransferTimeout:** 内部マルチアレイ転送の同期タイムアウトの設定です。並列処理 [▶ 72]の項を参照してください。

出力パラメータ

```

VAR_OUTPUT
    bError          : BOOL;                               // TRUE if an error occurs. Reset by next method call.
    hrErrorCode      : HRESULT;                           // '< 0' = error; '> 0' = info; '0' = no error/info
    ipErrorMessage  : I_TcMessage := fbErrorMessage;    // Shows detailed information about occurred errors, warnings and more.
    nCntResults      : ULINT;                             // Counts outgoing results (MultiArrays were calculated and sent to transfer tray).
END_VAR

```

- **bError:** エラーが発生した場合の出力は、TRUEです。
- **hrErrorCode:** エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、エラーコード一覧 [▶ 367]に記載されています。
- **ipErrorMessage:** 現在の戻り値に関するより詳細な情報を含みます。エラーの説明と情報 [▶ 81]の項を参照してください。この特別なインターフェイスポインタは、常に有効かつ割り当て済みであるよう、内部的に保護されています。

メソッド

```

METHOD Init : HRESULT
VAR_INPUT
    stInitPars      : ST_CM_VibrationAssessment_InitPars; // init parameter
    nOwnID          : UDINT;                             // ID for this FB instance
    aDestIDs        : ARRAY[1..cCMA_MaxDest] OF UDINT;    // IDs of destinations for output
    nResultBuffers  : UDINT := 4;                         // number of MultiArrays which should be initialized for results (0 for no initialization)
END_VAR

```

Call():

このメソッドは、現在の入力データにアルゴリズムを適用するために、サイクルごとに呼び出されます。メソッドが新しい結果もエラーも示さない場合、ファンクションブロックは入力データを待機します。これは、解析チェーンのプロセスにおける規則的な動作です。

- **戻り値** : エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD Call : HRESULT
VAR_OUTPUT
    bNewResult : BOOL;          // TRUE every time when outgoing MultiArray was calculated and sent to transfer tray.
    bError      : BOOL;          // TRUE if an error occurs.
    hrErrorCode : HRESULT;       // '< 0' = error; '> 0' = info; '0' = no error/info
END_VAR
```

- **bError**: エラーが発生した場合の出力は、TRUEです。
- **hrErrorCode**: エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。この出力はメソッドの戻り値と同じです。



タイムアウトが発生した場合、または結果を格納するマルチアレイバッファがない場合、入力データも結果データも失われません。次の呼び出しに転送されます。

Init():

このメソッドは通常、状態監視アプリケーションでは必要ありません。これはファンクションブロック初期化の代替手段を提供するものです。Init() メソッドは、PLCの初期化フェーズ中にのみ呼び出すことができます。実行時には使用できません。メソッドFB_init または、属性'call_after_init'の使用を参照してください（TwinCAT PLCリファレンスを参照）。さらに、これはファンクションブロックのカプセル化を容易にします。

Init()メソッドを使用して初期化を行う場合、ファンクションブロックインスタンスの入力パラメータを宣言で代入することはできません。

- **戻り値** : エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。
- **stInitPars**: ST_CM_VibrationAssessment_InitPars [▶ 301] 型の初期化パラメータを持つ、ファンクションブロック固有の構造体。パラメータは、上記の入出力バッファの定義と紐づいていなければなりません。
- **nOwnID**: 一意のIDでファンクションブロックインスタンスを識別します。これは常に0より大きくなければなりません。この目的のためには、列挙を定義するアプローチが確立されています。
- **aDestIDs**: 転送先のIDを指定することで、結果の転送先を定義します。出力バッファの定義（上記を参照）は、選択された各転送先の入力バッファの定義と紐づいていなければなりません。
- **nResultBuffers**: このファンクションブロックは、指定された数のマルチアレイバッファにより転送トレイストリームを初期化します。デフォルト値は4です。

Configure() :

このメソッドを呼び出す際には、最初に分類の引数と周波数帯域を設定しておく必要があります。対応するPLCアレイは以下のように定義する必要があります。Configure()メソッドは、異なる引数セットを持つ新しい設定にも使用できます。

- **戻り値** : エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD Configure : HRESULT
VAR_INPUT
    pArg1 : POINTER TO LREAL; // pointer to 2-dimensional
    array (LREAL) of arguments for classification
    nArgSize1 : UDINT;         // size of arguments buffer in bytes
    pArg2 : POINTER TO LREAL; // pointer to 2-dimensional
    array (LREAL) of arguments for frequency bands
    nArgSize2 : UDINT;         // size of arguments buffer in bytes
END_VAR
```

入力バッファは、以下の定義（入力形状）のいずれかに紐づいています。変数パラメータは、ファンクションブロック入力stInitParsの一部です。周波数帯域ごとに設定する2つのパラメータは、[fLowerFrequencyLimit, fUpperFrequencyLimit]（入力バッファ2）です。

バージョン	入力バッファ 1（マルチアレイ 入カストリーム） 要素タイプ、次元数、次元サイズ	入力バッファ 2（マルチアレイ 入カストリーム） 要素タイプ、次元数、次元サイズ
全てのチャンネルで同一設定 (nChannels >= 1)	LREAL, 2, nOrder+1 x nMaxClasses	LREAL, 2, nMaxBands x 2

ResetData() :

このメソッドは、既に追加された全てのデータレコードを削除します。ファンクションブロックのメモリ特性を参照してください。ResetData() の後、Call() メソッドが再度呼び出された場合、有効な結果を計算するために内部メモリを補充する必要があります。

- **戻り値** : エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD ResetData : HRESULT
VAR_INPUT
END_VAR
```

PassInputs() :

API PLC リファレンス [▶ 79]で説明されているとおり、FB_CMA_Source インスタンスが呼び出され、信号データがターゲットブロックに転送される限り、解析チェーンの以降の全てのファンクションブロックを周期的に呼び出す必要があります。

アルゴリズムを一定時間実行しない方が有用な場合があります。例えば、いくつかのアルゴリズムは、事前に教示や設定をした後でのみ実行すべきです。ファンクションブロックは周期的に呼び出す必要がありますが、ファンクションブロックに到着したデータが[通信リング \[▶ 72\]](#)内で転送されれば十分です。これは、Call() メソッドの代わりにPassInputs() メソッドを使用して行われます。アルゴリズム自体はここでは呼び出されず、したがって結果も計算されず、出力バッファも生成されません。

- **戻り値** : エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD PassInputs : HRESULT
VAR_INPUT
END_VAR
```

GetChannelErrors() :

このメソッドは、複数のチャンネル（nChannels > 1）を処理する際に、チャンネル固有の戻り値のリストを照会できるようにします。呼び出しは、ファンクションブロックの戻り値が eCM_InfRTTime_AmbiguousChannelResults または eCM_ErrRTTime_ErrornousChannelResults のいずれかに該当する場合に有用です。

- **戻り値** : エラーコード一覧の読み込み処理に関する情報です。この値は、クエリが成功した場合は TRUE に、そうでない場合は FALSE に設定されます。

```
METHOD GetChannelErrors : BOOL
VAR_IN_OUT
aChannelErrors : ARRAY[*] OF HRESULT;
END_VAR
```

- aChannelErrors: HRESULT 型の長さnChannelsのエラーリストです。

類似したファンクションブロック

ファンクションブロックFB_CMA_IntegratedRMS [▶ 157] は、シングルおよびマルチチャンネルの実数値時系列の積分RMS値を計算します（オプション）。

ファンクションブロックFB_CMA_WatchUpperThresholds [▶ 246] は、マルチチャンネルデータの設定可能な閾値監視を実行します。

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM, Tc3_CM_Base



CM 3.1 では一部の機能がすでに使用可能です。[互換性 \[▶ 61\]](#)の項を参照してください。

5.1.41 FB_CMA_WatchUpperThresholds

マルチチャンネルデータの設定可能な閾値監視を実行します。

FB_CMA_DiscreteClassification [▶ 113]ファンクションブロックと同様に、このファンクションブロックは、設定可能な閾値に基づいて、マルチチャンネル信号の各チャンネルを設定可能な離散カテゴリに割り当てます。設定後、ファンクションブロックは各入力ベクトルに対して正確に2つの値を持つ1次元配列を計算します。両要素の型は符号付き32ビット整数です。結果の最初の値は、決定された最も高いカテゴリの番号を示し、2番目の値は最も高いカテゴリを持つチャンネル番号を示します。どちらの場合も番号はゼロから始まります。あるチャンネルの入力値が最低カテゴリの閾値と一致しない場合、結果の値は -1 になります。入力値があるカテゴリの閾値と等しい場合、そのカテゴリにカウントされます。複数のチャンネルに最高のカテゴリが割り当てられている場合、より小さいチャンネル番号が返されます。

設定

このファンクションブロックは、実行時に各チャンネルと各カテゴリの閾値を指定することで設定できます。

メモリ特性

bMemorizeを使用したファンクションブロックの設定により、最高閾値カテゴリの番号とトリガチャンネルの番号は、メソッドResetData()が呼び出されるまで保存されるか、または各ステップ後に値が再計算されます。

NaN（非数）の発生

入力値がNaNの場合、分類結果は-2 になります。出力にはNaN値は含まれないことが想定されます。

マルチチャンネル入力データ処理時の動作

複数のチャンネル（nChannels > 1）を処理する場合、各チャンネルが異なる戻り値になる可能性があります。この場合、戻り値はファンクションブロックで個別に問い合わせできます。全てのチャンネルではなく、1つ以上のチャンネルの結果が許容されない場合、ファンクションブロックの値はeCM_InfrTime_AmbiguousChannelResultsに対応します。全てのチャンネルの結果が許容されない場合、ファンクションブロック上の値はeCM_ErrRTime_ErrornousChannelResults に対応します。

全チャンネルの戻り値のリストは、GetChannelErrors()のメソッドで照会できます。

複数のサブチャンネル(nSubChannels > 0)を処理する場合、入出力データのフォーマットには特に注意が必要です。入力データが上流のファンクションブロックによるマルチチャンネルの結果で構成されている場合、nChannels の値を採用する必要があります。この場合、パラメータnSubChannels を使用して追加の設定を行います。

例：マルチチャンネルスペクトル（FB_CMA_MagnitudeSpectrumなど）の周波数チャンネルの統計的考察（FB_CMA_Quantilesなど）では、nChannels の値は、入力信号の数と同じである必要があります。サブチャンネルの数nSubChannels は、スペクトルの長さに対応します。

入力と出力

入力バッファと出力バッファは、以下の定義（入力/出力形状）のいずれかに該当します。変数パラメータは、ファンクションブロック入力stInitPars の一部です。

バージョン	入力バッファ（マルチアレイ入力ストリーム） 要素タイプ、次元数、次元サイズ	出力バッファ（マルチアレイ出力ストリーム） 要素タイプ、次元数、次元サイズ
標準バージョン (nSubChannels = 0)	LREAL, 1, nChannels	DINT (32bit), 1, 2
マルチチャンネルバージョン (nSubChannels > 0)	LREAL, 2, nChannels x nSubChannels	DINT (32bit), 2, nChannels x 2

```

VAR_INPUT
    stInitPars      : ST_CM_WatchUpperThresholds_InitPars; // init parameter
    nOwnID          : UDINT;                               // ID for this FB instance
    aDestIDs        : ARRAY[1..cCMA_MaxDest] OF UDINT;     // IDs of destinations for output
    nResultBuffers  : UDINT := 4;                          // number of MultiArrays which should be
    initialized for results (0 for no initialization)
    tTransferTimeout: LTIME := LTIME#500US;               // timeout checking off during access to
    inter-task FIFOs
END_VAR

```

入力パラメータ

このファンクションブロックの入力パラメータは初期化パラメータを表し、FBインスタンスの宣言で割り当て済みである必要があります。(または、Init()メソッド) 一度しか割り当てできません。実行中の変更はできません。

- **stInitPars:** ST_CM_WatchUpperThresholds_InitPars [▶ 302]型の初期化パラメータを持つファンクションブロック固有の構造体。パラメータは、上記の入出力バッファの定義と紐づいていなければなりません。
- **nOwnID:** 一意のIDでファンクションブロックインスタンスを識別します。これは常に0より大きくなければなりません。この目的のためには、列挙を定義するアプローチが確立されています。
- **aDestIDs:** 転送先のIDを指定することで、結果の転送先を定義します。出力バッファの定義（上記を参照）は、選択された各転送先の入力バッファの定義と紐づいていなければなりません。
- **nResultBuffers:** このファンクションブロックは、指定された数のマルチアレイバッファにより転送トレイストリームを初期化します。デフォルト値は4です。
- **tTransferTimeout:** 内部マルチアレイ転送の同期タイムアウトの設定です。[並列処理 \[▶ 72\]](#)の項を参照してください。

出力パラメータ

```

VAR_OUTPUT
    bError          : BOOL;                                // TRUE if an error occurs. Reset by next method call.
    hrErrorCode      : HRESULT;                             // '< 0' = error; '> 0' = info; '0' = no error/info
    ipErrorMessage  : I_TcMessage := fbErrorMessage; // Shows detailed information about occurred errors, warnings and more.
    nCntResults      : ULINT;                               // Counts outgoing results (MultiArrays were calculated and sent to transfer tray).
END_VAR

```

- **bError:** エラーが発生した場合の出力は、TRUEです。
- **hrErrorCode:** エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。
- **ipErrorMessage:** 現在の戻り値に関するより詳細な情報を含みます。[エラーの説明と情報 \[▶ 81\]](#)の項を参照してください。この特別なインターフェースポイントは、常に有効かつ割り当て済みであるよう、内部的に保護されています。

メソッド

Call():

このメソッドは、現在の入力データにアルゴリズムを適用するために、サイクルごとに呼び出されます。メソッドが新しい結果もエラーも示さない場合、ファンクションブロックは入力データを待機します。これは、解析チェーンのプロセスにおける規則的な動作です。

- **戻り値:** エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```

METHOD Call : HRESULT
VAR_OUTPUT
    bNewResult : BOOL;          // TRUE every time when outgoing MultiArray was calculated and sent to transfer tray.
    bError      : BOOL;          // TRUE if an error occurs.
    hrErrorCode : HRESULT;       // '< 0' = error; '> 0' = info; '0' = no error/info
END_VAR

```

- **bError**:エラーが発生した場合の出力は、TRUEです。
- **hrErrorCode**:エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。この出力はメソッドの戻り値と同じです。



タイムアウトが発生した場合、または結果を格納するマルチアレイバッファがない場合、入力データも結果データも失われません。次の呼び出しに転送されます。

Init():

このメソッドは通常、状態監視アプリケーションでは必要ありません。これはファンクションブロック初期化の代替手段を提供するものです。Init() メソッドは、PLCの初期化フェーズ中にのみ呼び出すことができます。実行時には使用できません。メソッドFB_init または、属性'call_after_init' の使用を参照してください（TwinCAT PLCリファレンスを参照）。さらに、これはファンクションブロックのカプセル化を容易にします。

Init()メソッドを使用して初期化を行う場合、ファンクションブロックインスタンスの入力パラメータを宣言で代入することはできません。

- **戻り値**: エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。
- **stInitPars**: ST_CM_WatchUpperThresholds_InitPars [▶ 302]型の初期化パラメータを持つファンクションブロック固有の構造体。パラメータは、上記の入出力バッファの定義と紐づいていなければなりません。
- **nOwnID**: 一意のIDでファンクションブロックインスタンスを識別します。これは常に0より大きくなければなりません。この目的のためには、列挙を定義するアプローチが確立されています。
- **aDestIDs**: 転送先のIDを指定することで、結果の転送先を定義します。出力バッファの定義（上記を参照）は、選択された各転送先の入力バッファの定義と紐づいていなければなりません。
- **nResultBuffers**: このファンクションブロックは、指定された数のマルチアレイバッファにより転送ストリームを初期化します。デフォルト値は4です。

Configure() :

分類の引数は、このメソッドを呼び出す際にあらかじめ設定する必要があります。対応するPLC配列は以下のように定義してください。Configure()メソッドは、異なる引数セットを持つ新しい設定にも使用できます。

- **戻り値**: エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```

METHOD Configure : HRESULT
VAR_INPUT
    pArg : POINTER TO LREAL; // pointer to array (LREAL) of arguments
    nArgSize : UDINT;        // size of arguments buffer in bytes
END_VAR

```

入力バッファは、以下の定義（入力形状）のいずれかに紐づいています。変数パラメータは、ファンクションブロック入力stInitPars の一部です。

バージョン	入力バッファ（マルチアレイ入力ストリーム）要素タイプ、次元数、次元サイズ
全てのチャンネルおよびサブチャンネルで同一設定	LREAL, 1, nMaxClasses
チャンネル別の設定 (nSubChannels = 0)	LREAL, 2, nChannels x nMaxClasses

バージョン	入力バッファ（マルチアレイ入力ストリーム） 要素タイプ、次元数、次元サイズ
サブチャンネル別の設定 (nSubChannels > 0)	LREAL, 2, nSubChannels x nMaxClasses
チャンネルおよびサブチャンネル別の設定 (nSubChannels > 0)	LREAL, 3, nChannels x nSubChannels x nMaxClasses

ResetData() :

このメソッドは、既に追加された全てのデータレコードを削除します。ファンクションブロックのメモリ特性を参照してください。ResetData() の後、Call() メソッドが再度呼び出された場合、有効な結果を計算するために内部メモリを補充する必要があります。

- **戻り値** : エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD ResetData : HRESULT
VAR_INPUT
END_VAR
```

あるいは、自動リセットのためにbMemorize=FALSE を初期化構造に設定することもできます。

PassInputs() :

API PLC リファレンス [▶ 79]で説明されているとおり、FB_CMA_Source インスタンスが呼び出され、信号データがターゲットブロックに転送される限り、解析チェーンの以降の全てのファンクションブロックを周期的に呼び出す必要があります。

アルゴリズムを一定時間実行しない方が有用な場合があります。例えば、いくつかのアルゴリズムは、事前に教示や設定をした後でのみ実行すべきです。ファンクションブロックは周期的に呼び出す必要がありますが、ファンクションブロックに到着したデータが[通信リング \[▶ 72\]](#)内で転送されれば十分です。これは、Call() メソッドの代わりにPassInputs() メソッドを使用して行われます。アルゴリズム自体はここでは呼び出されず、したがって結果も計算されず、出力バッファも生成されません。

- **戻り値** : エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD PassInputs : HRESULT
VAR_INPUT
END_VAR
```

GetChannelErrors() :

このメソッドは、複数のチャンネル（nChannels > 1）を処理する際に、チャンネル固有の戻り値のリストを照会できるようにします。呼び出しは、ファンクションブロックの戻り値が eCM_InfRTTime_AmbiguousChannelResults または eCM_ErrRTTime_ErrornousChannelResults のいずれかに該当する場合に有効です。

- **戻り値** : エラーコード一覧の読み込み処理に関する情報です。この値は、クエリが成功した場合は TRUE に、そうでない場合は FALSE に設定されます。

```
METHOD GetChannelErrors : BOOL
VAR_IN_OUT
aChannelErrors : ARRAY[*] OF HRESULT;
END_VAR
```

- aChannelErrors: HRESULT 型の長さnChannelsのエラーリストです。

類似したファンクションブロック

[FB_CMA_DiscreteClassification \[▶ 113\]](#) ファンクションブロックは、マルチチャンネル入力データを分類します。

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86, x64)	Tc3_CM, Tc3_CM_Base



CM 3.1 では一部の機能がすでに使用可能です。[互換性 \[▶ 61\]](#)の項を参照してください。

5.1.42 FB_CMA_ZoomFFT

定義された周波数帯域において、スペクトル値を効率的に計算します。

ファンクションブロックFB_CMA_ZoomFFTは、ファンクションブロックに適用される実数値入力信号 $x[n]$ のフーリエ変換のセクション（周波数帯域）を計算します。ファンクションブロックの結果は、検査対象の周波数帯域において FB_CMA_RealFFT の結果と同じですが、計算負荷はそれに応じて低くなります。

計算される周波数帯域（ズーム）は、中心周波数（センタリング周波数） $f_{\text{Center}}f_c$ と間引き係数（デシメーションファクタ） $n_{\text{DecimationFactor}}D$ によって定義されます。デシメーション係数は2の累乗（2、4、8、16、...）である必要があります。周波数帯域は次のように定義されます：

$$\left[f_c - \frac{D}{2} \Delta f, f_c + \frac{D}{2} \Delta f \right]$$

周波数帯域における周波数分解能 Δf は、スペクトル全体にわたる通常のFFTで期待されるとおりです。以下に注意してください。

$$\Delta f = \frac{f_s}{N} = \frac{f_s/D}{N/D} = \frac{\tilde{f}_s}{\tilde{N}} = \Delta \tilde{f}, \quad \tilde{f}_s = \frac{f_s}{D}, \quad \tilde{N} = \frac{N}{D}$$

サンプリングの定理：入力信号中の成分の最高周波数は、エイリアシング効果を避けるために、入力信号のサンプリングレートの最大半分程度であるべきである。

NaN（非数）の発生

入力ベクトルに1つ以上のNaN（非数）値が含まれる場合、出力セクタ全体がNaNで満たされます。詳細については別項[NaN値 \[▶ 71\]](#)を参照してください。



NaN値の取り扱い

NaN値につながる上記の状況を除外できない、または安全に無視できない場合、アプリケーションプログラムはこれらのエラー値を適切に処理できるようにしてください。

マルチチャンネル入力データ処理時の動作

複数のチャンネル（ $n_{\text{Channels}} > 1$ ）を処理する場合、各チャンネルが異なる戻り値になる可能性があります。この場合、戻り値はファンクションブロックで個別に問い合わせできます。全てのチャンネルではなく、1つ以上のチャンネルの結果が許容されない場合、ファンクションブロックの値は `eCM_InfRTIME_AmbiguousChannelResults` に対応します。全てのチャンネルの結果が許容されない場合、ファンクションブロック上の値は `eCM_ErrRTIME_ErrornousChannelResults` に対応します。

全チャンネルの戻り値のリストは、`GetChannelErrors()` のメソッドで照会できます。

サンプルの実装

実装用のサンプルは以下のリンクから入手できます：[ズームFFT \[▶ 357\]](#)

入力と出力

入力バッファと出力バッファは、以下の定義（入力/出力形状）のいずれかに該当します。変数パラメータは、ファンクションブロック入力 `stInitPars` の一部です。

バージョン	入力バッファ (マルチアレイ 入力 ストリーム) 要素タイプ、次元数、次元サイズ	出力バッファ (マルチアレイ 出力 ストリーム) 要素タイプ、次元数、次元サイズ
標準バージョン (nChannels = 1)	LREAL, 1, nFFT_Length	LCOMPLEX, 1, nFFT_Length/ (2*nDecimationFactor+1)
標準バージョン (nChannels = 1)	LCOMPLEX, 1, nFFT_Length	LCOMPLEX, 1, nFFT_Length/ (2*nDecimationFactor+1)
マルチチャンネルバージョン (nChannels > 1)	LREAL, 2, nChannels x nFFT_Length	LCOMPLEX, 2, nChannels x nFFT_Length/ (2*nDecimationFactor+1)
マルチチャンネルバージョン (nChannels > 1)	LCOMPLEX, 2, nChannels x nFFT_Length	LCOMPLEX, 2, nChannels x nFFT_Length/ (2*nDecimationFactor+1)

```

VAR_INPUT
    stInitPars      : ST_CM_ZoomFFT_InitPars;           // init parameter
    nOwnID           : UDINT;                           // ID for this FB instance
    aDestIDs         : ARRAY[1..cCMA_MaxDest] OF UDINT; // IDs of destinations for output
    nResultBuffers   : UDINT := 4;                      // number of MultiArrays which should be in
    tTransferTimeout : LTIME := LTIME#500US;           // timeout checking off during access to in
    ter-task FIFOs
END_VAR

```

入力パラメータ

このファンクションブロックの入力パラメータは初期化パラメータを表し、FBインスタンスの宣言で割り当て済みである必要があります。(または、Init()メソッド) 一度しか割り当てできません。実行中の変更はできません。

- **stInitPars:** ST_CM_ZoomFFT_InitPars [▶ 303] 型の初期化パラメータを持つファンクションブロック固有の構造体。パラメータは、上記の入出力バッファの定義と紐づいていなければなりません。
- **nOwnID:** 一意のIDでファンクションブロックインスタンスを識別します。これは常に0より大きくなければなりません。この目的のためには、列挙を定義するアプローチが確立されています。
- **aDestIDs:** 転送先のIDを指定することで、結果の転送先を定義します。出力バッファの定義（上記を参照）は、選択された各転送先の入力バッファの定義と紐づいていなければなりません。
- **nResultBuffers:** このファンクションブロックは、指定された数のマルチアレイバッファにより転送トレイストリームを初期化します。デフォルト値は4です。
- **tTransferTimeout:** 内部マルチアレイ転送の同期タイムアウトの設定です。[並列処理 \[▶ 72\]](#)の項を参照してください。

出力パラメータ

```

VAR_OUTPUT
    bError           : BOOL;                           // TRUE if an error occurs. Reset by next metho
    d call.
    hrErrorCode       : HRESULT;                       // '< 0' = error; '> 0' = info; '0' = no error/
    info
    ipErrorMessage    : I_TcMessage := fbErrorMessage; // Shows detailed information about occurred er
    rors, warnings and more.
    nCntResults       : ULINT;                         // Counts outgoing results (MultiArrays were ca
    lculated and sent to transfer tray).
END_VAR

```

- **bError:** エラーが発生した場合の出力は、TRUEです。
- **hrErrorCode:** エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。
- **ipErrorMessage:** 現在の戻り値に関するより詳細な情報を含みます。[エラーの説明と情報 \[▶ 81\]](#)の項を参照してください。この特別なインターフェースポイントは、常に有効かつ割り当て済みであるよう、内部的に保護されています。

メソッド

Call():

このメソッドは、現在の入力データにアルゴリズムを適用するために、サイクルごとに呼び出されます。メソッドが新しい結果もエラーも示さない場合、ファンクションブロックは入力データを待機します。これは、解析チェーンのプロセスにおける規則的な動作です。

- **戻り値**：エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD Call : HRESULT
VAR_OUTPUT
    bNewResult    : BOOL;           // TRUE every time when outgoing MultiArray was calculated and sent to transfer tray.
    bError        : BOOL;           // TRUE if an error occurs.
    hrErrorCode    : HRESULT;       // '< 0' = error; '> 0' = info; '0' = no error/info
END_VAR
```

- **bError**:エラーが発生した場合の出力は、TRUEです。
- **hrErrorCode**:エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。この出力はメソッドの戻り値と同じです。



タイムアウトが発生した場合、または結果を格納するマルチアレイバッファがない場合、入力データも結果データも失われません。次の呼び出しに転送されます。

Configure() :

このメソッドを呼び出すことで、実行時にパラメータfCenterを調整できます。ここで、周波数fにおけるFFTインデックスが $k = f * fSampleRate / nFFT\_Length$ に計算されます。f が周波数分解能fSampleRate / nFFT_Length の整数倍でない場合、スペクトル成分は連続する2つのスペクトル値に分割されます。

対応するPLC配列は以下のように定義してください。Configure()メソッドは、異なる引数セットを持つ新しい設定にも使用できます。

- **戻り値**：エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD Configure : HRESULT
VAR_INPUT
    pArg      : PVOID;           // pointer to array of arguments
    nArgSize  : UDINT;           // size of arguments buffer in bytes
END_VAR
```

入力バッファは、以下の定義（入力形状）のいずれかに紐づいています。変数パラメータは、ファンクションブロック入力stInitPars の一部です。

バージョン	入力バッファ（マルチアレイ 入カストリーム）要素タイプ、次元数、次元サイズ
全てのチャンネルで同一設定	LREAL, 1, 1
チャンネル別の設定 (nChannels > 1)	LREAL, 1, nChannels

```
METHOD Init : HRESULT
VAR_INPUT
    stInitPars    : ST_CM_ZoomFFT_InitPars;           // init parameter
    nOwnID        : UDINT;                           // ID for this FB instance
    aDestIDs      : ARRAY[1..cCMA_MaxDest] OF UDINT; // IDs of destinations for output
    nResultBuffers : UDINT := 4;                     // number of MultiArrays which should be initialized for results (0 for no initialization)
END_VAR
```

Init():

このメソッドは通常、状態監視アプリケーションでは必要ありません。これはファンクションブロック初期化の代替手段を提供するものです。Init() メソッドは、PLCの初期化フェーズ中にのみ呼び出すことができます。実行時には使用できません。メソッドFB_init または、属性'call_after_init'の使用を参照してください（TwinCAT PLCリファレンスを参照）。さらに、これはファンクションブロックのカプセル化を容易にします。

Init()メソッドを使用して初期化を行う場合、ファンクションブロックインスタンスの入力パラメータを宣言で代入することはできません。

- **戻り値** : エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。
- **stInitPars:ST_CM_ZoomFFT_InitPars [▶ 303]** 型の初期化パラメータを持つファンクションブロック固有の構造体。パラメータは、上記の入出力バッファの定義と紐づいていなければなりません。
- **nOwnID**:一意のIDでファンクションブロックインスタンスを識別します。これは常に0より大きくなければなりません。この目的のためには、列挙を定義するアプローチが確立されています。
- **aDestIDs**:転送先のIDを指定することで、結果の転送先を定義します。出力バッファの定義（上記を参照）は、選択された各転送先の入力バッファの定義と紐づいていなければなりません。
- **nResultBuffers**:このファンクションブロックは、指定された数のマルチアレイバッファにより転送ストリームを初期化します。デフォルト値は4です。

PassInputs() :

API PLC リファレンス [▶ 79]で説明されているとおり、FB_CMA_Source インスタンスが呼び出され、信号データがターゲットブロックに転送される限り、解析チェーンの以降の全てのファンクションブロックを周期的に呼び出す必要があります。

アルゴリズムを一定時間実行しない方が有用な場合があります。例えば、いくつかのアルゴリズムは、事前に教示や設定をした後でのみ実行すべきです。ファンクションブロックは周期的に呼び出す必要がありますが、ファンクションブロックに到着したデータが通信リング [▶ 72]内で転送されれば十分です。これは、Call() メソッドの代わりにPassInputs() メソッドを使用して行われます。アルゴリズム自体はここでは呼び出されず、したがって結果も計算されず、出力バッファも生成されません。

- **戻り値** : エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD PassInputs : HRESULT
VAR_INPUT
END_VAR
```

GetChannelErrors() :

このメソッドは、複数のチャンネル（nChannels > 1）を処理する際に、チャンネル固有の戻り値のリストを照会できるようにします。呼び出しは、ファンクションブロックの戻り値が eCM_InfRTTime_AmbiguousChannelResults または eCM_ErrRTTime_ErrornousChannelResults のいずれかに該当する場合に有用です。

- **戻り値**: エラーコード一覧の読み込み処理に関する情報です。この値は、クエリが成功した場合は TRUE に、そうでない場合は FALSE に設定されます。

```
METHOD GetChannelErrors : BOOL
VAR_IN_OUT
aChannelErrors : ARRAY[*] OF HRESULT;
END_VAR
```

- **aChannelErrors**:HRESULT 型の長さnChannelsのエラーリストです。

類似したファンクションブロック

ファンクションブロックFB_CMA_RealFFT [▶ 203]は、実数値入力信号に対して高速フーリエ変換を計算します。

ファンクションブロックST_CM_SparseSpectrum_InitPars [▶ 299] は、個別に設定可能なマグニチュードとスペクトル値を計算します。

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM, Tc3_CM_Base

5.1.43 FB_CMA_MovingMean

実数値時系列データの移動平均値を計算します。

このファンクションブロックは、入力信号を時系列データとして扱い、必要に応じて複数の独立したチャンネルとともに処理します。移動（算術）平均値は、式にしたがって各チャンネルについて計算されます。

$$\bar{x} = \frac{1}{N} \sum_{n=0}^{N-1} x[n]$$

各チャンネルごとに固定数 (nInLength) のサンプルを、各サイクルのサンプル数に追加できます（入出力を参照）。平均値は常にサンプルごとに計算されます。

メモリ特性

サンプル数Nは、新しいデータセットが入力されるたびに自動的に増加し、現在の結果を計算します。つまり、ファンクションブロックがインスタンス化されてからの全ての入力値を使用します。サンプル数のゼロリセット（FBの内部メモリの削除）は、ResetData() メソッド、またはCallEx() メソッドが使用される場合、変数bResetData によって提供されます。

NaN（非数）の発生

入力値の数が特定のチャンネルの結果を計算するのに十分でない場合、または分散がゼロの場合、そのチャンネルに対してIEEE 754に従ってNaN（非数）の値が返されます。このエラー値があるかどうかは、ファンクションLrealIsNaN()を使用して確認できます。原因として、これまでのところ十分な入力データが転送されていないか、個々のチャンネルの入力値としてNaN値しか転送されていないことが考えられます。例えば、断線やスイッチングエラーによってセンサデータが転送されなかった場合など、時系列データの値が一定であれば、分散がゼロになることがあります。

入力値セットに特別な定数 NaN が含まれる場合、この時間ステップでは、チャンネルの統計に値は追加されません。つまり、NaN は欠損値の指標として扱われます。



NaN値の取り扱い

NaN値につながる上記の状況を除外できない、または安全に無視できない場合、アプリケーションプログラムはこれらのエラー値を適切に処理できるようにしてください。

マルチチャンネル入力データ処理時の動作

複数のチャンネル (nChannels > 1) を処理する場合、各チャンネルが異なる戻り値になる可能性があります。この場合、戻り値はファンクションブロックで個別に問い合わせできます。全てのチャンネルではなく、1つ以上のチャンネルの結果が許容されない場合、ファンクションブロックの値はeCM_InfRTime_AmbiguousChannelResultsに対応します。全てのチャンネルの結果が許容されない場合、ファンクションブロック上の値はeCM_ErrRTime_ErrornousChannelResults に対応します。

全チャンネルの戻り値のリストは、GetChannelErrors()のメソッドで照会できます。

複数のサブチャンネル(nSubChannels > 0)を処理する場合、入出力データのフォーマットには特に注意が必要です。入力データが上流のファンクションブロックによるマルチチャンネルの結果で構成されている場合、nChannels の値を採用する必要があります。この場合、パラメータnSubChannels を使用して追加の設定を行います。

例：マルチチャンネルスペクトル（FB_CMA_MagnitudeSpectrumなど）の周波数チャンネルの統計的考察（FB_CMA_Quantilesなど）では、nChannels の値は、入力信号の数と同じである必要があります。サブチャンネルの数nSubChannels は、スペクトルの長さに対応します。

サンプルの実装

実装用のサンプルは以下のリンクから入手できます：[移動平均 \[▶ 364\]](#)

入力と出力

入力バッファと出力バッファは、以下の定義（形状）に該当します。変数パラメータは、ファンクションブロック入力stInitParsの一部です。

バージョン	入力バッファ（マルチアレイ入力ストリーム） 要素タイプ、次元数、次元サイズ	出力バッファ（マルチアレイ出力ストリーム） 要素タイプ、次元数、次元サイズ
標準バージョン (nSubChannels = 0)	LREAL, 1, nInLength	LREAL, 1, nInLength
複数データセット用標準バージョン (nSubChannels = 0)	LREAL, 2, nChannels x nInLength	LREAL, 2, nChannels x nInLength
マルチチャンネルバージョン (nSubChannels > 0)	LREAL, 2, nChannels x nSubChannels	LREAL, 2, nChannels x nSubChannels
複数データセット用マルチチャンネルバージョン (nSubChannels > 0)	LREAL, 3, nChannels x nSubChannels x nInLength	LREAL, 3, nChannels x nSubChannels x nInLength

*:この次元の長さは任意に選択できるため、アプリケーションや先行のアルゴリズムの出力バッファに合わせる必要があります。

入力パラメータ

このファンクションブロックの入力パラメータは初期化パラメータを表し、FBインスタンスの宣言で割り当て済みである必要があります。(または、Init()メソッド) 一度しか割り当てできません。実行中の変更はできません。

```
VAR_INPUT
    stInitPars      : ST_CM_MovingMoments_InitPars;    // init parameter
    nOwnID          : UDINT;                          // ID for this FB instance
    aDestIDs: ARRAY[1..cCMA_MaxDest] OF UDINT;        // IDs of destinations for output
    nResultBuffers  : UDINT := 4;                     // number of MultiArrays which should be initialized for results (0 for no initialization)
    tTransferTimeout : LTIME := LTIME#500US;          // timeout checking off during access to inter-task FIFOs
END_VAR
```

- **stInitPars:** [ST_CM_MovingMoments_InitPars \[▶ 291\]](#) 型の初期化パラメータを持つ、ファンクションブロック固有の構造体。パラメータは、上記の入出力バッファの定義と紐づいていなければなりません。
- **nOwnID:**一意のIDでファンクションブロックインスタンスを識別します。これは常に0より大きくなければなりません。この目的のためには、列挙を定義するアプローチが確立されています。
- **aDestIDs:**転送先のIDを指定することで、結果の転送先を定義します。出力バッファの定義（上記を参照）は、選択された各転送先の入力バッファの定義と紐づいていなければなりません。
- **nResultBuffers:**このファンクションブロックは、指定された数のマルチアレイバッファにより転送トレイストリームを初期化します。デフォルト値は4です。
- **tTransferTimeout:**内部マルチアレイ転送の同期タイムアウトの設定です。[並列処理 \[▶ 72\]](#)の項を参照してください。

出力パラメータ

```
VAR_OUTPUT
    bError          : BOOL;                          // TRUE if an error occurs. Reset by next method call.
    hrErrorCode      : HRESULT;                       // '< 0' = error; '> 0' = info; '0' = no error/info
    ipErrorMessage  : I_TcMessage := fbErrorMessage; // Shows detailed information about occurred errors, warnings and more.
    nCntResults      : ULINT;                         // Counts outgoing results (MultiArrays were calculated and sent to transfer tray).
END_VAR
```

- **bError**:エラーが発生した場合の出力は、TRUEです。
- **hrErrorCode**:エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。
- **ipErrorMessage**:現在の戻り値に関するより詳細な情報を含みます。[エラーの説明と情報 \[▶ 81\]](#)の項を参照してください。この特別なインターフェイスポインタは、常に有効かつ割り当て済みであるよう、内部的に保護されています。

Call():

このメソッドは、現在の入力データにアルゴリズムを適用するために、サイクルごとに呼び出されます。メソッドが新しい結果もエラーも示さない場合、ファンクションブロックは入力データを待機します。これは、解析チェーンのプロセスにおける規則的な動作です。

- **戻り値**:エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD Call : HRESULT
VAR_OUTPUT
    bNewResult      : BOOL;           // TRUE every time when outgoing MultiArray was calculated and sent to transfer tray.
    bError          : BOOL;           // TRUE if an error occurs.
    hrErrorCode      : HRESULT;       // '< 0' = error; '> 0' = info; '0' = no error/info
END_VAR
```

- **bError**:エラーが発生した場合の出力は、TRUEです。
- **hrErrorCode**:エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。この出力はメソッドの戻り値と同じです。



タイムアウトが発生した場合、または結果を格納するマルチアレイバッファがない場合、入力データも結果データも失われません。次の呼び出しに転送されます。

Init():

このメソッドは通常、状態監視アプリケーションでは必要ありません。これはファンクションブロック初期化の代替手段を提供するものです。Init() メソッドは、PLCの初期化フェーズ中にのみ呼び出すことができます。実行時には使用できません。メソッドFB_init または、属性'call_after_init' の使用を参照してください（TwinCAT PLCリファレンスを参照）。さらに、これはファンクションブロックのカプセル化を容易にします。

Init()メソッドを使用して初期化を行う場合、ファンクションブロックインスタンスの入力パラメータを宣言で代入することはできません。

- **戻り値**:エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD Init : HRESULT
VAR_INPUT
    stInitPars      : ST_CM_MovingMoments_InitPars;           // init parameter
    nOwnID          : UDINT;                                   // ID for this FB instance
    aDestIDs        : ARRAY[1..cCMA_MaxDest] OF UDINT;         // IDs of destinations for output
    nResultBuffers  : UDINT := 4;                             // number of MultiArrays which should be initialized for results (0 for no initialization)
END_VAR
```

- **stInitPars**:[ST_CM_MovingMoments_InitPars \[▶ 291\]](#) 型の初期化パラメータを持つ、ファンクションブロック固有の構造体。パラメータは、上記の入出力バッファの定義と紐づいていなければなりません。
- **nOwnID**:一意のIDでファンクションブロックインスタンスを識別します。これは常に0より大きくなければなりません。この目的のためには、列挙を定義するアプローチが確立されています。
- **aDestIDs**:転送先のIDを指定することで、結果の転送先を定義します。出力バッファの定義（上記を参照）は、選択された各転送先の入力バッファの定義と紐づいていなければなりません。
- **nResultBuffers**:このファンクションブロックは、指定された数のマルチアレイバッファにより転送ストリームを初期化します。デフォルト値は4です。

ResetData() :

このメソッドは、既に追加された全てのデータレコードを削除します。ファンクションブロックのメモリ特性を参照してください。ResetData() の後、Call() メソッドが再度呼び出された場合、有効な結果を計算するために内部メモリを補充する必要があります。

- **戻り値**：エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD ResetData : HRESULT
VAR_INPUT
END_VAR
```

PassInputs() :

API PLC リファレンス [▶ 79]で説明されているとおり、FB_CMA_Source インスタンスが呼び出され、信号データがターゲットブロックに転送される限り、解析チェーンの以降の全てのファンクションブロックを周期的に呼び出す必要があります。

アルゴリズムを一定時間実行しない方が有用な場合があります。例えば、いくつかのアルゴリズムは、事前に教示や設定をした後でのみ実行すべきです。ファンクションブロックは周期的に呼び出す必要がありますが、ファンクションブロックに到着したデータが通信リング [▶ 72]内で転送されれば十分です。これは、Call() メソッドの代わりにPassInputs() メソッドを使用して行われます。アルゴリズム自体はここでは呼び出されず、したがって結果も計算されず、出力バッファも生成されません。

- **戻り値**：エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD PassInputs : HRESULT
VAR_INPUT
END_VAR
```

GetChannelErrors() :

このメソッドは、複数のチャンネル (nChannels > 1) を処理する際に、チャンネル固有の戻り値のリストを照会できるようにします。呼び出しは、ファンクションブロックの戻り値が eCM_InfRTTime_AmbiguousChannelResults または eCM_ErrRTTime_ErrornousChannelResults のいずれかに該当する場合に有用です。

- **戻り値**：エラーコード一覧の読み込み処理に関する情報です。この値は、クエリが成功した場合は TRUE に、そうでない場合は FALSE に設定されます。

```
METHOD GetChannelErrors : BOOL
VAR_IN_OUT
aChannelErrors : ARRAY[*] OF HRESULT;
END_VAR
```

- aChannelErrors: HRESULT 型の長さ nChannels のエラーリストです。

類似したファンクションブロック

ファンクションブロック [FB_CMA_MovingStandardDeviation \[▶ 258\]](#) は、入力値の移動標準偏差を計算します。

ファンクションブロック [FB_CMA_MovingSkew \[▶ 262\]](#) は、入力値の移動歪度を計算します。

ファンクションブロック [FB_CMA_MovingExcess \[▶ 266\]](#) は、入力値の移動超過を計算します。

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM, Tc3_CM_Base



CM 3.1 では一部の機能がすでに使用可能です。[互換性 \[▶ 61\]](#)の項を参照してください。

5.1.44 FB_CMA_MovingStandardDeviation

移動標準偏差は、シングルチャンネルおよびマルチチャンネルの実数値時系列データの移動標準偏差を計算します。

このファンクションブロックは、入力信号を時系列データとして扱い、必要に応じて複数の独立したチャンネルとともに処理します。移動標準偏差は、以下の式にしたがって各チャンネルについて計算されます。

$$s = \sqrt{\frac{1}{N-1} \sum_{n=0}^{N-1} (x[n] - \bar{x})^2}$$

各チャンネルごとに固定数 (nInLength) のサンプルを、各サイクルのサンプル数に追加できます（入出力を参照）。標準偏差は常にサンプルごとに計算されます。

追加のコメント

初期結果を計算するためには、2つの値が必要です。入力値が桁数で大きく異なる場合、結果が不正確になる可能性があります。

メモリ特性

サンプル数Nは、新しいデータセットが入力されるたびに自動的に増加し、現在の結果を計算します。つまり、ファンクションブロックがインスタンス化されてからの全ての入力値を使用します。サンプル数のゼロリセット（FBの内部メモリの削除）は、ResetData() メソッド、またはCallEx() メソッドが使用される場合、変数bResetData によって提供されます。

NaN（非数）の発生

入力値の数が特定のチャンネルの結果を計算するのに十分でない場合、または分散がゼロの場合、そのチャンネルに対してIEEE 754に従ってNaN（非数）の値が返されます。このエラー値があるかどうかは、ファンクションLrealIsNaN()を使用して確認できます。原因として、これまでのところ十分な入力データが転送されていないか、個々のチャンネルの入力値としてNaN値しか転送されていないことが考えられます。例えば、断線やスイッチングエラーによってセンサデータが転送されなかった場合など、時系列データの値が一定であれば、分散がゼロになることがあります。

入力値セットに特別な定数 NaN が含まれる場合、この時間ステップでは、チャンネルの統計に値は追加されません。つまり、NaN は欠損値の指標として扱われます。

● NaN値の取り扱い



NaN値につながる上記の状況を除外できない、または安全に無視できない場合、アプリケーションプログラムはこれらのエラー値を適切に処理できるようにしてください。

マルチチャンネル入力データ処理時の動作

複数のチャンネル (nChannels > 1) を処理する場合、各チャンネルが異なる戻り値になる可能性があります。この場合、戻り値はファンクションブロックで個別に問い合わせできます。全てのチャンネルではなく、1つ以上のチャンネルの結果が許容されない場合、ファンクションブロックの値は eCM_InfRTime_AmbiguousChannelResults に対応します。全てのチャンネルの結果が許容されない場合、ファンクションブロック上の値は eCM_ErrRTime_ErrornousChannelResults に対応します。

全チャンネルの戻り値のリストは、GetChannelErrors()のメソッドで照会できます。

複数のサブチャンネル (nSubChannels > 0) を処理する場合、入出力データのフォーマットには特に注意が必要です。入力データが上流のファンクションブロックによるマルチチャンネルの結果で構成されている場合、nChannels の値を採用する必要があります。この場合、パラメータ nSubChannels を使用して追加の設定を行います。

例：マルチチャンネルスペクトル (FB_CMA_MagnitudeSpectrumなど) の周波数チャンネルの統計的考察 (FB_CMA_Quantilesなど) では、nChannels の値は、入力信号の数と同じである必要があります。サブチャンネルの数 nSubChannels は、スペクトルの長さに対応します。

サンプルの実装

実装用のサンプルは以下のリンクから入手できます：[移動平均 \[▶ 364\]](#)

入力と出力

入力バッファと出力バッファは、以下の定義（形状）に該当します。変数パラメータは、ファンクションブロック入力stInitParsの一部です。

バージョン	入力バッファ（マルチアレイ入力ストリーム） 要素タイプ、次元数、次元サイズ	出力バッファ（マルチアレイ出力ストリーム） 要素タイプ、次元数、次元サイズ
標準バージョン (nSubChannels = 0)	LREAL, 1, nInLength	LREAL, 1, nInLength
複数データセット用標準バージョン (nSubChannels = 0)	LREAL, 2, nChannels x nInLength	LREAL, 2, nChannels x nInLength
マルチチャンネルバージョン (nSubChannels > 0)	LREAL, 2, nChannels x nSubChannels	LREAL, 2, nChannels x nSubChannels
複数データセット用マルチチャンネルバージョン (nSubChannels > 0)	LREAL, 3, nChannels x nSubChannels x nInLength	LREAL, 3, nChannels x nSubChannels x nInLength

*:この次元の長さは任意に選択できるため、アプリケーションや先行のアルゴリズムの出力バッファに合わせるができます。

入力パラメータ

このファンクションブロックの入力パラメータは初期化パラメータを表し、FBインスタンスの宣言で割り当て済みである必要があります。(または、Init()メソッド) 一度しか割り当てできません。実行中の変更はできません。

```
VAR_INPUT
    stInitPars      : ST_CM_MovingMoments_InitPars;    // init parameter
    nOwnID          : UDINT;                          // ID for this FB instance
    aDestIDs: ARRAY[1..cCMA_MaxDest] OF UDINT;        // IDs of destinations for output
    nResultBuffers  : UDINT := 4;                     // number of MultiArrays which should be initialized for results (0 for no initialization)
    tTransferTimeout : LTIME := LTIME#500US;          // timeout checking off during access to inter-task FIFOs
END_VAR
```

- **stInitPars:** [ST_CM_MovingMoments_InitPars \[▶ 291\]](#) 型の初期化パラメータを持つ、ファンクションブロック固有の構造体。パラメータは、上記の入出力バッファの定義と紐づいていなければなりません。
- **nOwnID:**一意のIDでファンクションブロックインスタンスを識別します。これは常に0より大きくなければなりません。この目的のためには、列挙を定義するアプローチが確立されています。
- **aDestIDs:**転送先のIDを指定することで、結果の転送先を定義します。出力バッファの定義（上記を参照）は、選択された各転送先の入力バッファの定義と紐づいていなければなりません。
- **nResultBuffers:**このファンクションブロックは、指定された数のマルチアレイバッファにより転送トレイストリームを初期化します。デフォルト値は4です。
- **tTransferTimeout:**内部マルチアレイ転送の同期タイムアウトの設定です。[並列処理 \[▶ 72\]](#)の項を参照してください。

出力パラメータ

```
VAR_OUTPUT
    bError          : BOOL;                          // TRUE if an error occurs. Reset by next method call.
    hrErrorCode      : HRESULT;                       // '< 0' = error; '> 0' = info; '0' = no error/info
    ipErrorMessage : I_TcMessage := fbErrorMessage; // Shows detailed information about occurred errors, warnings and more.
    nCntResults      : ULINT;                         // Counts outgoing results (MultiArrays were calculated and sent to transfer tray).
END_VAR
```

- **bError**:エラーが発生した場合の出力は、TRUEです。
- **hrErrorCode**:エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。
- **ipErrorMessage**:現在の戻り値に関するより詳細な情報を含みます。[エラーの説明と情報 \[▶ 81\]](#)の項を参照してください。この特別なインターフェイスポインタは、常に有効かつ割り当て済みであるよう、内部的に保護されています。

Call():

このメソッドは、現在の入力データにアルゴリズムを適用するために、サイクルごとに呼び出されます。メソッドが新しい結果もエラーも示さない場合、ファンクションブロックは入力データを待機します。これは、解析チェーンのプロセスにおける規則的な動作です。

- **戻り値**:エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD Call : HRESULT
VAR_OUTPUT
    bNewResult      : BOOL;           // TRUE every time when outgoing MultiArray was calculated and sent to transfer tray.
    bError          : BOOL;           // TRUE if an error occurs.
    hrErrorCode      : HRESULT;       // '< 0' = error; '> 0' = info; '0' = no error/info
END_VAR
```

- **bError**:エラーが発生した場合の出力は、TRUEです。
- **hrErrorCode**:エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。この出力はメソッドの戻り値と同じです。



タイムアウトが発生した場合、または結果を格納するマルチアレイバッファがない場合、入力データも結果データも失われません。次の呼び出しに転送されます。

Init():

このメソッドは通常、状態監視アプリケーションでは必要ありません。これはファンクションブロック初期化の代替手段を提供するものです。Init() メソッドは、PLCの初期化フェーズ中にのみ呼び出すことができます。実行時には使用できません。メソッドFB_init または、属性'call_after_init' の使用を参照してください（TwinCAT PLCリファレンスを参照）。さらに、これはファンクションブロックのカプセル化を容易にします。

Init()メソッドを使用して初期化を行う場合、ファンクションブロックインスタンスの入力パラメータを宣言で代入することはできません。

- **戻り値**:エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD Init : HRESULT
VAR_INPUT
    stInitPars      : ST_CM_MovingMoments_InitPars;           // init parameter
    nOwnID          : UDINT;                                   // ID for this FB instance
    aDestIDs        : ARRAY[1..cCMA_MaxDest] OF UDINT;         // IDs of destinations for output
    nResultBuffers  : UDINT := 4;                               // number of MultiArrays which should be initialized for results (0 for no initialization)
END_VAR
```

- **stInitPars**:[ST_CM_MovingMoments_InitPars \[▶ 291\]](#) 型の初期化パラメータを持つ、ファンクションブロック固有の構造体。パラメータは、上記の入出力バッファの定義と紐づいていなければなりません。
- **nOwnID**:一意のIDでファンクションブロックインスタンスを識別します。これは常に0より大きくなければなりません。この目的のためには、列挙を定義するアプローチが確立されています。
- **aDestIDs**:転送先のIDを指定することで、結果の転送先を定義します。出力バッファの定義（上記を参照）は、選択された各転送先の入力バッファの定義と紐づいていなければなりません。
- **nResultBuffers**:このファンクションブロックは、指定された数のマルチアレイバッファにより転送ストリームを初期化します。デフォルト値は4です。

ResetData() :

このメソッドは、既に追加された全てのデータレコードを削除します。ファンクションブロックのメモリ特性を参照してください。ResetData() の後、Call() メソッドが再度呼び出された場合、有効な結果を計算するために内部メモリを補充する必要があります。

- **戻り値**：エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD ResetData : HRESULT
VAR_INPUT
END_VAR
```

PassInputs() :

API PLC リファレンス [▶ 79]で説明されているとおり、FB_CMA_Source インスタンスが呼び出され、信号データがターゲットブロックに転送される限り、解析チェーンの以降の全てのファンクションブロックを周期的に呼び出す必要があります。

アルゴリズムを一定時間実行しない方が有用な場合があります。例えば、いくつかのアルゴリズムは、事前に教示や設定をした後でのみ実行すべきです。ファンクションブロックは周期的に呼び出す必要がありますが、ファンクションブロックに到着したデータが通信リング [▶ 72]内で転送されれば十分です。これは、Call() メソッドの代わりにPassInputs() メソッドを使用して行われます。アルゴリズム自体はここでは呼び出されず、したがって結果も計算されず、出力バッファも生成されません。

- **戻り値**：エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD PassInputs : HRESULT
VAR_INPUT
END_VAR
```

GetChannelErrors() :

このメソッドは、複数のチャンネル (nChannels > 1) を処理する際に、チャンネル固有の戻り値のリストを照会できるようにします。呼び出しは、ファンクションブロックの戻り値が eCM_InfRTTime_AmbiguousChannelResults または eCM_ErrRTTime_ErrornousChannelResults のいずれかに該当する場合に有用です。

- **戻り値**：エラーコード一覧の読み込み処理に関する情報です。この値は、クエリが成功した場合は TRUE に、そうでない場合は FALSE に設定されます。

```
METHOD GetChannelErrors : BOOL
VAR_IN_OUT
aChannelErrors : ARRAY[*] OF HRESULT;
END_VAR
```

- aChannelErrors: HRESULT 型の長さnChannelsのエラーリストです。

類似したファンクションブロック

ファンクションブロックFB_CMA_MovingMean [▶ 254] は、入力値の移動平均値を計算します。

ファンクションブロック FB_CMA_MovingSkew [▶ 262]は、入力値の移動歪度を計算します。

ファンクションブロックFB_CMA_MovingExcess [▶ 266] は、入力値の移動超過を計算します。

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM, Tc3_CM_Base



CM 3.1 では一部の機能がすでに使用可能です。[互換性 \[▶ 61\]](#)の項を参照してください。

5.1.45 FB_CMA_MovingSkew

シングルチャンネルおよびマルチチャンネルの実数値時系列データに対して移動歪度を計算します。

このファンクションブロックは、入力信号を時系列データとして扱い、必要に応じて複数の独立したチャンネルとともに処理します。経験的歪度は、次の式にしたがって各チャンネルで計算されます。

$$v = \frac{1}{N} \sum_{n=0}^{N-1} \left(\frac{x[n] - \bar{x}}{s} \right)^3$$

ここでsは移動標準偏差です。歪度は、サンプルの非対称性を定量的に表現します。これは、振動信号における衝撃（例えば、ローラーベアリングの局所的な損傷の上を転がるなど）を評価できます。局所的な損傷が必ずしも非対称な信号分布につながるわけではありませんが、計算された歪度は、尖度/過剰度よりも堅牢な基準です。

各チャンネルごとに固定数 (nInLength) のサンプルを、各サイクルのサンプル数に追加できます（入出力を参照）。平均値は常にサンプルごとに計算されます。

追加のコメント

初期結果を計算するためには、3つの値が必要です。さらに、標準偏差はゼロであってはなりません。入力値が桁数で大きく異なる場合、結果が不正確になる可能性があります。

メモリ特性

サンプル数Nは、新しいデータセットが入力されるたびに自動的に増加し、現在の結果を計算します。つまり、ファンクションブロックがインスタンス化されてからの全ての入力値を使用します。サンプル数のゼロリセット（FBの内部メモリの削除）は、ResetData() メソッド、またはCallEx() メソッドが使用される場合、変数bResetData によって提供されます。

NaN（非数）の発生

入力値の数が特定のチャンネルの結果を計算するのに十分でない場合、または分散がゼロの場合、そのチャンネルに対してIEEE 754に従ってNaN（非数）の値が返されます。このエラー値があるかどうかは、ファンクションLrealIsNaN()を使用して確認できます。原因として、これまでのところ十分な入力データが転送されていないか、個々のチャンネルの入力値としてNaN値しか転送されていないことが考えられます。例えば、断線やスイッチングエラーによってセンサデータが転送されなかった場合など、時系列データの値が一定であれば、分散がゼロになることがあります。

入力値セットに特別な定数 NaN が含まれる場合、この時間ステップでは、チャンネルの統計に値は追加されません。つまり、NaN は欠損値の指標として扱われます。

● NaN値の取り扱い

I NaN値につながる上記の状況を除外できない、または安全に無視できない場合、アプリケーションプログラムはこれらのエラー値を適切に処理できるようにしてください。

マルチチャンネル入力データ処理時の動作

複数のチャンネル (nChannels > 1) を処理する場合、各チャンネルが異なる戻り値になる可能性があります。この場合、戻り値はファンクションブロックで個別に問い合わせできます。全てのチャンネルではなく、1つ以上のチャンネルの結果が許容されない場合、ファンクションブロックの値はeCM_InfRTime_AmbiguousChannelResultsに対応します。全てのチャンネルの結果が許容されない場合、ファンクションブロック上の値はeCM_ErrRTime_ErrornousChannelResults に対応します。

全チャンネルの戻り値のリストは、GetChannelErrors()のメソッドで照会できます。

複数のサブチャンネル(nSubChannels > 0)を処理する場合、入出力データのフォーマットには特に注意が必要です。入力データが上流のファンクションブロックによるマルチチャンネルの結果で構成されている場合、nChannels の値を採用する必要があります。この場合、パラメータnSubChannels を使用して追加の設定を行います。

例：マルチチャンネルスペクトル（FB_CMA_MagnitudeSpectrumなど）の周波数チャンネルの統計的考察（FB_CMA_Quantilesなど）では、nChannels の値は、入力信号の数と同じである必要があります。サブチャンネルの数nSubChannels は、スペクトルの長さに対応します。

サンプルの実装

実装用のサンプルは以下のリンクから入手できます：[移動平均 \[▶ 364\]](#)

入力と出力

入力バッファと出力バッファは、以下の定義（形状）に該当します。変数パラメータは、ファンクションブロック入力stInitPars の一部です。

バージョン	入力バッファ（マルチアレイ入力ストリーム） 要素タイプ、次元数、次元サイズ	出力バッファ（マルチアレイ出力ストリーム） 要素タイプ、次元数、次元サイズ
標準バージョン (nSubChannels = 0)	LREAL, 1, nInLength	LREAL, 1, nInLength
複数データセット用標準バージョン (nSubChannels = 0)	LREAL, 2, nChannels x nInLength	LREAL, 2, nChannels x nInLength
マルチチャンネルバージョン (nSubChannels > 0)	LREAL, 2, nChannels x nSubChannels	LREAL, 2, nChannels x nSubChannels
複数データセット用マルチチャンネルバージョン (nSubChannels > 0)	LREAL, 3, nChannels x nSubChannels x nInLength	LREAL, 3, nChannels x nSubChannels x nInLength

*:この次元の長さは任意に選択できるため、アプリケーションや先行のアルゴリズムの出力バッファに合わせるができます。

入力パラメータ

このファンクションブロックの入力パラメータは初期化パラメータを表し、FBインスタンスの宣言で割り当て済みである必要があります。(または、Init()メソッド) 一度しか割り当てできません。実行中の変更はできません。

```
VAR_INPUT
    stInitPars      : ST_CM_MovingMoments_InitPars;           // init parameter
    nOwnID           : UDINT;                                   // ID for this FB instance
    aDestIDs: ARRAY[1..cCMA_MaxDest] OF UDINT;                // IDs of destinations for output
    nResultBuffers   : UDINT := 4;                             // number of MultiArrays which should be initialized for results (0 for no initialization)
    tTransferTimeout : LTIME := LTIME#500US;                  // timeout checking off during access to inter-task FIFOs
END_VAR
```

- stInitPars:ST_CM_MovingMoments_InitPars [▶ 291] 型の初期化パラメータを持つ、ファンクションブロック固有の構造体。パラメータは、上記の入出力バッファの定義と紐づいていなければなりません。
- nOwnID:一意のIDでファンクションブロックインスタンスを識別します。これは常に0より大きくなければなりません。この目的のためには、列挙を定義するアプローチが確立されています。
- aDestIDs:転送先のIDを指定することで、結果の転送先を定義します。出力バッファの定義（上記を参照）は、選択された各転送先の入力バッファの定義と紐づいていなければなりません。
- nResultBuffers:このファンクションブロックは、指定された数のマルチアレイバッファにより転送トレイストリームを初期化します。デフォルト値は4です。
- tTransferTimeout:内部マルチアレイ転送の同期タイムアウトの設定です。[並列処理 \[▶ 72\]](#)の項を参照してください。

出力パラメータ

```
VAR_OUTPUT
    bError           : BOOL;                                   // TRUE if an error occurs. Reset by next method call.
    hrErrorCode       : HRESULT;                               // '< 0' = error; '> 0' = info; '0' = no error/
```

```
info
  ipErrorMessage : I_TcMessage := fbErrorMessage; // Shows detailed information about occurred errors, warnings and more.
  nCntResults    : ULINT;                          // Counts outgoing results (MultiArrays were calculated and sent to transfer tray).
END_VAR
```

- **bError:**エラーが発生した場合の出力は、TRUEです。
- **hrErrorCode:**エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。
- **ipErrorMessage:**現在の戻り値に関するより詳細な情報を含みます。[エラーの説明と情報 \[▶ 81\]](#)の項を参照してください。この特別なインターフェースポインタは、常に有効かつ割り当て済みであるよう、内部的に保護されています。

Call():

このメソッドは、現在の入力データにアルゴリズムを適用するために、サイクルごとに呼び出されます。メソッドが新しい結果もエラーも示さない場合、ファンクションブロックは入力データを待機します。これは、解析チェーンのプロセスにおける規則的な動作です。

- **戻り値:** エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD Call : HRESULT
VAR_OUTPUT
  bNewResult    : BOOL;          // TRUE every time when outgoing MultiArray was calculated and sent to transfer tray.
  bError        : BOOL;          // TRUE if an error occurs.
  hrErrorCode    : HRESULT;      // '< 0' = error; '> 0' = info; '0' = no error/info
END_VAR
```

- **bError:**エラーが発生した場合の出力は、TRUEです。
- **hrErrorCode:**エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。この出力はメソッドの戻り値と同じです。



タイムアウトが発生した場合、または結果を格納するマルチアレイバッファがない場合、入力データも結果データも失われません。次の呼び出しに転送されます。

Init():

このメソッドは通常、状態監視アプリケーションでは必要ありません。これはファンクションブロック初期化の代替手段を提供するものです。Init() メソッドは、PLCの初期化フェーズ中にのみ呼び出すことができます。実行時には使用できません。メソッドFB_init または、属性'call_after_init' の使用を参照してください（TwinCAT PLCリファレンスを参照）。さらに、これはファンクションブロックのカプセル化を容易にします。

Init()メソッドを使用して初期化を行う場合、ファンクションブロックインスタンスの入力パラメータを宣言で代入することはできません。

- **戻り値:** エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD Init : HRESULT
VAR_INPUT
  stInitPars    : ST_CM_MovingMoments_InitPars; // init parameter
  nOwnID        : UDINT;                        // ID for this FB instance
  aDestIDs      : ARRAY[1..cCMA_MaxDest] OF UDINT; // IDs of destinations for output
  nResultBuffers : UDINT := 4;                  // number of MultiArrays which should be initialized for results (0 for no initialization)
END_VAR
```

- **stInitPars:**[ST_CM_MovingMoments_InitPars \[▶ 291\]](#) 型の初期化パラメータを持つ、ファンクションブロック固有の構造体。パラメータは、上記の入出力バッファの定義と紐づいていなければなりません。
- **nOwnID:**一意のIDでファンクションブロックインスタンスを識別します。これは常に0より大きくなければなりません。この目的のためには、列挙を定義するアプローチが確立されています。
- **aDestIDs:**転送先のIDを指定することで、結果の転送先を定義します。出力バッファの定義（上記を参照）は、選択された各転送先の入力バッファの定義と紐づいていなければなりません。

- **nResultBuffers**: このファンクションブロックは、指定された数のマルチアレイバッファにより転送トレイストリームを初期化します。デフォルト値は4です。

ResetData() :

このメソッドは、既に追加された全てのデータレコードを削除します。ファンクションブロックのメモリ特性を参照してください。ResetData() の後、Call() メソッドが再度呼び出された場合、有効な結果を計算するために内部メモリを補充する必要があります。

- **戻り値**: エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD ResetData : HRESULT
VAR_INPUT
END_VAR
```

PassInputs() :

API PLC リファレンス [▶ 79]で説明されているとおり、FB_CMA_Source インスタンスが呼び出され、信号データがターゲットブロックに転送される限り、解析チェーンの以降の全てのファンクションブロックを周期的に呼び出す必要があります。

アルゴリズムを一定時間実行しない方が有用な場合があります。例えば、いくつかのアルゴリズムは、事前に教示や設定をした後でのみ実行すべきです。ファンクションブロックは周期的に呼び出す必要がありますが、ファンクションブロックに到着したデータが通信リング [▶ 72]内で転送されれば十分です。これは、Call() メソッドの代わりにPassInputs() メソッドを使用して行われます。アルゴリズム自体はここでは呼び出されず、したがって結果も計算されず、出力バッファも生成されません。

- **戻り値**: エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD PassInputs : HRESULT
VAR_INPUT
END_VAR
```

GetChannelErrors() :

このメソッドは、複数のチャンネル (nChannels > 1) を処理する際に、チャンネル固有の戻り値のリストを照会できるようにします。呼び出しは、ファンクションブロックの戻り値が eCM_InfRTTime_AmbiguousChannelResults または eCM_ErrRTTime_ErrornousChannelResults のいずれかに該当する場合に有用です。

- **戻り値**: エラーコード一覧の読み込み処理に関する情報です。この値は、クエリが成功した場合は TRUE に、そうでない場合は FALSE に設定されます。

```
METHOD GetChannelErrors : BOOL
VAR_IN_OUT
    aChannelErrors : ARRAY[*] OF HRESULT;
END_VAR
```

- **aChannelErrors**: HRESULT 型の長さ nChannels のエラーリストです。

類似したファンクションブロック

ファンクションブロック [FB_CMA_MovingMean \[▶ 254\]](#) は、入力値の移動平均値を計算します。

ファンクションブロック [FB_CMA_MovingStandardDeviation \[▶ 258\]](#) は、入力値の移動標準偏差を計算します。

ファンクションブロック [FB_CMA_MovingExcess \[▶ 266\]](#) は、入力値の移動超過を計算します。

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM, Tc3_CM_Base



CM 3.1 では一部の機能がすでに使用可能です。[互換性 \[▶ 61\]](#)の項を参照してください。

5.1.46 FB_CMA_MovingExcess

実数値時系列データの移動超過を計算します。

このファンクションブロックは、入力信号を時系列データとして扱い、必要に応じて複数の独立したチャンネルとともに処理します。経験的超過値は、式に従って各チャンネルごとに計算されます。

$$E = \frac{1}{N} \sum_{n=0}^{N-1} \left(\frac{x[n] - \bar{x}}{s} \right)^4 - 3$$

ここでsは移動標準偏差です。超過値は、経験的尖度から3を引いた値です（ここで、3は正規分布の尖度に該当します）。その結果、超過値の解釈となります：

Exceed > 0: 正規分布よりも尖った分布。サンプル内で頻繁にピークが現れることを示しています。

Exceed < 0: 正規分布に比べて平坦な分布。

超過値は、例えば、ローラーベアリングの局所的な損傷の上を転がる際に発生する振動信号のショックを評価できます。

各チャンネルごとに固定数 (nInLength) のサンプルを、各サイクルのサンプル数に追加できます（入出力を参照）。平均値は常にサンプルごとに計算されます。

追加のコメント

初期結果を計算するためには、4つの値が必要です。さらに、標準偏差はゼロであってはなりません。入力値が桁数で大きく異なる場合、結果が不正確になる可能性があります。

メモリ特性

サンプル数Nは、新しいデータセットが入力されるたびに自動的に増加し、現在の結果を計算します。つまり、ファンクションブロックがインスタンス化されてからの全ての入力値を使用します。サンプル数のゼロリセット（FBの内部メモリの削除）は、ResetData() メソッド、またはCallEx() メソッドが使用される場合、変数bResetData によって提供されます。

NaN（非数）の発生

入力値の数が特定のチャンネルの結果を計算するのに十分でない場合、または分散がゼロの場合、そのチャンネルに対してIEEE 754に従ってNaN（非数）の値が返されます。このエラー値があるかどうかは、ファンクションLrealIsNaN()を使用して確認できます。原因として、これまでのところ十分な入力データが転送されていないか、個々のチャンネルの入力値としてNaN値しか転送されていないことが考えられます。例えば、断線やスイッチングエラーによってセンサデータが転送されなかった場合など、時系列データの値が一定であれば、分散がゼロになることがあります。

入力値セットに特別な定数 NaN が含まれる場合、この時間ステップでは、チャンネルの統計に値は追加されません。つまり、NaN は欠損値の指標として扱われます。

● NaN値の取り扱い

i NaN値につながる上記の状況を除外できない、または安全に無視できない場合、アプリケーションプログラムはこれらのエラー値を適切に処理できるようにしてください。

マルチチャンネル入力データ処理時の動作

複数のチャンネル (nChannels > 1) を処理する場合、各チャンネルが異なる戻り値になる可能性があります。この場合、戻り値はファンクションブロックで個別に問い合わせできます。全てのチャンネルではなく、1つ以上のチャンネルの結果が許容されない場合、ファンクションブロックの値はeCM_InfRTIME_AmbiguousChannelResultsに対応します。全てのチャンネルの結果が許容されない場合、ファンクションブロック上の値はeCM_ErrRTIME_ErrornousChannelResults に対応します。

全チャンネルの戻り値のリストは、GetChannelErrors()のメソッドで照会できます。

複数のサブチャンネル($nSubChannels > 0$)を処理する場合、入出力データのフォーマットには特に注意が必要です。入力データが上流のファンクションブロックによるマルチチャンネルの結果で構成されている場合、 $nChannels$ の値を採用する必要があります。この場合、パラメータ $nSubChannels$ を使用して追加の設定を行います。

例：マルチチャンネルスペクトル（FB_CMA_MagnitudeSpectrumなど）の周波数チャンネルの統計的考察（FB_CMA_Quantilesなど）では、 $nChannels$ の値は、入力信号の数と同じである必要があります。サブチャンネルの数 $nSubChannels$ は、スペクトルの長さに対応します。

サンプルの実装

実装用のサンプルは以下のリンクから入手できます：[移動平均 \[▶ 364\]](#)

入力と出力

入力バッファと出力バッファは、以下の定義（形状）に該当します。変数パラメータは、ファンクションブロック入力 $stInitPars$ の一部です。

バージョン	入力バッファ（マルチアレイ入力ストリーム） 要素タイプ、次元数、次元サイズ	出力バッファ（マルチアレイ出力ストリーム） 要素タイプ、次元数、次元サイズ
標準バージョン ($nSubChannels = 0$)	LREAL, 1, $nInLength$	LREAL, 1, $nInLength$
複数データセット用標準バージョン ($nSubChannels = 0$)	LREAL, 2, $nChannels \times nInLength$	LREAL, 2, $nChannels \times nInLength$
マルチチャンネルバージョン ($nSubChannels > 0$)	LREAL, 2, $nChannels \times nSubChannels$	LREAL, 2, $nChannels \times nSubChannels$
複数データセット用マルチチャンネルバージョン ($nSubChannels > 0$)	LREAL, 3, $nChannels \times nSubChannels \times nInLength$	LREAL, 3, $nChannels \times nSubChannels \times nInLength$

*:この次元の長さは任意に選択できるため、アプリケーションや先行のアルゴリズムの出力バッファに合わせるができます。

入力パラメータ

このファンクションブロックの入力パラメータは初期化パラメータを表し、FBインスタンスの宣言で割り当て済みである必要があります。（または、Init()メソッド）一度しか割り当てできません。実行中の変更はできません。

```
VAR_INPUT
    stInitPars      : ST_CM_MovingMoments_InitPars;    // init parameter
    nOwnID          : UDINT;                          // ID for this FB instance
    aDestIDs: ARRAY[1..cCMA_MaxDest] OF UDINT;        // IDs of destinations for output
    nResultBuffers  : UDINT := 4;                      // number of MultiArrays which should be initialized for results (0 for no initialization)
    tTransferTimeout : LTIME := LTIME#500US;          // timeout checking off during access to inter-task FIFOs
END_VAR
```

- **stInitPars:** ST_CM_MovingMoments_InitPars [▶ 291] 型の初期化パラメータを持つ、ファンクションブロック固有の構造体。パラメータは、上記の入出力バッファの定義と紐づいていなければなりません。
- **nOwnID:** 一意のIDでファンクションブロックインスタンスを識別します。これは常に0より大きくなければなりません。この目的のためには、列挙を定義するアプローチが確立されています。
- **aDestIDs:** 転送先のIDを指定することで、結果の転送先を定義します。出力バッファの定義（上記を参照）は、選択された各転送先の入力バッファの定義と紐づいていなければなりません。
- **nResultBuffers:** このファンクションブロックは、指定された数のマルチアレイバッファにより転送トレイストリームを初期化します。デフォルト値は4です。
- **tTransferTimeout:** 内部マルチアレイ転送の同期タイムアウトの設定です。[並列処理 \[▶ 72\]](#)の項を参照してください。

出力パラメータ

```
VAR_OUTPUT
    bError      : BOOL;                // TRUE if an error occurs. Reset by next method call.
    hrErrorCode  : HRESULT;            // '< 0' = error; '> 0' = info; '0' = no error/info
    ipErrorMessage : I_TcMessage := fbErrorMessage; // Shows detailed information about occurred errors, warnings and more.
    nCntResults   : ULINT;            // Counts outgoing results (MultiArrays were calculated and sent to transfer tray).
END_VAR
```

- **bError:**エラーが発生した場合の出力は、TRUEです。
- **hrErrorCode:**エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。
- **ipErrorMessage:**現在の戻り値に関するより詳細な情報を含みます。[エラーの説明と情報 \[▶ 81\]](#)の項を参照してください。この特別なインターフェイスポイントは、常に有効かつ割り当て済みであるよう、内部的に保護されています。

Call():

このメソッドは、現在の入力データにアルゴリズムを適用するために、サイクルごとに呼び出されます。メソッドが新しい結果もエラーも示さない場合、ファンクションブロックは入力データを待機します。これは、解析チェーンのプロセスにおける規則的な動作です。

- **戻り値:**エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD Call : HRESULT
VAR_OUTPUT
    bNewResult   : BOOL;                // TRUE every time when outgoing MultiArray was calculated and sent to transfer tray.
    bError        : BOOL;                // TRUE if an error occurs.
    hrErrorCode   : HRESULT;            // '< 0' = error; '> 0' = info; '0' = no error/info
END_VAR
```

- **bError:**エラーが発生した場合の出力は、TRUEです。
- **hrErrorCode:**エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。この出力はメソッドの戻り値と同じです。



タイムアウトが発生した場合、または結果を格納するマルチアレイバッファがない場合、入力データも結果データも失われません。次の呼び出しに転送されます。

Init():

このメソッドは通常、状態監視アプリケーションでは必要ありません。これはファンクションブロック初期化の代替手段を提供するものです。Init() メソッドは、PLCの初期化フェーズ中にのみ呼び出すことができます。実行時には使用できません。メソッドFB_init または、属性'call_after_init'の使用を参照してください（TwinCAT PLCリファレンスを参照）。さらに、これはファンクションブロックのカプセル化を容易にします。

Init()メソッドを使用して初期化を行う場合、ファンクションブロックインスタンスの入力パラメータを宣言で代入することはできません。

- **戻り値:**エラーが発生した場合、HRESULTのエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD Init : HRESULT
VAR_INPUT
    stInitPars   : ST_CM_MovingMoments_InitPars; // init parameter
    nOwnID       : UDINT;                        // ID for this FB instance
    aDestIDs     : ARRAY[1..cCMA_MaxDest] OF UDINT; // IDs of destinations for output
    nResultBuffers: UDINT := 4;                  // number of MultiArrays which should be initialized for results (0 for no initialization)
END_VAR
```

- **stInitPars:**[ST_CM_MovingMoments_InitPars \[▶ 291\]](#) 型の初期化パラメータを持つ、ファンクションブロック固有の構造体。パラメータは、上記の入出力バッファの定義と紐づいていなければなりません。

- **nOwnID**:一意のIDでファンクションブロックインスタンスを識別します。これは常に0より大きくなければなりません。この目的のためには、列挙を定義するアプローチが確立されています。
- **aDestIDs**:転送先のIDを指定することで、結果の転送先を定義します。出力バッファの定義（上記を参照）は、選択された各転送先の入力バッファの定義と紐づいていなければなりません。
- **nResultBuffers**:このファンクションブロックは、指定された数のマルチアレイバッファにより転送トレイストリームを初期化します。デフォルト値は4です。

ResetData() :

このメソッドは、既に追加された全てのデータレコードを削除します。ファンクションブロックのメモリ特性を参照してください。ResetData() の後、Call() メソッドが再度呼び出された場合、有効な結果を計算するために内部メモリを補充する必要があります。

- **戻り値** : エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD ResetData : HRESULT
VAR_INPUT
END_VAR
```

PassInputs() :

API PLC リファレンス [▶ 79]で説明されているとおり、FB_CMA_Source インスタンスが呼び出され、信号データがターゲットブロックに転送される限り、解析チェーンの以降の全てのファンクションブロックを周期的に呼び出す必要があります。

アルゴリズムを一定時間実行しない方が有用な場合があります。例えば、いくつかのアルゴリズムは、事前に教示や設定をした後でのみ実行すべきです。ファンクションブロックは周期的に呼び出す必要がありますが、ファンクションブロックに到着したデータが[通信リング \[▶ 72\]](#)内で転送されれば十分です。これは、Call() メソッドの代わりにPassInputs() メソッドを使用して行われます。アルゴリズム自体はここでは呼び出されず、したがって結果も計算されず、出力バッファも生成されません。

- **戻り値** : エラーが発生した場合、HRESULT のエラーコードが出力されます。可能性のある値は、[エラーコード一覧 \[▶ 367\]](#)に記載されています。

```
METHOD PassInputs : HRESULT
VAR_INPUT
END_VAR
```

GetChannelErrors() :

このメソッドは、複数のチャンネル（nChannels > 1）を処理する際に、チャンネル固有の戻り値のリストを照会できるようにします。呼び出しは、ファンクションブロックの戻り値がeCM_InfRTTime_AmbiguousChannelResults またはeCM_ErrRTTime_ErrornousChannelResults のいずれかに該当する場合に有用です。

- **戻り値**: エラーコード一覧の読み込み処理に関する情報です。この値は、クエリが成功した場合はTRUE に、そうでない場合はFALSE に設定されます。

```
METHOD GetChannelErrors : BOOL
VAR_IN_OUT
aChannelErrors : ARRAY[*] OF HRESULT;
END_VAR
```

- **aChannelErrors**:HRESULT 型の長さnChannelsのエラーリストです。

類似したファンクションブロック

ファンクションブロックFB_CMA_MovingMean [▶ 254]は、入力以外の値の移動平均を計算します。

ファンクションブロックFB_CMA_MovingStandardDeviation [▶ 258] は、入力値の移動標準偏差を計算します。

ファンクションブロック FB_CMA_MovingSkew [▶ 262]は、入力値の移動歪度を計算します。

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM, Tc3_CM_Base



CM 3.1 では一部の機能がすでに使用可能です。[互換性 \[▶ 61\]](#)の項を参照してください。

5.2 ファンクション

5.2.1 F_MA_IsNaN

このファンクションは、NaN(非数)値の有無を検査し、値がNaNの場合にTRUEを返します。

```
FUNCTION F_MA_IsNaN : BOOL
VAR_INPUT
    fValue : LREAL;
END_VAR
```

詳細については、[NaN値の取り扱い \[▶ 71\]](#)の章を参照してください。



このファンクションは廃止されています。

代わりにLrealIsNaN()ファンクションを使用してください (Tc2_Uutilitiesライブラリ)。

要件

開発環境	ターゲットシステム	必要なPLCライブラリ
TwinCAT v3.1.4013	PCまたはCX (x86、x64)	Tc3_MultiArray

5.2.2 F_CM_CalculateRecommendedOverlap

このファンクションは、ウィンドウ関数の振幅平坦度とオーバーラップ相関に基づいて、推奨されるオーバーラップ値を計算します。[推奨オーバーラップの計算 \[▶ 21\]](#)の項を参照してください。

```
FUNCTION F_CM_CalculateRecommendedOverlap : UDINT
VAR_INPUT
    eWindowType      : E_CM_WindowType;      (* Window type. *)
    aWindowParameters : T_CM_WindowParameters; (* Window parameters for specific windows, e.g. FlatTop. *)
    nWindowLength     : UDINT;                (* Length of analysis window. *)
END_VAR
```

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM_Base

5.2.3 F_CM_CalculateWoehlerCurve

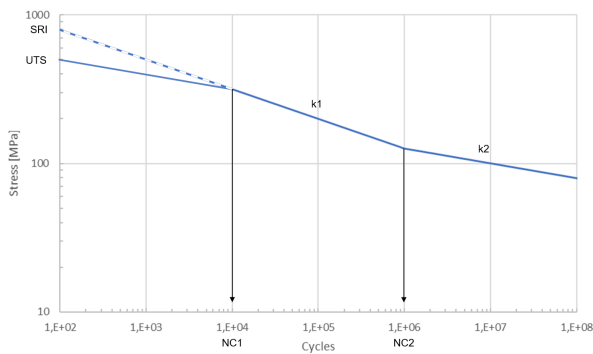
このファンクションは、材料パラメータを入力としてウェラー曲線を計算します。

```
FUNCTION F_CM_CalculateWoehlerCurve : HRESULT
VAR_INPUT
    fSRI      : LREAL;      (* Stress Range Intercept. *)
    fUTS      : LREAL;      (* Ultimate Tensile Strength OF the material. *)
    bUseUTSCorrection : BOOL; (* Apply UTS correction. *)
    fK1       : LREAL;      (* Gradient of Woehler Curve for region N = 1 *)
    nNC1      : ULINT;      (* Transition Point for UTS correction. *)
    fK2       : LREAL;      (* Gradient of Woehler Curve starting from nN *)
    nNC2      : ULINT;      (* Fatigue Transition Point between fK1 and fK2. *)
    fRRatio   : LREAL;      (* R-Ratio of the material test. *)
    fStepsize : LREAL;      (* Width of stress bins. *)
END_VAR
```

```

nCurveLength      : UDINT;          (* Length of curve. *)
pCurve            : POINTER TO POINTER TO LREAL; (* Pointer to array data. *)
END_VAR

```



要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM_Base

5.2.4 F_CM_CalculateWindow

このファンクションは、指定された入力パラメータに基づいてウィンドウ関数を計算します。[ウィンドウ関数 \[▶ 19\]](#)の項を参照してください。

```

FUNCTION F_CM_CalculateWindow : HRESULT
VAR_INPUT
    eWindowType      : E_CM_WindowType;          (* Window type. *)
    aWindowParameters : T_CM_WindowParameters;    (* Window parameters for specific windows, e.g.
    . FlatTop. *)
    nWindowLength     : UDINT;                    (* Length of analysis window. *)
    pWindow           : POINTER TO POINTER TO LREAL; (* Pointer to array data. *)
END_VAR

```

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM_Base

5.2.5 F_CM_ApplySpectralScaling

このファンクションはスペクトル値のスケーリングを行います。[スペクトルのスケーリング \[▶ 25\]](#)の項を参照してください。

```

FUNCTION F_CM_ApplySpectralScaling : HRESULT
VAR_INPUT
    eScalingType      : E_CM_ScalingType;          (* Scaling type. *)
    nFFT_Length       : UDINT;                    (* Length of FFT. *)
    nOrder            : UDINT;                    (* Order of spectrum: 1 = magnitude, 2 = power. *)
    fSampleRate       : LREAL;                    (* Sample rate in Hertz. *)
    bTransformToDecibel : BOOL;                    (* Transform to decibel. *)
    fDecibelThreshold : LREAL;                    (* Minimum argument of decadic logarithm for
    64-bit IEEE 754 arithmetic. *)
    nWindowLength     : UDINT;                    (* Length of analysis window. *)
    pWindow           : POINTER TO POINTER TO LREAL; (* Pointer to window data. *)
    nResultLength     : UDINT;                    (* Length of spectrum, i.e. nFFT_Length or nF
    FT_Length / 2 + 1. *)
    pSpectrum         : POINTER TO POINTER TO LREAL; (* Pointer to spectrum. *)
END_VAR

```


要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM_Base

5.3 データ型

5.3.1 E_CM_ComplexDataHandling

```

TYPE E_CM_ComplexDataHandling : (
    eCM_CplxReal      := 0,    (* Real part of complex data. *)
    eCM_CplxImag      := 1,    (* Imaginary part of complex data. *)
    eCM_CplxAbs       := 2,    (* Absolute value of complex data. *)
    eCM_CplxPhase     := 3,    (* Phase/angle of complex data. *)
    eCM_CplxUnwrappedPhase := 4 (* Unwrapped Phase/angle of complex data. *)
) UDINT;
END_TYPE

```

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM_Base

5.3.2 E_CM_CorrelationMode

```

TYPE E_CM_CorrelationMode :
    eCM_Base          := 0,    (* The correlation is calculate as the convolution of the two sig
nals. *)
    eCM_Normed        := 1,    (* The correlation is calculate as the of the two signals divided
by the number of elements added. *)
    eCM_Covariance    := 2,    (* The covariance with bias (divided by N instead N-1) is calcula
ted. *)
    eCM_CovarianceBessel := 3, (* The covariance is calculated. *)
    eCM_Pearson       := 4     (* The correlation is calculate as the Pearson correlation coeffi
cient. *)
) UDINT;
END_TYPE

```

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM_Base

5.3.3 E_CM_MCoefOrder

```

TYPE E_CM_MCoefOrder :
(
    eCM_N      := 0,    (* Count of included cases. *)
    eCM_Mean   := 1,    (* Mean value. *)
    eCM_StDev  := 2,    (* Standard deviation. *)
    eCM_Skew   := 3,    (* Skew value (third moment). *)
    eCM_Kurtosis := 4    (* Excess Kurtosis value. *)
) UDINT;
END_TYPE

```

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4013	PCまたはCX (x86、x64)	Tc3_CM_Base

5.3.4 E_CM_MeanStressCorrection

```
TYPE E_CM_MeanStressCorrection : (
    eCM_NoCorrection := 0,      (* No correction of the stress matrix. *)
    eCM_Goodman      := 1,      (* Goodman correction of the stress matrix. *)
    eCM_Gerber       := 2,      (* Gerber correction of the stress matrix. *)
) UDINT;
END_TYPE
```

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM_Base

5.3.5 E_CM_ScalingType

詳細はスペクトルスケーリング [▶ 371]の章を参照してください。

```
TYPE E_CM_ScalingType :
(
    eCM_NoScaling          := 0,
    eCM_DiracScaling       := 1,
    eCM_PeakAmplitude      := 2,
    eCM_RootPowerSum       := 3,
    eCM_RMS                := 4,
    eCM_GainCorrection     := 5,
    eCM_PowerSpectralDensity := 6,
    eCM_UnitaryScaling     := 7
) UDINT;
END_TYPE
```

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4013	PCまたはCX (x86、x64)	Tc3_CM_Base

5.3.6 E_CM_SlidingDFTType

```
TYPE E_CM_SlidingDFTType : (
    eCM_SDFT      := 0,      (* Plain (damped) SDFT. *)
    eCM_mSDFT     := 1,      (* Modulated SDFT. *)
    eCM_SlidingGoertzel := 2, (* Sliding Goertzel transform. *)
    eCM_DouglasAndSoh  := 3, (* Douglas and Soh algorithm. *)
) UDINT;
END_TYPE
```

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM_Base

5.3.7 E_CM_SpectrumType

```
TYPE E_CM_SpectrumType : (
    eCM_DFT      := 0,      (* Plain DFT. *)
    eCM_Magnitude := 1,      (* Magnitude spectrum. *)
    eCM_Power     := 2,      (* Power spectrum. *)
) UDINT;
END_TYPE
```

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM_Base

5.3.8 E_CM_UnwrapMethod

```

TYPE E_CM_UnwrapMethod :
(
    eCM_NoUnwrapping      := 0,
    eCM_ThresholdUnwrapping := 1
) UDINT;
END_TYPE

```

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM_Base

5.3.9 E_CM_WindowMode

```

TYPE E_CM_WindowMode : (
    eCM_SlidingWindow := 0,      (* The analysis is done over the last cycles in each cycle. *)
    eCM_FixWindow      := 1,      (* The analysis is done over an interval of fix size. If one interval
    is passed another interval begins. *)
    eCM_Continuous     := 2      (* The analysis is done for the whole time since starting of the anal
    ysis. *)
) UDINT;
END_TYPE

```

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM_Base

5.3.10 E_CM_WindowType

詳細はウィンドウ関数 [▶ 19]の章を参照してください。

```

TYPE E_CM_WindowType :
(
    eCM_HannWindow          := 16#05300901,
    eCM_RectangularWindow   := 16#05300902,
    eCM_BartlettWindow       := 16#05300905,
    eCM_HammingWindow        := 16#05300906,
    eCM_KaiserWindow         := 16#05300907,
    eCM_FlatTopWindow        := 16#05300917
) UDINT;
END_TYPE

```

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM_Base

5.3.11 E_MA_ElementTypeCode

```

TYPE E_MA_ElementTypeCode :
(
    eMA_TypeCode_NONE      := 0,      (* Internally used. *)
    eMA_TypeCode_BYTE      := 2,
    eMA_TypeCode_CHAR      := 3,
    eMA_TypeCode_WCHAR     := 4,
    eMA_TypeCode_BOOL      := 5,      (* Boolean type. *)
    eMA_TypeCode_INT16     := 6,
    eMA_TypeCode_UINT16    := 7,
    eMA_TypeCode_INT32     := 8,      (* Used e.g. for classification results. *)
    eMA_TypeCode_UINT32    := 9,
    eMA_TypeCode_INT64     := 10,
    eMA_TypeCode_UINT64    := 11,      (* 64-bit long unsigned. Use for statistical counters. *)
    eMA_TypeCode_REAL      := 12,      (* Unsupported: 32-bit floating point type. *)
    eMA_TypeCode_LREAL     := 13,      (* Standard floating-point type. *)
    eMA_TypeCode_LCOMPLEX  := 15,      (* Standard 128-bit complex type (real part first). *)

```

```
eMA_TypeCode_SPUNKNOWN := 22,      (* Used for TCOM Pointers. *)
) UDINT;
END_TYPE
```

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM_Base

5.3.12 T_CM_WindowParameters

```
TYPE T_CM_WindowParameters : ARRAY [0..4] OF LREAL;
END_TYPE
```

● T_CM_WindowParametersの使用方法

i ウィンドウ関数の設定用フリーパラメータは、長さ5の一般的な配列で指定されます。eCM_KaiserWindow が選択された場合は、この配列の最初の値のみが使用され、eCM_FlatTopWindow が選択された場合は、全ての値が使用されます。このパラメータは、他の全てのウィンドウタイプには影響しません。

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM_Base

5.3.13 エラーコード

5.3.13.1 E_CM_ErrorCode

コード(HRESULT)	シンボル (識別子)	説明/対策
0	eCM_OK	エラーはなく全て正常です。
ロジックエラー		
16#9851_0100	eCM_ErrLogic	一般的なロジックエラーです。
16#9851_0102	eCM_ErrLogic_NotImplemented	機能が実装されていません。
16#9851_0110	eCM_ErrLogic_LackOfInitialization	アルゴリズムが正しく初期化されていません。
16#9851_0125	eCM_ErrLogic_InvalidObjectState	このオブジェクト状態では操作は無効です。
設定エラー		
16#9851_1000	eCM_ErrConfig	一般的な設定エラーです。
ADSERR_DEVICE_NOMEMORYへのマッピング		
16#9851_1100	eCM_ErrConfig_OutOfMemory	メモリ割り当てに失敗しました。=> ルータのメモリを増やしてください（メモリ管理 [▶ 68]の章を参照）。
以下の全てのエラーは、HRESULT が ADS_E_INVALIDPARM になります。		
16#9851_1800	eCM_ErrConfig_IllegalParameter	設定パラメータが無効です。
16#9851_1900	eCM_ErrConfig_ParameterOutOfRange	設定パラメータが範囲外です。
16#9851_1901	eCM_ErrConfig_ParameterOutOfRange_NoPowerOfTwo	パラメータは2の累乗にしてください。
16#9851_1902	eCM_ErrConfig_ParameterOutOfRange_FFT_length_Zero	FFTの長さがゼロです。
16#9851_1903	eCM_ErrConfig_ParameterOutOfRange_DecibelThreshold_too_small	デシベルの閾値が低すぎるため、アンダーフローが発生する可能性があります。

コード(HRESULT)	シンボル (識別子)	説明／対策
16#9851_1904	eCM_ErrConfig_ParameterOutOfRange_LogThreshold_too_small	対数の閾値が低すぎるため、アンダーフローが発生する可能性があります。
16#9851_1905	eCM_ErrConfig_ParameterOutOfRange_nInLength_Minimum_two	入力長が小さすぎます。値は2以上である必要があります。
16#9851_190D	eCM_ErrConfig_ParameterOutOfRange_nChannels_smaller_one	チャンネル数がゼロです。
16#9851_190E	eCM_ErrConfig_ParameterOutOfRange_nBins_smaller_one	ビン数がゼロです。
16#9851_190F	eCM_ErrConfig_ParameterOutOfRange_invalid_limit_interval	下限値が上限値より小さくありません。
16#9851_1910	eCM_ErrConfig_ParameterOutOfRange_unknown_scaling_type	スケーリングタイプが不明です。
16#9851_1911	eCM_ErrConfig_ParameterOutOfRange_illegal_quantile_argument	分位値が [0～. 1] の範囲外です。
16#9851_1912	eCM_ErrConfig_ParameterOutOfRange_illegal_threshold_order	閾値の順序が無効です。閾値は昇順にしてください。
16#9851_1913	eCM_ErrConfig_ParameterOutOfRange_threshold_number_toolarge	設定した閾値よりも多くの閾値が指定されました。
16#9851_1914	eCM_ErrConfig_ParameterOutOfRange_Integration_limit_too_low	積分の上限が低すぎます。
16#9851_1915	eCM_ErrConfig_ParameterOutOfRange_Integration_limit_too_high	積分の上限が高すぎます。
16#9851_1916	eCM_ErrConfig_ParameterOutOfRange_Integration_limits_inconsistent	積分の制限値が不整合です。
16#9851_1917	eCM_ErrConfig_ParameterOutOfRange_Samplerate_not_positive	サンプリングレートがゼロまたはマイナスです。
16#9851_191A	eCM_ErrConfig_ParameterOutOfRange_TimeConstant_too_small	転送時の定数が小さすぎます。
16#9851_192C	eCM_ErrConfig_ParameterOutOfRange_fScaleFactor_invalid	スケーリング係数が無効です。
16#9851_192D	eCM_ErrConfig_ParameterOutOfRange_DivThreshold_too_small	分割の閾値が低すぎるため、アンダーフローが発生する可能性があります。
16#9851_192E	eCM_ErrConfig_ParameterOutOfRange_nMaxBands_zero	設定可能な帯域数がゼロです。
16#9851_192F	eCM_ErrConfig_ParameterOutOfRange_nOrder_invalid	操作が許可されていません。
16#9851_1930	eCM_ErrConfig_ParameterOutOfRange_fDecayTime_invalid	fDecayTimeは許可されていません。
16#9851_1B00	eCM_ErrConfig_ParameterMismatch	パラメータの依存関係が満たされていません。
16#9851_1B01	eCM_ErrConfig_ParameterMismatch_WindowLength_larger_FFT_length	ウィンドウ長がFFT長より大きくなっています。
16#9851_1B06	eCM_ErrConfig_ParameterMismatch_overlap_larger_BufferLength	オーバーラップがバッファ長より大きくなっています。
ランタイムエラー (データ処理中)		
これらのエラーは、HRESULT が ADS_E_INVALIDPARM になります。		
16#9851_2000	eCM_ErrRuntime	一般的なランタイムエラーです。
16#9851_2011	eCM_ErrRuntime_IllegalPointer	許可されていない (インターフェース) ポインタまたはメモリアドレスです。

コード(HRESULT)	シンボル (識別子)	説明／対策
16#9851_2015	eCM_ErrRTime_NonMonotonousInputData	入力データが要求された単調性を満たしていません。
16#9851_2016	eCM_ErrRTime_ErornousChannelResults	少なくとも1つのチャンネルの結果に誤りがあります。
16#9851_2021	eCM_ErrRTime_IllegalBuffer	許可されていないデータバッファです。
不正な入力バッファパラメータです (固定バッファを持つADS呼び出しで発生する可能性あり)。		
16#9851_2023	eCM_ErrRTime_IllegalInput	許可されていない入力バッファパラメータです。
16#9851_2025	eCM_ErrRTime_IllegalInputDimensionNumber	入力バッファに許可されていない次元数があります。
16#9851_2026	eCM_ErrRTime_IllegalInputShape	入力バッファの形式が許可されていません。
16#9851_2028	eCM_ErrRTime_IllegalInputDataType	入力バッファの要素型が許可されていません。
16#9851_202A	eCM_ErrRTime_IllegalInputNoArray	入力パラメータにマルチアレイが含まれていません。
不正な出力バッファパラメータ (固定バッファを持つADS呼び出しで発生する可能性あり)。		
16#9851_2030	eCM_ErrRTime_IllegalOutput	許可されていない出力バッファパラメータです。
16#9851_2032	eCM_ErrRTime_IllegalOutputDimensionNumber	出力バッファに許可されていない次元数があります。
16#9851_2033	eCM_ErrRTime_IllegalOutputShape	出力バッファの形式が許可されていません。
16#9851_2034	eCM_ErrRTime_IllegalOutputDataType	出力バッファの要素型が許可されていません。
16#9851_2035	eCM_ErrRTime_IllegalOutputNoArray	出力パラメータにマルチアレイが含まれていません。
補間エラー		
16#9851_2060	eCM_ErrRTime_Interpolation	一般的な補間エラーです。
16#9851_2063	eCM_ErrRTime_Interpolation_OutOfBounds	入力データが制限値を超えています。
16#9851_2064	eCM_ErrRTime_Interpolation_InvalidDimension	入力データを含む配列が、使用方法に対して許容できないサイズです。
16#9851_2065	eCM_ErrRTime_Interpolation_InvalidConstraints	終点での導関数の制限を満たすことができません。
情報コード		
16#9851_B103	eCM_InfRTime_InsufficientInputData	代表的な結果を得るための入力データが不十分です。
16#9851_B104	eCM_InfRTime_InvalidInputData	入力データは許可されていません。
16#9851_B105	eCM_InfRTime_AmbiguousChannelResults	少なくとも1つのチャンネルの結果が不明確です。
16#9851_B106	eCM_InfRTime_nBufferLength_too_small	設定したバッファ長が入力データに対して小さすぎます。

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM_Base

5.3.13.2 E_CMA_ErrorCode

これらのエラーコードはリアルタイムの状況でのみ必要です。解析ファンクションブロックは、PLC宣言部で正しく割り当てる必要があることに注意してください。

最初に設定エラーを解決し、次に初期化エラーを解決します。

例えば、eCMA_ErrConfig_InvalidOwnID というエラーを投げるインスタンスがあれば、最初にそれを解決する必要があります。他のファンクションブロックでのランタイムエラーは、後続のエラーの可能性あります。

コード (HRESULT)	シンボル (識別子)	説明 / 対策
0	eCMA_OK	エラーはなく全て正常です。
設定エラー		
16#9852_0101	eCMA_ErrConfig_InvalidOwnID	無効な転送IDが割り当てられた
16#9852_0102	eCMA_ErrConfig_InvalidDestID	無効な転送先IDが割り当てられた
16#9852_0103	eCMA_ErrConfig_InvalidBufferNumber	結果のために初期化されるべきマルチアレイの数が無効である。
16#9852_0104	eCMA_ErrConfig_InvalidTimeout	無効なタイムアウト: 0us << tTransferTimeout << タスクサイクル時間
初期化エラー		
16#9852_0201	eCMA_ErrInit_IllegalInitContext	初期化ができない。初期化コンテキストが不正か、内部メンバが初期化されていない。
16#9852_0202	eCMA_ErrInit_InitTransferTrayFailed	搬送トレイの初期化に失敗しました。TcCOMオブジェクトの状態とルータのメモリをチェックする（メモリ [▶ 68] 管理を参照）。インストールされているTwinCATのバージョンを確認します（システム要件を [▶ 61] 参照）。
16#9852_0203	eCMA_ErrInit_NoStreamAllocated	解析チェーンが間違っている。全てのOwnIDとDestIDをチェックする。
16#9852_0204	eCMA_ErrInit_StreamOverrun	十分なストリームがない。ST_CM_TransferTray_InitParsを調整します。
実行時エラー		
16#9852_0301	eCMA_ErrRTime_InvalidPointer	NULLポインタが確保された
16#9852_0302	eCMA_ErrRTime_InvalidDataBufferSize	データバッファのサイズが不正に割り当てられた
16#9852_0303	eCMA_ErrRTime_InvalidElementType	無効な要素タイプが割り当てられた
16#9852_0304	eCMA_ErrRTime_InvalidElementCnt	要素数が一致しない。（要素数、マルチアレイバッファサイズ、開始インデックスをチェック）
16#9852_0305	eCMA_ErrRTime_InvalidStartIndex	無効なpStartIndexが割り当てられた（バッファサイズをチェックする）
16#9852_0311	eCMA_ErrRTime_MissingConfiguration	引数が設定されていない。まずConfigure() メソッドを呼び出す。
16#9852_0321	eCMA_ErrRTime_NoMultiArrayAvailable	結果にマルチアレイは使用できない。解析チェーン、タスクのサイクルタイム、マルチアレイの数（通常、各リングに少なくとも3つ）をチェックする。

要件

開発環境	ターゲットシステム	必要なPLCライブラリ
TwinCAT v3.1.4013	PCまたはCX (x86、x64)	Tc3_CM

5.3.13.3 E_MA_ErrorCode

コード (HRESULT)	symbol	説明 / 対策
0	eMA_OK	エラーはなく全て正常です。
ロジックエラー		
16#9871_0100	eMA_ErrLogic	一般的なロジックエラーです。
16#9871_0102	eMA_ErrLogic_NotImplemented	機能が実装されていません。
16#9871_0110	eMA_ErrLogic_LackOfInitialization	アルゴリズムが正しく初期化されていません。
16#9871_0126	eMA_ErrLogic_ObjectCreationFailed	オブジェクトの作成に失敗しました。ドライバが無いが、古い可能性があります。
設定エラー		
16#9871_1000	eMA_ErrConfig	一般的な設定エラーです。
ADSERR_DEVICE_NOMEMORYへのマッピング		
16#9871_1100	eMA_ErrConfig_OutOfMemory	メモリ割り当てに失敗しました。=> ルータのメモリを増やしてください（詳細はメモリ管理 [▶ 68]の章を参照）。ADSERR_DEVICE_NOMEMORY と同等です。
以下の全てのエラーは、HRESULT が ADS_E_INVALIDPARAM になります。		
16#9871_1800	eMA_ErrConfig_IllegalParameter	設定パラメータが無効です。
16#9871_1900	eMA_ErrConfig_ParameterOutOfRange	設定パラメータが範囲外です。
ランタイムエラー（データ処理中）		
16#9871_2000	eMA_ErrRTime	一般的なランタイムエラーです。
16#9871_2011	eMA_ErrRTime_IllegalPointer	不正な（インターフェース）ポインタまたはメモリアドレスです。
16#9871_2012	eMA_ErrRTime_EmptyArray	マルチアレイにデータがありません（次元サイズの積がゼロです）。
16#9871_2013	eMA_ErrRTime_InstanceExists	すでに割り当てられたインスタンスがあります。
16#9871_2014	eMA_ErrRTime_NoInstanceExists	割り当てられたインスタンスも初期化されたインスタンスもありません。
16#9871_2021	eMA_ErrRTime_IllegalBuffer	不正なデータバッファです。
16#9871_2022	eMA_ErrRTime_IllegalSubarraySize	サブ配列のサイズが不正です。
16#9871_2029	eMA_ErrRTime_IllegalPermutation	要求された次元の並べ替えは無効です。
不正な入力バッファパラメータ		
16#9871_2023	eMA_ErrRTime_IllegalInput	無効な入力署名です。
16#9871_2025	eMA_ErrRTime_IllegalInputDimensionNumber	入力データに無効な次元数があります。
16#9871_2026	eMA_ErrRTime_IllegalInputShape	入力データが不正な形式です。
16#9871_2028	eMA_ErrRTime_IllegalInputDataType	入力データストリームの要素型が不正です。
16#9871_202A	eMA_ErrRTime_IllegalInputNoArray	入力パラメータとして渡されたマルチアレイがありません
不正な出力バッファパラメータ		
16#9871_2030	eMA_ErrRTime_IllegalOutput	無効な出力署名です。
16#9871_2035	eMA_ErrRTime_IllegalOutputNoArray	出力パラメータとして渡されたマルチアレイがありません。

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM_Base

5.3.14 InitPars構造体

5.3.14.1 ST_CM_AnalyticSignal_InitPars

ファンクションブロック固有の構造体で、ファンクションブロックの初期化時に解析される初期化パラメータを保持します。

```

TYPE ST_CM_AnalyticSignal_InitPars EXTENDS ST_CM_Object_InitPars :
STRUCT
    nFFT_Length      : UDINT := 512;      (* Length of FFT. *)
    nWindowLength    : UDINT := 400;      (* Length of analysis window. *)
    nChannels         : UDINT := 1;       (* Number of channels. *)
END_STRUCT
END_TYPE

```

- nFFT_Length はFFTの長さです。1より大きく、2の累乗である必要があります。
- nWindowLength は解析ウィンドウの長さ（サンプル数）です。長さは1より大きく、偶数である必要があります。
- nChannels は、独立したチャンネルの数を定義します。これはゼロより大きくなければなりません。

● アーティファクト（解析上の誤差）

1 nFFT_Length は、nWindowLength の値以上でなければなりません。計算上のアーティファクトを回避するため、nFFT_Length は、nWindowLength より少なくとも25大きくする必要があります。このファンクションブロックでは、円形エイリアシングを回避するために、ウィンドウ長に対してFFT長を長くするのが合理的です。

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM_Base

5.3.14.2 ST_CM_ArgSort_InitPars

ファンクションブロック固有の構造体で、ファンクションブロックの初期化時に解析される初期化パラメータを保持します。

```

TYPE ST_CM_ArgSort_InitPars EXTENDS ST_CM_Object_InitPars :
STRUCT
    nInLength        : UDINT := 256;      (* Length of input data array. *)
    bSortDownward    : BOOL := FALSE;     (* If true, sort in descending order (largest values first). *)
    fScaleFactor      : LREAL := 1.0;      (* Scaling factor to transform index values, for example to frequency values. *)
    nChannels         : UDINT := 1;       (* Number of channels. *)
END_STRUCT
END_TYPE

```

- nInLength は、入力配列の長さです。
- bSortDownward は、データを昇順で並べるか降順で並べるかを選択するフラグです。bSortDownward が TRUE の場合、最大の値が先頭に配置されます。
- fScaleFactor を使用すると、例えば、インデックス位置の代わりに、振幅に関連する周波数を直接表示できます（fScaleFactor = 1）。
- nChannels は、独立したチャンネルの数を定義します。これはゼロより大きくなければなりません。

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM_Base

5.3.14.3 ST_CM_CrestFactor_InitPars

ファンクションブロック固有の構造体で、ファンクションブロックの初期化時に解析される初期化パラメータを保持します。

```

TYPE ST_CM_CrestFactor_InitPars EXTENDS ST_CM_Object_InitPars :
STRUCT
  nChannels          : UDINT := 32;          (* Number of channels. *)
  nSubChannels       : UDINT := 0;          (* Number of subchannels. *)
  nBufferLength      : UDINT := 250;        (* Buffer length. *)
  fDecibelThreshold  : LREAL := cCM_MinArgLog10; (* Minimum argument of decadic logarithm for 6
4-bit IEEE 754 arithmetic. *)
END_STRUCT
END_TYPE

```

- nChannels は、独立したチャンネルの数を定義します。これはゼロより大きくなければなりません。
- nSubChannels は、独立したサブチャンネルの数を定義します。これはゼロ以上でなければなりません。
- nBufferLength は、内部バッファに保持されているチャンネルごとの入力値の数です。
- fDecibelThreshold は非常に小さな浮動小数点値で、ゼロより大きな値です。この数値より小さい値は、デシベルスケールへの変換前にこの値に置き換えられます。(その目的は、値域エラーの回避です。ゼロの対数は定義されておらず、小さな引数の極限では値が負の無限大に発散します。数字のゼロの引数、arg(0)も同様です。可能な最小値は2.3e-308で、これは定数cCM_MinArgLog10 と同じ値です。)

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM_Base

5.3.14.4 ST_CM_CrestFactorPlus_InitPars

ファンクションブロック固有の構造体で、ファンクションブロックの初期化時に解析される初期化パラメータを保持します。

```

TYPE ST_CM_CrestFactorPlus_InitPars EXTENDS ST_CM_Object_InitPars :
STRUCT
  nChannels          : UDINT := 32;          (* Number of channels. *)
  nSubChannels       : UDINT := 0;          (* Number of subchannels. *)
  nBufferLength      : UDINT := 250;        (* Buffer length. *)
  bTransformToDecibel : BOOL := TRUE;        (* Transform to Decibel. *)
  fDecibelThreshold  : LREAL := cCM_MinArgLog10; (* Minimum argument of decadic logarithm for 64-
bit IEEE 754 arithmetic. *)
END_STRUCT
END_TYPE

```

- nChannels は、独立したチャンネルの数を定義します。これはゼロより大きくなければなりません。
- nSubChannels は、独立したサブチャンネルの数を定義します。これはゼロ以上でなければなりません。
- nBufferLength は、内部バッファに保持されているチャンネルごとの入力値の数です。
- bTransformToDecibel は、FFTの結果をデシベル単位に変換するかどうかを示すブール値で、変換 $x \rightarrow 20 * \log_{10}(x)$ に従って処理されます。
- fDecibelThreshold は非常に小さな浮動小数点値で、ゼロより大きな値です。この数値より小さい値は、デシベルスケールへの変換前にこの値に置き換えられます。(その目的は、値域エラーの回避です。ゼロの対数は定義されておらず、小さな引数の極限では値が負の無限大に発散します。数字のゼロの引数、arg(0)も同様です。可能な最小値は2.3e-308で、これは定数cCM_MinArgLog10 と同じ値です。)

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM_Base

5.3.14.5 ST_CM_ComplexDataHandling_InitPars

ファンクションブロック固有の構造体で、ファンクションブロックの初期化時に解析される初期化パラメータを保持します。

```

TYPE ST_CM_ComplexDataHandling_InitPars EXTENDS ST_CM_Object_InitPars :
STRUCT
    eComplexDataHandling : E_CM_ComplexDataHandling := E_CM_ComplexDataHandling.eCM_CplxReal;    (*
Extraction of real or imaginary part or calculation of absolute value or phase of complex data. *)
    nInLength             : UDINT := 256;                                                       (*
Length of input data array. *)
    fPhaseThreshold       : LREAL := 3.14159265358979;                                          (*
Threshold value for phase unwrapping. *)
    nChannels             : UDINT := 1;                                                         (*
Number of channels. *)
END_STRUCT
END_TYPE

```

- eComplexDataHandling は、Call() を使用して実行するアクションを定義します。例えば、実数部または虚数部の抽出、複素数入力データの量または位相の計算などです。
- nInLength は、入力配列の長さです。
- fPhaseThreshold フェーズを展開するための制限値を定義します。Piという値がルールとして使用されます。
- nChannels は、独立したチャンネルの数を定義します。これはゼロより大きくなければなりません。

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM_Base

5.3.14.6 ST_CM_ComplexFFT_InitPars

ファンクションブロック固有の構造体で、ファンクションブロックの初期化時に解析される初期化パラメータを保持します。

```

TYPE ST_CM_ComplexFFT_InitPars EXTENDS ST_CM_Object_InitPars :
STRUCT
    nFFT_Length : UDINT := 256;    (* Length of FFT. *)
    bForward     : BOOL  := TRUE;   (* Flag indicating forward FFT. *)
    nChannels    : UDINT := 1;      (* Number of channels. *)
END_STRUCT
END_TYPE

```

- nFFT_Length はFFTの長さです。1より大きく、2の累乗である必要があります。
- bForward はFFTの方向を指定するブール型のパラメータです。値がTRUE の場合、通常のFFTが計算されます。そうでない場合は、逆FFTが使用されます。
- nChannels は、独立したチャンネルの数を定義します。これはゼロより大きくなければなりません。

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM_Base

5.3.14.7 ST_CM_Correlation_InitPars

ファンクションブロック固有の構造体で、ファンクションブロックの初期化時に解析される初期化パラメータを保持します。

```

TYPE ST_CM_Correlation_InitPars EXTENDS ST_CM_Object_InitPars :
STRUCT
    nNegativeShift    : UDINT := 0;          (* Maximum negative
time shift in samples. *)
    nPositiveShift    : UDINT := 0;          (* Maximum positive
time shift in samples. *)
    nStepsize         : UDINT := 0;          (* Stepsize in Samp
les. *)
    eCorrelationMode  : E_CM_CorrelationMode := E_CM_CorrelationMode.eCM_Normed; (* Calculation mode
of correlation. *)
    eWindowMode       : E_CM_WindowMode := E_CM_WindowMode.eCM_FixWindow; (* Window handling,
i.e. sliding, fixed, continuous. *)
    nWindowLength     : UDINT := 1000;      (* Length of analys
is window. *)
    nChannels         : UDINT := 1;          (* Number of channe
ls. *)
END_STRUCT
END_TYPE

```

- nNegativeShift は、負方向への最大時間シフトをサンプル単位で指定します。
- nPositiveShift は、正方向への最大時間シフトをサンプル単位で指定します。
- nStepsize は、相関係数を計算する際のサンプル単位のステップサイズを定義します。
- eCorrelationMode は、相関パラメータの計算方式を定義します。
- eWindowMode は計算の基準となるウィンドウを定義します。ウィンドウの長さは、パラメータ nWindowLength で決定します。
- nWindowLength は解析ウィンドウの長さ（サンプル数）です。長さは1より大きく、偶数である必要があります。
- nChannels は、独立したチャンネルの数を定義します。これはゼロより大きくなければなりません。

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM_Base

5.3.14.8 ST_CM_DiscreteClassification_InitPars

ファンクションブロック固有の構造体で、ファンクションブロックの初期化時に解析される初期化パラメータを保持します。

```

TYPE ST_CM_DiscreteClassification_InitPars EXTENDS ST_CM_Object_InitPars :
STRUCT
    nChannels         : UDINT := 10;        (* Number of channels. *)
    nSubChannels       : UDINT := 0;        (* Number of subchannels. *)
    nMaxClasses        : UDINT := 3;        (* Number of configurable threshold classes. *)
END_STRUCT
END_TYPE

```

- nChannels は、独立したチャンネルの数を定義します。これはゼロより大きくなければなりません。
- nSubChannels は、独立したサブチャンネルの数を定義します。これはゼロ以上でなければなりません。
- nMaxClasses は、設定されているクラスの最大数です。これは1以上である必要があります。

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM_Base

5.3.14.9 ST_CM_EmpiricalMoments_InitPars

ファンクションブロック固有の構造体で、ファンクションブロックの初期化時に解析される初期化パラメータを保持します。

```

TYPE ST_CM_EmpiricalMoments_InitPars EXTENDS ST_CM_Object_InitPars :
STRUCT
    nChannels         : UDINT := 512;      (* Number of channels. *)

```

```

    nSubChannels      : UDINT := 0;      (* Number of subchannels. *)
    bPopulationEstimates : BOOL := TRUE;  (* Apply Bessel's correction to results. *)
END_STRUCT
END_TYPE

```

- nChannels は、独立したチャンネルの数を定義します。これはゼロより大きくなければなりません。
- nSubChannels は、独立したサブチャンネルの数を定義します。これはゼロ以上でなければなりません。
- bPopulationEstimates はブール型のパラメータで、ベッセル補正を行うかどうかを指定します。

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM_Base

5.3.14.10 ST_CM_Envelope_InitPars

ファンクションブロック固有の構造体で、ファンクションブロックの初期化時に解析される初期化パラメータを保持します。

```

TYPE ST_CM_Envelope_InitPars EXTENDS ST_CM_Object_InitPars :
STRUCT
    nFFT_Length      : UDINT := 512;      (* Length of FFT. *)
    nWindowLength    : UDINT := 400;      (* Length of analysis window. *)
    nChannels        : UDINT := 1;        (* Number of channels. *)
END_STRUCT
END_TYPE

```

- nFFT_Length はFFTの長さです。1より大きく、2の累乗である必要があります。
- nWindowLength は解析ウィンドウの長さ（サンプル数）です。長さは1より大きく、偶数である必要があります。
- nChannels は、独立したチャンネルの数を定義します。これはゼロより大きくなければなりません。

● アーティファクト（解析上の誤差）

I nFFT_Length は、nWindowLength の値以上でなければなりません。計算上のアーティファクトを回避するため、nFFT_Length は、nWindowLength より少なくとも25大きくする必要があります。このファンクションブロックでは、円形エイリアシングを回避するために、ウィンドウ長に対してFFT長を長くするのが合理的です。

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM_Base

5.3.14.11 ST_CM_EnvelopeSpectrum_InitPars

ファンクションブロック固有の構造体で、ファンクションブロックの初期化時に解析される初期化パラメータを保持します。

```

TYPE ST_CM_EnvelopeSpectrum_InitPars EXTENDS ST_CM_InitPars :
STRUCT
    nFFT_Length      : UDINT := 512;      (* Length of FFT. *)
    nWindowLength    : UDINT := 400;      (* Length of analysis window. *)
    bTransformToDecibel : BOOL := TRUE;    (* Transform to decibel. *)
    fDecibelThreshold : LREAL := cCM_MinArgLog10; (* Minimum
argument of decadic logarithm for 64-bit IEEE 754 arithmetic. *)
    eWindowType      : E_CM_WindowType := eCM_HannWindow; (* Window type. *)
    aWindowParameters : T_CM_WindowParameters := [2.5,1,1,1,1]; (* Window parameters for specifi
c windows, e.g. FlatTop. *)
    nOverlap         : UDINT := -1;        (* Size of overlap in samples. *)
    eScalingType      : E_CM_ScalingType := eCM_DiracScaling; (* Scaling type. *)
    nChannels        : UDINT := 1;        (* Number of channels. *)
END_STRUCT
END_TYPE

```

- `nFFT_Length` はスペクトルのFFTの長さです。1より大きく、2の累乗である必要があります。
- `nWindowLength` は解析ウィンドウの長さ（サンプル数）です。長さは1より大きく、偶数である必要があります。
- `bTransformToDecibel` は、FFTの結果をデシベル単位に変換するかどうかを示すブール値で、変換 $x \rightarrow 20 * \log_{10}(x)$ に従って処理されます。
- `fDecibelThreshold` は非常に小さな浮動小数点値で、ゼロより大きな値です。この数値より小さい値は、デシベルスケールへの変換前にこの値に置き換えられます。（その目的は、値域エラーの回避です。ゼロの対数は定義されておらず、小さな引数の極限では値が負の無限大に発散します。数字のゼロの引数、`arg(0)`も同様です。可能な最小値は $2.3e-308$ で、これは定数`cCM_MinArgLog10`と同じ値です。）
- `eWindowType` 使用するウィンドウ関数（`E_CM_WindowType` [▶ 274]型）を定義します。良いデフォルト値はウィンドウタイプ`eCM_HannWindow`です。
- `eWindowType` 使用するウィンドウ関数（`E_CM_WindowType` [▶ 274]型）を定義します。良いデフォルト値はウィンドウタイプ`eCM_HannWindow`です。
- `aWindowParameters` には、選択したウィンドウ関数のフリーパラメータが含まれます。
`eCM_KaiserWindow`を使用する場合、最初のエントリはパラメータ β を定義します。`eCM_FlatTopWindow`を使用する場合、全てのパラメータが使用されます。[ウィンドウ関数 \[▶ 19\]](#)を参照してください。
- `nOverlap` はオーバーラップするサンプル数を定義します。これはゼロ以上でなければなりません。値`cCM_OverlapRecommended`が選択された場合、推奨されるオーバーラップが内部で計算されます（`F_CM_CalculateRecommendedOverlap` [▶ 270]を参照）。値`cCM_OverlapInactive`は、内部バッファを無効化し、値をゼロに設定します。
- `eScalingType`は、絶対スケーリングが必要な場合に、使用するスケーリング方式（`E_CM_ScalingType` [▶ 273]型）を選択できるようにします。デフォルト値は`eCM_DiracScaling`です。スケーリングを選択する際には、信号が決定論的な信号なのか、確率的な部分を持つ広帯域信号なのか、信号の種類を考慮する必要があります。それぞれ異なるスケーリングを必要とします。
- `nChannels` は、独立したチャンネルの数を定義します。これはゼロより大きくなければなりません。

● アーティファクト（解析上の誤差）

I `nFFT_Length` は、`nWindowLength` の値以上でなければなりません。計算上のアーティファクトを回避するため、`nFFT_Length` は、`nWindowLength` より少なくとも25大きくする必要があります。このファンクションブロックでは、円形エイリアシングを回避するために、ウィンドウ長に対してFFT長を長くするのが合理的です。

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM_Base

5.3.14.12 ST_CM_HistArray_InitPars

ファンクションブロック固有の構造体で、ファンクションブロックの初期化時に解析される初期化パラメータを保持します。

```

TYPE ST_CM_HistArray_InitPars EXTENDS ST_CM_Object_InitPars :
STRUCT
    nChannels      : UDINT := 512; (* Number of channels. *)
    nSubChannels   : UDINT := 0;   (* Number of subchannels. *)
    nBins          : UDINT := 100; (* Number of bins. *)
    fMinBinned     : LREAL := -120; (* Minimum binned value. *)
    fMaxBinned     : LREAL := 100;  (* Maximum binned value. *)
END_STRUCT
END_TYPE

```

- `nChannels` は、独立したチャンネルの数を定義します。これはゼロより大きくなければなりません。
- `nSubChannels` は、独立したサブチャンネルの数を定義します。これはゼロ以上でなければなりません。

- nBins はヒストグラムのビンの数です。これは1以上である必要があります。多くの場合、10と20の間の値を選択するのが合理的です。この値には、fMinBinned 以下またはfMaxBinned 以上の値を示す2つの特別なピンは含まれません。
- fMinBinned は、通常のヒストグラムビンでサンプルをカウントする際の下限值です。
- fMaxBinned は、通常のヒストグラムビンでサンプルをカウントする際の上限值です。fMaxBinnedは、fMinBinnedよりも大きくする必要があります。

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM_Base

5.3.14.13 ST_CM_InstantaneousFrequency_InitPars

ファンクションブロック固有の構造体で、ファンクションブロックの初期化時に解析される初期化パラメータを保持します。

```

TYPE ST_CM_InstantaneousFrequency_InitPars EXTENDS ST_CM_Object_InitPars :
STRUCT
    nFFT_Length      : UDINT := 512;          (* Length of FFT. *)
    nWindowLength    : UDINT := 400;          (* Length of analysis window. *)
    fMagnitudeThreshold : LREAL := cCM_MinArgDiv; (* Minimum value for the numerical calculability. *)
    fSampleRate      : LREAL := 50000;        (* Sample rate in Hertz. *)
    nChannels        : UDINT := 1;            (* Number of channels. *)
END_STRUCT
END_TYPE

```

- nFFT_Length はFFTの長さです。1より大きく、2の累乗である必要があります。
- nWindowLength は解析ウィンドウの長さ（サンプル数）です。長さは1より大きく、偶数である必要があります。
- fMagnitudeThreshold は、瞬時周波数の数値計算における制限値を定義します。制限値は特定の値に関連します。 $\text{conj}\{\mathcal{H}\{x[n]\}\}\mathcal{H}\{x[n+1]\} < \text{cCM\_MinArgDiv}$
- fSampleRate 入力される時間信号のサンプリング・レート。この値は、Hz単位での結果のスケーリングに使用されます。
- nChannels は、独立したチャンネルの数を定義します。これはゼロより大きくなければなりません。

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM_Base

5.3.14.14 ST_CM_InstantaneousPhase_InitPars

ファンクションブロック固有の構造体で、ファンクションブロックの初期化時に解析される初期化パラメータを保持します。

```

TYPE ST_CM_InstantaneousPhase_InitPars EXTENDS ST_CM_Object_InitPars :
STRUCT
    nFFT_Length      : UDINT := 512;          (* Length of FFT. *)
    nWindowLength    : UDINT := 400;          (* Length of analysis window. *)
    eUnwrapMethod    : E_CM_UnwrapMethod := eCM_ThresholdUnwrapping; (* Unwrap method for phase values. *)
    fPhaseThreshold  : LREAL := cCM_MinArgDiv; (* Minimum value for calculating the instantaneous phase. *)
    nChannels        : UDINT := 1;            (* Number of channels. *)
END_STRUCT
END_TYPE

```

- nFFT_Length はFFTの長さです。1より大きく、2の累乗である必要があります。

- `nWindowLength` は解析ウィンドウの長さ（サンプル数）です。長さは1より大きく、偶数である必要があります。
- `eUnwrapMethod`は、位相のアンラッピングに使用される方法を、2 PIの倍数の位相に関して定義します（[E_CM_UnwrapMethod](#) [▶ 274]を参照）。
- `fPhaseThreshold` 瞬時位相を計算するための制限値。この値は信号の包絡線に関係します。解釈：信号レベルが低すぎる場合、位相の計算は数値的に不確実であり、信頼できる評価ができません。その場合、0が位相として出力されます。

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM_Base

5.3.14.15 ST_CM_IntegratedRMS_InitPars

ファンクションブロック固有の構造体で、ファンクションブロックの初期化時に解析される初期化パラメータを保持します。

```

TYPE ST_CM_IntegratedRMS_InitPars EXTENDS ST_CM_Object_InitPars :
STRUCT
    nFFT_Length          : UDINT := 512;                (* Length of FFT. *)
    nWindowLength        : UDINT := 400;                (* Length of analysis window. *)
    *)
    fSampleRate           : LREAL := 20000;             (* Sample rate in Hertz. *)
    fLowerFrequencyLimit  : LREAL := 20.0;              (* Lower limit of frequency b
and in Hertz. *)
    fUpperFrequencyLimit  : LREAL := 1000.0;            (* Upper limit of frequency b
and in Hertz. *)
    nMaxBands             : UDINT := 1;                (* Maximum number of frequenc
y bands. *)
    nOrder                : UDINT := 2;                (* Maximum order of integrati
on: 0 = acceleration, 1 = velocity, 2 = place. *)
    nChannels             : UDINT := 2;                (* Number of channels. *)
    eWindowType           : E_CM_WindowType := eCM_HannWindow; (* Window type. *)
    aWindowParameters     : T_CM_WindowParameters := [2.5,1,1,1,1]; (* Window parameters for spec
ific windows, e.g. FlatTop. *)
    nOverlap              : UDINT := -1;                (* Size of overlap in samples
. *)
    bTransformToDecibel   : BOOL := TRUE;               (* Transform to decibel. *)
    fDecibelThreshold     : LREAL := cCM_MinArgLog10;    (* Minimum argument of decadi
c logarithm for 64- bit IEEE 754 arithmetic. *)
END_STRUCT
END_TYPE

```

- `nFFT_Length` はFFTの長さです。1より大きく、2の累乗である必要があります。
- `nWindowLength` は解析ウィンドウの長さ（サンプル数）です。長さは1より大きく、偶数である必要があります。
- `fSampleRate` 入力される時間信号のサンプリング・レート。この値は、Hz単位での結果のスケールングに使用されます。
- `fLowerFrequencyLimit` 考慮される周波数間隔の下限。下限周波数は、サンプリングレートをFFT長で割った値以上でなければなりません。
- `fUpperFrequencyLimit` 考慮される周波数間隔の上限。上限周波数は、サンプリングレートの半分以下で、下限周波数より大きくなければなりません。
- `nMaxBands` は、RMS値を計算する周波数帯域の最大数を指定します。
- `nOrder` は積分の最大次数です。これは0から2までの整数でなければなりません。チャンネルごとに決定される値の数は(`nOrder+1`)です。
- `nChannels` は、独立したチャンネルの数を定義します。これはゼロより大きくなければなりません。
- `eWindowType` 使用するウィンドウ関数（[E_CM_WindowType](#) [▶ 274]型）を定義します。良いデフォルト値はウィンドウタイプ`eCM_HannWindow` です。
- `aWindowParameters` には、選択したウィンドウ関数のフリーパラメータが含まれます。
`eCM_KaiserWindow`を使用する場合、最初のエントリはパラメータ β を定義します。`eCM_FlatTopWindow`を使用する場合、全てのパラメータが使用されます。[ウィンドウ関数](#) [▶ 19]を参照してください。

- `nOverlap` はオーバーラップするサンプル数を定義します。これはゼロ以上でなければなりません。値 `cCM_OverlapRecommended` が選択された場合、推奨されるオーバーラップが内部で計算されます (`F_CM_CalculateRecommendedOverlap` [▶ 270] を参照)。値 `cCM_OverlapInactive` は、内部バッファを無効化し、値をゼロに設定します。
- `bTransformToDecibel` は、FFTの結果をデシベル単位に変換するかどうかを示すブール値で、変換 $x \rightarrow 20 * \log_{10}(x)$ に従って処理されます。
- `fDecibelThreshold` は非常に小さな浮動小数点値で、ゼロより大きな値です。この数値より小さい値は、デシベルスケールへの変換前にこの値に置き換えられます。(その目的は、値域エラーの回避です。ゼロの対数は定義されておらず、小さな引数の極限では値が負の無限大に発散します。数字のゼロの引数、`arg(0)` も同様です。可能な最小値は $2.3e-308$ で、これは定数 `cCM_MinArgLog10` と同じ値です。)

● ウィンドウの長さに注意

i `nWindowLength` の値は `nFFT_Length` の値以下になるようにしてください。FFTの長さは、必要な周波数分解能に合わせることができます。一般的には、FFT長の約4/5の値がウィンドウ長としてよく使用されます。

`nFFT_Length` が `nWindowLength` よりも大きい場合、FFTの周波数分解能が（したがって、戻り値ベクトルの長さも）増加します。長さの差は、フーリエ変換の前にゼロで満たされます。これは、より高い周波数分解能を得るためや、時間領域での逆変換による計算で円形エイリアシングを避けるために有用です。ただし、より高い周波数分解能にもかかわらず、結果にはそれ以上の情報は含まれていません。

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM_Base

5.3.14.16 ST_CM_MagnitudeSpectrum_InitPars

ファンクションブロック固有の構造体で、ファンクションブロックの初期化時に解析される初期化パラメータを保持します。

```

TYPE ST_CM_MagnitudeSpectrum_InitPars EXTENDS ST_CM_Object_InitPars :
STRUCT
    nFFT_Length      : UDINT := 512;                (* Length of FFT. *)
    nWindowLength    : UDINT := 400;                (* Length of analysis window. *)
    fDecibelThreshold : LREAL := cCM_MinArgLog10;    (* Minimum argument of decadic
    logarithm for 64-bit IEEE 754 arithmetic. *)
    bTransformToDecibel : BOOL := TRUE;             (* Transform to decibel. *)
    eWindowType       : E_CM_WindowType := eCM_HannWindow; (* Window type. *)
    aWindowParameters : T_CM_WindowParameters := [2.5,1,1,1,1]; (* Window parameters for spec-
    ific windows, e.g. FlatTop. *)
    nOverlap          : UDINT := -1;                (* Size of overlap in samples. *)
    eScalingType       : E_CM_ScalingType := eCM_DiracScaling; (* Scaling type. *)
    nChannels          : UDINT := 1;                (* Number of channels. *)
END_STRUCT
END_TYPE

```

- `nFFT_Length` はFFTの長さです。1より大きく、2の累乗である必要があります。
- `nWindowLength` は解析ウィンドウの長さ（サンプル数）です。長さは1より大きく、偶数である必要があります。
- `fDecibelThreshold` は非常に小さな浮動小数点値で、ゼロより大きな値です。この数値より小さい値は、デシベルスケールへの変換前にこの値に置き換えられます。(その目的は、値域エラーの回避です。ゼロの対数は定義されておらず、小さな引数の極限では値が負の無限大に発散します。数字のゼロの引数、`arg(0)` も同様です。可能な最小値は $2.3e-308$ で、これは定数 `cCM_MinArgLog10` と同じ値です。)
- `bTransformToDecibel` は、FFTの結果をデシベル単位に変換するかどうかを示すブール値で、変換 $x \rightarrow 20 * \log_{10}(x)$ に従って処理されます。
- `eWindowType` 使用するウィンドウ関数 (`E_CM_WindowType` [▶ 274] 型) を定義します。良いデフォルト値はウィンドウタイプ `eCM_HannWindow` です。

- aWindowParameters には、選択したウィンドウ関数のフリーパラメータが含まれます。eCM_KaiserWindowを使用する場合、最初のエントリはパラメータβを定義します。eCM_FlatTopWindowを使用する場合、全てのパラメータが使用されます。ウィンドウ関数 [▶ 19]を参照してください。
- nOverlap はオーバーラップするサンプル数を定義します。これはゼロ以上でなければなりません。値cCM_OverlapRecommendedが選択された場合、推奨されるオーバーラップが内部で計算されます (F_CM_CalculateRecommendedOverlap [▶ 270]を参照)。値cCM_OverlapInactiveは、内部バッファを無効化し、値をゼロに設定します。
- eScalingTypeは、絶対スケーリングが必要な場合に、使用するスケーリング方式 (E_CM_ScalingType [▶ 273]型) を選択できるようにします。デフォルト値はeCM_DiracScalingです。スケーリングを選択する際には、信号が決定論的な信号なのか、確率的な部分を持つ広帯域信号なのか、信号の種類を考慮する必要があります。それぞれ異なるスケーリングを必要とします。
- nChannels は、独立したチャンネルの数を定義します。これはゼロより大きくなければなりません。

● ウィンドウの長さに注意

I nWindowLength の値はnFFT_Length の値以下になるようにしてください。FFTの長さは、必要な周波数分解能に合わせるすることができます。一般的には、FFT長の約4/5の値がウィンドウ長としてよく使用されます。

nFFT_Length がnWindowLengthよりも大きい場合、FFTの周波数分解能が（したがって、戻り値ベクトルの長さも）増加します。長さの差は、フーリエ変換の前にゼロで満たされます。これは、より高い周波数分解能を得るためや、時間領域での逆変換による計算で円形エイリアシングを避けるために有用です。ただし、より高い周波数分解能にもかかわらず、結果にはそれ以上の情報は含まれていません。

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM_Base

5.3.14.17 ST_CM_MeanStressCorrection_InitPars

ファンクションブロック固有の構造体で、ファンクションブロックの初期化時に解析される初期化パラメータを保持します。

```

TYPE ST_CM_MeanStressCorrection_InitPars EXTENDS ST_CM_Object_InitPars :
STRUCT
    eMeanStressCorrection : E_CM_MeanStressCorrection
                        := E_CM_MeanStressCorrection.eCM_Goodman; (* Mean stress correction typ
e, e.g. Goodman or Gerber correction. *)
    nBins                : UDINT := 1;                          (* Number of bins for the str
ess levels. *)
    fBinWidth            : LREAL := 1.0;                        (* Width of the stress bins.
*)
    nBinsMean            : UDINT := 1;                          (* Number of bins for the mea
n levels. *)
    fBinWidthMean        : LREAL := 1.0;                        (* Width of the mean stress b
ins. *)
    fUTS                 : LREAL := 700.0;                      (* Ultimate tensile strength
of material. *)
    nChannels            : UDINT := 1;                          (* Number of channels. *)
END_STRUCT
END_TYPE

```

- eMeanStressCorrection は使用する補正モードを定義します。グッドマン補正とガーバー補正が可能で、補正をしないという選択肢もあります。
- nBins はストレスレベルのビンの数を定義します。ビンの総数はnBins + 2です。この値は1以上である必要があります。各行の最初のビンには、定義された最小値以下のストレスレベルが含まれます。同様に、定義された最大値を超える全ての値は、各行の最後のビンに格納されます。
- fBinWidth はストレスレベルのビンの幅を定義します。このパラメータは、入力されるカウント行列の次元に応じて選択する必要があります。すなわち、ファンクションブロック FB_CMA_RainflowCounting からの出力を補正する場合、ここで値(fMaxStress - fMinStress)/nBinsを選択する必要があります。

- nBinsMean は平均値のビンの数を定義します。値は1以上である必要があります。Halfcycle Count Matrix（ハーフサイクル・カウント・マトリックス）の最初と最後の列に、2つのビンが追加されます。それぞれ、定義された制限値以下の平均値、または制限値を超える平均値が保持されます。
- fBinWidthMean は応力サイクルの平均値のビンの幅を定義します。このパラメータは、入力されるカウント行列の次元に応じて選択する必要があります。すなわち、ファンクションブロック FB_CMA_RainflowCounting からの出力を補正する場合、ここで値(fMaxMean - fMinMean)/nBinsMeanを選択する必要があります。
- fUTS 監視の対象となる材料の引張強さを定義します。
- nChannels は、独立したチャンネルの数を定義します。これはゼロより大きくなければなりません。

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM_Base

5.3.14.18 ST_CM_MinersRule_InitPars

ファンクションブロック固有の構造体で、ファンクションブロックの初期化時に解析される初期化パラメータを保持します。

```

TYPE ST_CM_MinersRule_InitPars EXTENDS ST_CM_Object_InitPars :
STRUCT
  nBins      : UDINT := 64;    (* Number of bins for the stress levels. *)
  nBinsMean  : UDINT := 64;    (* Number of bins for the mean levels. *)
  fDamage    : LREAL := 0.0;   (* Additional constant damage *)
  nCycles    : ULINT := 0;     (* Initial number of cycles counted *)
  nChannels  : UDINT := 1;     (* Number of channels. *)
END_STRUCT
END_TYPE

```

- nBins はストレスレベルのビンの数を定義します。ビンの総数はnBins + 2です。この値は1以上である必要があります。各行の最初のビンには、定義された最小値以下のストレスレベルが含まれます。同様に、定義された最大値を超える全ての値は、各行の最後のビンに格納されます。
- nBinsMean は平均値のビンの数を定義します。値は1以上である必要があります。Halfcycle Count Matrix（ハーフサイクル・カウント・マトリックス）の最初と最後の列に、2つのビンが追加されます。それぞれ、定義された制限値以下の平均値、または制限値を超える平均値が保持されます。
- fDamage は初期状態から一定の損傷を与えることを定義します。このようにして、設定されたウェラー曲線に対するfDamageと累積損傷の合計から、全体の損傷が算出されます。
- nCycles は、カウントされるハーフサイクルの初期値を定義します。総サイクル数は、nCyclesと現在の入力データ数の合計から計算されます。
- nChannels は、独立したチャンネルの数を定義します。これはゼロより大きくなければなりません。

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM_Base

5.3.14.19 ST_CM_MomentCoefficients_InitPars

ファンクションブロック固有の構造体で、ファンクションブロックの初期化時に解析される初期化パラメータを保持します。

```

TYPE ST_CM_MomentCoefficients_InitPars EXTENDS ST_CM_Object_InitPars :
STRUCT
  nChannels      : UDINT := 512;    (* Number of channels. *)
  nSubChannels   : UDINT := 0;      (* Number of subchannels. *)
  nOrder         : E_CM_MCoefOrder := E_CM_MCoefOrder.eCM_Kurtosis;  (* Maximum order of the moment coefficients that are calculated. *)
  bPopulationEstimates : BOOL := FALSE;    (* Apply Bessel's correction to results. *)
  bKurtosisExcess : BOOL := TRUE;    (* Flag if kurtosis

```

```
is reduced by 3. *)
END_STRUCT
END_TYPE
```

- nChannels は、独立したチャンネルの数を定義します。これはゼロより大きくなければなりません。
- nSubChannels は、独立したサブチャンネルの数を定義します。これはゼロ以上でなければなりません。
- nOrder は、計算されるモーメント係数の最大次数 (E_CM_MCoefOrder [▶ 272]) です。これは1から4までの整数でなければなりません。オーダー番号は以下の通りです：0 = カウンタ、1 = 平均値、2 = 標準偏差、3 = 歪度、4 = 尖度。決定された係数の数は(nOrder+1)です。
- bPopulationEstimatesはブール値で、対応するベッセル補正を標本分散、歪度、尖度に適用するかどうかを示します。
- bKurtosisExcess はブール値で、正規分布の尖度の値を差し引いて、過剰尖度を調整するかどうかを指定します。

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM_Base

5.3.14.20 ST_CM_MovingMoments_InitPars

ファンクションブロック固有の構造体で、ファンクションブロックの初期化時に解析される初期化パラメータを保持します。

```
TYPE ST_CM_MovingMoments_InitPars EXTENDS ST_CM_Object_InitPars :
STRUCT
    nChannels          : UDINT := 1;          (* Number of channels. *)
    nSubChannels       : UDINT := 0;          (* Number of subchannels. *)
    nInLength          : UDINT := 100;        (* Length of input data array. *)
    nPopulation        : UDINT := 100;        (* Size of the population. *)
    bPopulationEstimates : BOOL := TRUE;      (* Apply Bessel's correction to results. *)
END_STRUCT
END_TYPE
```

- nChannels は、独立したチャンネルの数を定義します。これはゼロより大きくなければなりません。
- nSubChannels は、独立したサブチャンネルの数を定義します。これはゼロ以上でなければなりません。
- nInLength は入力データ配列の長さを定義します。
- nPopulation 移動モーメントを計算するための内部データバッファのサイズを定義します。
- bPopulationEstimatesはブール値で、対応するベッセル補正を標本分散、歪度、尖度に適用するかどうかを示します。
- bKurtosisExcess はブール値で、正規分布の尖度の値を差し引いて、過剰尖度を調整するかどうかを指定します。

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM_Base

5.3.14.21 ST_CM_MultiBandRMS_InitPars

ファンクションブロック固有の構造体で、ファンクションブロックの初期化時に解析される初期化パラメータを保持します。

```
TYPE ST_CM_MultiBandRMS_InitPars EXTENDS ST_CM_Object_InitPars :
STRUCT
    nFFT_Length        : UDINT := 512;          (* Length of FFT. *)
    nWindowLength      : UDINT := 400;          (* Length of analysis window. *)
    *)
    fSampleRate        : LREAL := 20000;        (* Sample rate in Hertz. *)
    nMaxBands          : UDINT := 10;           (* Maximum number of frequency bands. *)
END_STRUCT
```



```

nChannels      : UDINT := 10;          (* Number of channels. *)
eWindowType    : E_CM_WindowType := eCM_HannWindow;  (* Window type. *)
aWindowParameters : T_CM_WindowParameters := [2.5,1,1,1,1]; (* Window parameters for spec
ific windows, e.g. FlatTop. *)
nOverlap       : UDINT := -1;          (* Size of overlap in samples
. *)
bTransformToDecibel : BOOL := TRUE;    (* Transform to decibel. *)
fDecibelThreshold : LREAL := cCM_MinArgLog10; (* Minimum argument of decadi
c logarithm for 64-bit IEEE 754 arithmetic. *)
END_STRUCT
END_TYPE

```

- `nFFT_Length` はFFTの長さです。1より大きく、2の累乗である必要があります。
- `nWindowLength` は解析ウィンドウの長さ（サンプル数）です。長さは1より大きく、偶数である必要があります。
- `fSampleRate` 入力される時間信号のサンプリング・レート。この値は、Hz単位での結果のスケールに使用されます。
- `nMaxBands` は、RMS値を計算する周波数帯域の最大数を指定します。
- `nChannels` は、独立したチャンネルの数を定義します。これはゼロより大きくなければなりません。
- `eWindowType` 使用するウィンドウ関数（`E_CM_WindowType` [▶ 274]型）を定義します。良いデフォルト値はウィンドウタイプ`eCM_HannWindow` です。
- `aWindowParameters` には、選択したウィンドウ関数のフリーパラメータが含まれます。`eCM_KaiserWindow`を使用する場合、最初のエントリはパラメータ β を定義します。`eCM_FlatTopWindow`を使用する場合、全てのパラメータが使用されます。[ウィンドウ関数 \[▶ 19\]](#)を参照してください。
- `nOverlap` はオーバーラップするサンプル数を定義します。これはゼロ以上でなければなりません。値`cCM_OverlapRecommended`が選択された場合、推奨されるオーバーラップが内部で計算されます（`F_CM_CalculateRecommendedOverlap` [▶ 270]を参照）。値`cCM_OverlapInactive`は、内部バッファを無効化し、値をゼロに設定します。
- `bTransformToDecibel` は、FFTの結果をデシベル単位に変換するかどうかを示すブール値で、変換 $x \rightarrow 20 * \log_{10}(x)$ に従って処理されます。
- `fDecibelThreshold` は非常に小さな浮動小数点値で、ゼロより大きな値です。この数値より小さい値は、デシベルスケールへの変換前にこの値に置き換えられます。（その目的は、値域エラーの回避です。ゼロの対数は定義されておらず、小さな引数の極限では値が負の無限大に発散します。数字のゼロの引数、`arg(0)`も同様です。可能な最小値は $2.3e-308$ で、これは定数`cCM_MinArgLog10`と同じ値です。）

● ウィンドウの長さに注意

i `nWindowLength` の値は`nFFT_Length` の値以下になるようにしてください。FFTの長さは、必要な周波数分解能に合わせることができます。一般的には、FFT長の約4/5の値がウィンドウ長としてよく使用されます。

`nFFT_Length` が`nWindowLength`よりも大きい場合、FFTの周波数分解能が（したがって、戻り値ベクトルの長さも）増加します。長さの差は、フーリエ変換の前にゼロで満たされます。これは、より高い周波数分解能を得るためや、時間領域での逆変換による計算で円形エイリアシングを避けるために有用です。ただし、より高い周波数分解能にもかかわらず、結果にはそれ以上の情報は含まれていません。

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86, x64)	Tc3_CM_Base

5.3.14.22 ST_CM_Normalization_InitPars

ファンクションブロック固有の構造体で、ファンクションブロックの初期化時に解析される初期化パラメータを保持します。

```

TYPE ST_CM_Normalization_InitPars EXTENDS ST_CM_Object_InitPars :
STRUCT
  nChannels      : UDINT := 1;          (* Number of channels. *)
  nSubChannels    : UDINT := 0;          (* Number of subchannels. *)
  nInLength       : UDINT := 256;       (* Length of input data array. *)

```

```

fParameter_s      : LREAL := 1.0;  (* Parameter s for normalization. *)
nParameter_p      : UDINT := 1;    (* Parameter p for normalization. *)
nParameter_q      : UDINT := 1;    (* Parameter q for normalization. *)
nWorkingDimension : UDINT := 0;    (* Dimension index of MultiArray for normalization. *)
bAbsoluteValues   : BOOL := TRUE;  (* Use absolute values in calculation. *)
END_STRUCT
END_TYPE

```

- nChannels は、独立したチャンネルの数を定義します。これはゼロより大きくなければなりません。
- nSubChannels は、独立したサブチャンネルの数を定義します。これはゼロ以上でなければなりません。
- nInLength は、入力配列の長さです。
- fParameter_s は、正規化のスケーリング係数を定義します。
- fParameter_p は、正規化における和の指数を定義します。
- fParameter_q は、正規化における外側の指数の比を定義します。
- nWorkingDimension は、総和／正規化を行う際の入力マルチアレイのゼロ基準の次元を指定します。
- bAbsoluteValues は、個々の値を合計に加算するかどうかを選択するためのフラグです。

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM, Tc3_CM_Base

5.3.14.23 ST_CM_OrderPowerSpectrum_InitPars

ファンクションブロック固有の構造体で、ファンクションブロックの初期化時に解析される初期化パラメータを保持します。

```

TYPE ST_CM_OrderPowerSpectrum_InitPars EXTENDS ST_CM_Object_InitPars :
STRUCT
  fSampleRateSignal : LREAL := 1000;          (* Sample rate in Hertz of vibration signal. *)
  fSampleRatePosition : LREAL := 1000;        (* Sample rate in Hertz of positions. *)
  fPositionScaling   : LREAL := 1;            (* One rotation needs to be represented by 1. *)
  fMaxRPM            : LREAL := 1800;        (* Max. rotations per minute. *)
  nFFT_Length        : UDINT := 512;          (* Length of FFT. *)
  nWindowLength      : UDINT := 400;          (* Length of analysis window. *)
  fDecibelThreshold  : LREAL := cCM_MinArgLog10; (* Minimum argument of decadic logarithm for 64-bit IEEE 754 arithmetic. *)
  bTransformToDecibel : BOOL := TRUE;         (* Transform to decibel. *)
  eWindowType        : E_CM_WindowType := eCM_HannWindow; (* Window type. *)
  aWindowParameters  : T_CM_WindowParameters := [2.5,1,1,1,1]; (* Window parameters for specific windows, e.g. FlatTop. *)
  nOverlap           : UDINT := -1;           (* Size of overlap in samples. *)
  eScalingType       : E_CM_ScalingType := eCM_DiracScaling; (* Scaling type. *)
  nChannels          : UDINT := 1;            (* Number of channels. *)
END_STRUCT
END_TYPE

```

- fSampleRateSignal は、入力される振動信号のサンプリングレートをヘルツで定義します。
- fSampleRatePosition は、入力される位置信号のサンプリングレートをヘルツで定義する。これは fSampleRateSignal と同じではない可能性があります。
- fPositionScaling は位置信号のスケーリング係数です。全回転はこの値で表す必要があります。位置信号が度単位でスケーリングされる場合は、fPositionScaling = 360 を設定します。位置信号が、シャフトの一回転で値 2π を生成するようにスケーリングされている場合は fPositionScaling = 2 * π です。
- fMaxRPM は1分間の最大回転数です。fSampleRateSignal の場合、このパラメータは次数スペクトルで分解可能な最大次数を定義します。

その他のパラメータは全て、パワースペクトルのパラメータ設定と同様です (ST_CM_PowerSpectrum_InitPars ▶ 295) を参照)。

- `nFFT_Length` はFFTの長さです。1より大きく、2の累乗である必要があります。
- `nWindowLength` は解析ウィンドウの長さ（サンプル数）です。長さは1より大きく、偶数である必要があります。
- `fDecibelThreshold` は非常に小さな浮動小数点値で、ゼロより大きな値です。この数値より小さい値は、デシベルスケールへの変換前にこの値に置き換えられます。（その目的は、値域エラーの回避です。ゼロの対数は定義されておらず、小さな引数の極限では値が負の無限大に発散します。数字のゼロの引数、`arg(0)`も同様です。可能な最小値は`2.3e-308`で、これは定数`cCM_MinArgLog10`と同じ値です。）
- `bTransformToDecibel` は、FFTの結果をデシベル単位に変換するかどうかを示すブール値で、変換 $x \rightarrow 20 * \log_{10}(x)$ に従って処理されます。
- `eWindowType` 使用するウィンドウ関数（`E_CM_WindowType` [▶ 274]型）を定義します。良いデフォルト値はウィンドウタイプ`eCM_HannWindow`です。
- `aWindowParameters` には、選択したウィンドウ関数のフリーパラメータが含まれます。
`eCM_KaiserWindow`を使用する場合、最初のエントリはパラメータ β を定義します。`eCM_FlatTopWindow`を使用する場合、全てのパラメータが使用されます。ウィンドウ関数 [▶ 19]を参照してください。
- `nOverlap` はオーバーラップするサンプル数を定義します。これはゼロ以上でなければなりません。値`cCM_OverlapRecommended`が選択された場合、推奨されるオーバーラップが内部で計算されます（`F_CM_CalculateRecommendedOverlap` [▶ 270]を参照）。値`cCM_OverlapInactive`は、内部バッファを無効化し、値をゼロに設定します。
- `eScalingType`は、絶対スケーリングが必要な場合に、使用するスケーリング方式（`E_CM_ScalingType` [▶ 273]型）を選択できるようにします。デフォルト値は`eCM_DiracScaling`です。スケーリングを選択する際には、信号が決定論的な信号なのか、確率的な部分を持つ広帯域信号なのか、信号の種類を考慮する必要があります。それぞれ異なるスケーリングを必要とします。
- `nChannels` は、独立したチャンネルの数を定義します。これはゼロより大きくなければなりません。

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM_Base

5.3.14.24 ST_CM_PowerCepstrum_InitPars

ファンクションブロック固有の構造体で、ファンクションブロックの初期化時に解析される初期化パラメータを保持します。

```

TYPE ST_CM_PowerCepstrum_InitPars EXTENDS ST_CM_Object_InitPars :
STRUCT
    nFFT_Length      : UDINT := 512;                (* Length of FFT. *)
    nWindowLength    : UDINT := 400;                (* Length of analysis window. *)
    eWindowType      : E_CM_WindowType := eCM_HannWindow; (* Window type. *)
    aWindowParameters : T_CM_WindowParameters := [2.5,1,1,1,1]; (* Window parameters for specific windows, e.g. FlatTop. *)
    nOverlap         : UDINT := -1;                 (* Size of overlap in samples. *)
)
    fLogThreshold     : LREAL := cCM_MinArgLogN;    (* Minimum argument of decadic logarithm for 64-bit IEEE 754 arithmetic *)
    eScalingType      : E_CM_ScalingType := eCM_DiracScaling; (* Scaling type. *)
    nChannels         : UDINT := 1;                 (* Number of channels. *)
*)
END_STRUCT
END_TYPE

```

- `nFFT_Length` はFFTの長さです。1より大きく、2の累乗である必要があります。
- `nWindowLength` は解析ウィンドウの長さ（サンプル数）です。長さは1より大きく、偶数である必要があります。
- `eWindowType` 使用するウィンドウ関数（`E_CM_WindowType` [▶ 274]型）を定義します。良いデフォルト値はウィンドウタイプ`eCM_HannWindow`です。
- `aWindowParameters` には、選択したウィンドウ関数のフリーパラメータが含まれます。
`eCM_KaiserWindow`を使用する場合、最初のエントリはパラメータ β を定義します。`eCM_FlatTopWindow`を使用する場合、全てのパラメータが使用されます。ウィンドウ関数 [▶ 19]を参照してください。

- `nOverlap` はオーバーラップするサンプル数を定義します。これはゼロ以上でなければなりません。値 `cCM_OverlapRecommended` が選択された場合、推奨されるオーバーラップが内部で計算されます (`F_CM_CalculateRecommendedOverlap` [▶ 270] を参照)。値 `cCM_OverlapInactive` は、内部バッファを無効化し、値をゼロに設定します。
- `fLogThreshold` は非常に小さな浮動小数点値で、ゼロより大きな値です。可能な最小値は $3.75e-324$ であり、これは定数 `cCM_MinArgLogN` と等価です。
この数値よりも絶対値が小さいスペクトル値は、スペクトルが対数化される前にこの値に置き換えられます。その目的は、値域誤差の回避です。ゼロの対数は定義されておらず、小さな引数の極限では値が負の無限大に発散します。数字のゼロの引数、`arg(0)` も同様です。
- `eScalingType` は、絶対スケーリングが必要な場合に、使用するスケーリング方式 (`E_CM_ScalingType` [▶ 273] 型) を選択できるようにします。デフォルト値は `eCM_DiracScaling` です。スケーリングを選択する際には、信号が決定論的な信号なのか、確率的な部分を持つ広帯域信号なのか、信号の種類を考慮する必要があります。それぞれ異なるスケーリングを必要とします。
- `nChannels` は、独立したチャンネルの数を定義します。これはゼロより大きくなければなりません。

● ウィンドウの長さ

I `nFFT_Length` は、`nWindowLength` の値以上でなければなりません。FFT の長さは、必要な周波数分解能に合わせることができます。一般的には、FFT 長の約 $3/4$ の値がウィンドウ長としてよく使用されます。

`nFFT_Length` が `nWindowLength` よりも大きい場合、FFT の周波数分解能が（したがって、戻り値ベクトルの長さも）増加します。長さの差は、フーリエ変換の前にゼロで満たされます。これは、より高い周波数分解能を得るためや、例えば時間領域で逆変換を使って計算する場合に、円形エイリアシングを回避するために有効です。しかし、より高い周波数分解能にもかかわらず、結果にはそれ以上の情報は含まれていません。

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM_Base

5.3.14.25 ST_CM_PowerSpectrum_InitPars

ファンクションブロック固有の構造体で、ファンクションブロックの初期化時に解析される初期化パラメータを保持します。

```

TYPE ST_CM_PowerSpectrum_InitPars EXTENDS ST_CM_Object_InitPars :
STRUCT
    nFFT_Length      : UDINT := 512;                (* Length of FFT. *)
    nWindowLength    : UDINT := 400;                (* Length of analysis window. *)
*)
    fDecibelThreshold : LREAL := cCM_MinArgLog10;    (* Minimum argument of decadic
logarithm for 64-bit IEEE 754 arithmetic. *)
    bTransformToDecibel : BOOL := TRUE;            (* Transform to decibel. *)
    eWindowType       : E_CM_WindowType := eCM_HannWindow; (* Window type. *)
    aWindowParameters : T_CM_WindowParameters := [2.5,1,1,1,1]; (* Window parameters for speci
fic windows, e.g. FlatTop. *)
    nOverlap          : UDINT := -1;                (* Size of overlap in samples. *)
*)
    eScalingType      : E_CM_ScalingType := eCM_DiracScaling; (* Scaling type. *)
    nChannels         : UDINT := 1;                 (* Number of channels. *)
END_STRUCT
END_TYPE

```

- `nFFT_Length` はFFTの長さです。1より大きく、2の累乗である必要があります。
- `nWindowLength` は解析ウィンドウの長さ（サンプル数）です。長さは1より大きく、偶数である必要があります。
- `fDecibelThreshold` は非常に小さな浮動小数点値で、ゼロより大きな値です。この数値より小さい値は、デシベルスケールへの変換前にこの値に置き換えられます。（その目的は、値域エラーの回避です。ゼロの対数は定義されておらず、小さな引数の極限では値が負の無限大に発散します。数字のゼロの引数、`arg(0)` も同様です。可能な最小値は $2.3e-308$ で、これは定数 `cCM_MinArgLog10` と同じ値です。）

- `bTransformToDecibel` は、FFTの結果をデシベル単位に変換するかどうかを示すブール値で、変換 $x \rightarrow 20 * \log_{10}(x)$ に従って処理されます。
- `eWindowType` 使用するウィンドウ関数 (`E_CM_WindowType` [▶ 274]型) を定義します。良いデフォルト値はウィンドウタイプ `eCM_HannWindow` です。
- `aWindowParameters` には、選択したウィンドウ関数のフリーパラメータが含まれます。
`eCM_KaiserWindow`を使用する場合、最初のエントリはパラメータ β を定義します。`eCM_FlatTopWindow`を使用する場合、全てのパラメータが使用されます。ウィンドウ関数 [▶ 19]を参照してください。
- `nOverlap` はオーバーラップするサンプル数を定義します。これはゼロ以上でなければなりません。値 `cCM_OverlapRecommended`が選択された場合、推奨されるオーバーラップが内部で計算されます (`F_CM_CalculateRecommendedOverlap` [▶ 270]を参照)。値 `cCM_OverlapInactive`は、内部バッファを無効化し、値をゼロに設定します。
- `eScalingType`は、絶対スケーリングが必要な場合に、使用するスケーリング方式 (`E_CM_ScalingType` [▶ 273]型) を選択できるようにします。デフォルト値は `eCM_DiracScaling` です。スケーリングを選択する際には、信号が決定論的な信号なのか、確率的な部分を持つ広帯域信号なのか、信号の種類を考慮する必要があります。それぞれ異なるスケーリングを必要とします。
- `nChannels` は、独立したチャンネルの数を定義します。これはゼロより大きくなければなりません。

● ウィンドウの長さに注意

1 `nWindowLength` の値は `nFFT_Length` の値以下になるようにしてください。FFTの長さは、必要な周波数分解能に合わせるすることができます。一般的には、FFT長の約4/5の値がウィンドウ長としてよく使用されます。

`nFFT_Length` が `nWindowLength` よりも大きい場合、FFTの周波数分解能が（したがって、戻り値ベクトルの長さも）増加します。長さの差は、フーリエ変換の前にゼロで満たされます。これは、より高い周波数分解能を得るためや、時間領域での逆変換による計算で円形エイリアシングを避けるために有用です。ただし、より高い周波数分解能にもかかわらず、結果にはそれ以上の情報は含まれていません。

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86, x64)	Tc3_CM_Base

5.3.14.26 ST_CM_Quantiles_InitPars

ファンクションブロック固有の構造体で、ファンクションブロックの初期化時に解析される初期化パラメータを保持します。

```

TYPE ST_CM_Quantiles_InitPars EXTENDS ST_CM_Object_InitPars :
STRUCT
  nChannels      : UDINT := 512;      (* Number of channels. *)
  nSubChannels   : UDINT := 0;        (* Number of subchannels. *)
  fMinBinned     : LREAL := -120;     (* Minimum binned value. *)
  fMaxBinned     : LREAL := 100;      (* Maximum binned value. *)
  nBins          : UDINT := 100;      (* Number of bins. *)
  nMaxQuantiles  : UDINT := 10;       (* Maximum number of quantiles. *)
END_STRUCT
END_TYPE

```

- `nChannels` は、独立したチャンネルの数を定義します。これはゼロより大きくなければなりません。
- `nSubChannels` は、独立したサブチャンネルの数を定義します。これはゼロ以上でなければなりません。
- `fMinBinned` は、通常ヒストグラムビンでサンプルをカウントする際の下限值です。
- `fMaxBinned` は、通常ヒストグラムビンでサンプルをカウントする際の上限值です。`fMaxBinned`は、`fMinBinned`よりも大きくする必要があります。
- `nBins` はヒストグラムのビンの数です。これは1以上である必要があります。多くの場合、10と20の間の値を選択するのが合理的です。この値には、`fMinBinned` 以下または `fMaxBinned` 以上の値を示す2つの特別なビンは含まれません。
- `nMaxQuantiles` は、各チャンネルについて計算される分位の数です。これはゼロ以上の整数でなければなりません。

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM_Base

5.3.14.27 ST_CM_RainflowCounting_InitPars

ファンクションブロック固有の構造体で、ファンクションブロックの初期化時に解析される初期化パラメータを保持します。

```

TYPE ST_CM_RainflowCounting_InitPars EXTENDS ST_CM_Object_InitPars :
STRUCT
  nBins      : UDINT := 64;      (* Number of bins for the stress levels. *)
  fminStress : LREAL := -40.0;   (* Lower limit of the stress bins. *)
  fmaxStress : LREAL := 40.0;    (* Upper limit of the stress bins. *)
  nBinsMean  : UDINT := 64;      (* Number of bins for the mean levels. *)
  fminMean   : LREAL := -40.0;   (* Lower limit of bins for mean stress values. *)
  fmaxMean   : LREAL := 40.0;    (* Upper limit of bins for mean stress values. *)
  nChannels  : UDINT := 1;       (* Number of channels. *)
END_STRUCT
END_TYPE

```

- nBins はストレスレベルのビンの数を定義します。ビンの総数はnBins + 2です。この値は1以上である必要があります。各行の最初のビンには、定義された最小値以下のストレスレベルが含まれます。同様に、定義された最大値を超える全ての値は、各行の最後のビンに格納されます。
- fminStress はストレスレベルの下限を定義します。この値以下の値は全てビンに累積されます。
- fmaxStress はストレスレベルの上限を定義します。最大値を超える値は、最後のビンでカウントされます。
- nBinsMean は平均値のビンの数を定義します。値は1以上である必要があります。*Halfcycle Count Matrix* (ハーフサイクル・カウント・マトリックス) の最初と最後の列に、2つのビンが追加されます。それぞれ、定義された制限値以下の平均値、または制限値を超える平均値が保持されます。
- fminMean は平均値の下限を定義します。この値以下の値は全てビンに累積されます。
- fmaxMean は平均値の上限を定義します。最大値を超える値は、最後のビンでカウントされます。
- nChannels は、独立したチャンネルの数を定義します。これはゼロより大きくなければなりません。

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM_Base

5.3.14.28 ST_CM_RealFFT_InitPars

ファンクションブロック固有の構造体で、ファンクションブロックの初期化時に解析される初期化パラメータを保持します。

```

TYPE ST_CM_RealFFT_InitPars EXTENDS ST_CM_Object_InitPars :
STRUCT
  nFFT_Length : UDINT := 512;    (* Length of FFT. *)
  bForward     : BOOL := TRUE;   (* Flag indicating forward FFT. *)
  bHalfSpec    : BOOL := TRUE;   (* Flag indicating length of output; if TRUE, the algorithm outputs half the spectrum (nFFT_Length/2 + 1). *)
  nChannels    : UDINT := 1;     (* Number of channels. *)
END_STRUCT
END_TYPE

```

- nFFT_Length はFFTの長さです。1より大きく、2の累乗である必要があります。
- bForward はFFTの方向を指定するブール型のパラメータです。値がTRUE の場合、通常のFFTが計算されます。そうでない場合は、逆FFTが使用されます。
- bHalfSpec は、結果バッファのサイズを指定するブール型のパラメータです。値がTRUE の場合、アルゴリズムはハーフスペクトル(nFFT_Length/2 + 1)を出力します。
- nChannels は、独立したチャンネルの数を定義します。これはゼロより大きくなければなりません。

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM_Base

5.3.14.29 ST_CM_RMS_InitPars

ファンクションブロック固有の構造体で、ファンクションブロックの初期化時に解析される初期化パラメータを保持します。

```

TYPE ST_CM_RMS_InitPars EXTENDS ST_CM_Object_InitPars :
STRUCT
  nChannels          : UDINT := 4;          (* Number of channels. *)
  nSubChannels       : UDINT := 0;          (* Number of subchannels. *)
  nBufferLength      : UDINT := 2000;       (* Buffer length. *)
  fDecibelThreshold  : LREAL := cCM_MinArgLog10; (* Minimum argument of decadic logarithm for
64-bit IEEE 754 arithmetic. *)
  bTransformToDecibel : BOOL := TRUE;       (* Transform to decibel. *)
END_STRUCT
END_TYPE

```

- nChannels は、独立したチャンネルの数を定義します。これはゼロより大きくなければなりません。
- nSubChannels は、独立したサブチャンネルの数を定義します。これはゼロ以上でなければなりません。
- nBufferLength は、内部バッファに保持されているチャンネルごとの入力値の数です。
- fDecibelThreshold は非常に小さな浮動小数点値で、ゼロより大きな値です。この数値より小さい値は、デシベルスケールへの変換前にこの値に置き換えられます。(その目的は、値域エラーの回避です。ゼロの対数は定義されておらず、小さな引数の極限では値が負の無限大に発散します。数字のゼロの引数、arg(0)も同様です。可能な最小値は2.3e-308で、これは定数cCM_MinArgLog10 と同じ値です。)
- bTransformToDecibel は、FFTの結果をデシベル単位に変換するかどうかを示すブール値で、変換 $x \rightarrow 20 * \log_{10}(x)$ に従って処理されます。

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM_Base

5.3.14.30 ST_CM_SlidingDFT_InitPars

ファンクションブロック固有の構造体で、ファンクションブロックの初期化時に解析される初期化パラメータを保持します。

```

TYPE ST_CM_SlidingDFT_InitPars EXTENDS ST_CM_Object_InitPars :
STRUCT
  nWindowLength      : UDINT := 1000;      (* Length of analysis window. *)
  nBins              : UDINT := 1;          (* Number of spectral bins. *)
  fDampingFactor     : LREAL := 0.995;     (* Damping factor for SDFT. *)
  nChannels          : UDINT := 1;          (* Number of channels. *)
END_STRUCT
END_TYPE

```

- nWindowLength は解析ウィンドウの長さ（サンプル数）です。長さは1より大きく、偶数である必要があります。
- nBins は計算するスペクトル値の数です。対応するインデックス($k := f / f_{\text{SampleRate}} / n_{\text{WindowLength}}$)は、初期化後にConfigure() メソッドを使って設定する必要があります。
- fDampingFactor は、再帰式の減衰パラメータの値を定義します。
- nChannels は、独立したチャンネルの数を定義します。これはゼロより大きくなければなりません。

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM_Base

5.3.14.31 ST_CM_SparseSpectrum_InitPars

ファンクションブロック固有の構造体で、ファンクションブロックの初期化時に解析される初期化パラメータを保持します。

```

TYPE ST_CM_SparseSpectrum_InitPars EXTENDS ST_CM_Object_InitPars :
STRUCT
  nWindowLength      : UDINT := 1000;                      (* Length of analysis window. *)
  fSampleRate         : LREAL := 10000;                    (* Sample rate in Hertz. *)
  nBins               : UDINT := 1;                        (* Number of spectral bins. *)
  eSpectrumType       : E_CM_SpectrumType := E_CM_SpectrumType.eCM_Magnitude; (* Type of spectrum, i.e. plain DFT, magnitude, power. *)
  eWindowType         : E_CM_WindowType := eCM_HannWindow; (* Window type. *)
  aWindowParameters   : T_CM_WindowParameters := [2.5,1,1,1,1]; (* Window parameters for specific windows, e.g. FlatTop. *)
  nOverlap            : UDINT := -1;                       (* Size of overlap in samples. *)
  eScalingType        : E_CM_ScalingType := eCM_DiracScaling; (* Scaling type. *)
  bTransformToDecibel : BOOL := FALSE;                    (* Transform to decibel. *)
  fDecibelThreshold   : LREAL := cCM_MinArgLog10;          (* Minimum argument of decadic logarithm for 64-bit IEEE 754 arithmetic. *)
  nChannels            : UDINT := 1;                       (* Number of channels. *)
END_STRUCT
END_TYPE

```

- **nWindowLength** は解析ウィンドウの長さ（サンプル数）です。長さは1より大きく、偶数である必要があります。
- **fSampleRate** 入力される時間信号のサンプリング・レート。この値は、Hz単位での結果のスケールングに使用されます。
- **nBins** は計算するスペクトル値の数です。対応するインデックス($k := f / fSampleRate / nWindowLength$)は、初期化後にConfigure() メソッドを使って設定する必要があります。
- **eSpectrumType** は、計算するスペクトル値（[E_CM_SpectrumType \[▶ 273\]](#)型）のタイプを定義します。値のスケールングは、それぞれのタイプに合わせて調整され、値がファンクションブロック [FB_CMA_RealFFT \[▶ 203\]](#)、[FB_CMA_MagnitudeSpectrum \[▶ 161\]](#) または [FB_CMA_PowerSpectrum \[▶ 189\]](#) の結果と一致ようになります。
- **eWindowType** 使用するウィンドウ関数（[E_CM_WindowType \[▶ 274\]](#)型）を定義します。良いデフォルト値はウィンドウタイプeCM_HannWindow です。
- **aWindowParameters** には、選択したウィンドウ関数のフリーパラメータが含まれます。eCM_KaiserWindowを使用する場合、最初のエントリはパラメータβを定義します。eCM_FlatTopWindowを使用する場合、全てのパラメータが使用されます。[ウィンドウ関数 \[▶ 19\]](#)を参照してください。
- **nOverlap** はオーバーラップするサンプル数を定義します。これはゼロ以上でなければなりません。値cCM_OverlapRecommendedが選択された場合、推奨されるオーバーラップが内部で計算されます（[F_CM_CalculateRecommendedOverlap \[▶ 270\]](#)を参照）。値cCM_OverlapInactiveは、内部バッファを無効化し、値をゼロに設定します。
- **eScalingType**は、絶対スケールが必要な場合に、使用するスケール方式（[E_CM_ScalingType \[▶ 273\]](#)型）を選択できるようにします。デフォルト値はeCM_DiracScalingです。スケール方式を選択するには、信号が決定論的な信号なのか、確率的な部分を持つ広帯域信号なのか、信号の種類を考慮する必要があります。それぞれ異なるスケール方式を必要とします。
- **bTransformToDecibel** は、FFTの結果をデシベル単位に変換するかどうかを示すブール値で、変換 $x \rightarrow 20 * \log_{10}(x)$ に従って処理されます。
- **fDecibelThreshold** は非常に小さな浮動小数点値で、ゼロより大きな値です。この数値より小さい値は、デシベルスケールへの変換前にこの値に置き換えられます。（その目的は、値域エラーの回避です。ゼロの対数は定義されておらず、小さな引数の極限では値が負の無限大に発散します。数字のゼロの引数、arg(0)も同様です。可能な最小値は2.3e-308で、これは定数cCM_MinArgLog10 と同じ値です。）
- **nChannels** は、独立したチャンネルの数を定義します。これはゼロより大きくなければなりません。

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM_Base

5.3.14.32 ST_CM_SpikeEnergySpectrum_InitPars

ファンクションブロック固有の構造体で、ファンクションブロックの初期化時に解析される初期化パラメータを保持します。

```

TYPE ST_CM_SpikeEnergySpectrum_InitPars EXTENDS ST_CM_Object_InitPars :
STRUCT
    nFFT_Length          : UDINT := 1024;                (* Length of FFT
. *)
    nWindowLength        : UDINT := 800;                (* Length of ana
lysis window. *)
    fSampleRate           : LREAL := 10000.0;           (* Sample rate i
n Hertz. *)
    fDecayTime            : LREAL := 0.08;              (* Decay time co
nstant in seconds. *)
    fLowerFrequencyLimit  : LREAL := 500.0;             (* Lower limit o
f frequency band in Hertz. *)
    fUpperFrequencyLimit  : LREAL := 4950.0;           (* Upper limit o
f frequency band in Hertz. *)
    nOrder                : UDINT := 4;                (* Order of band
pass filter in range [1 .. 10]. *)
    eWindowType           : E_CM_WindowType := E_CM_WindowType.eCM_HannWindow; (* Window type.
*)
    aWindowParameters     : T_CM_WindowParameters := [2.5, 1.0, 1.0, 1.0, 1.0]; (* Window parame
ters for specific windows, e.g. FlatTop. *)
    nOverlap              : UDINT := 4294967295;        (* Size of overl
ap in samples. *)
    eScalingType          : E_CM_ScalingType := E_CM_ScalingType.eCM_RootPowerSum; (* Scaling type.
*)
    bTransformToDecibel   : BOOL := FALSE;              (* Transform to
decibel. *)
    fDecibelThreshold      : LREAL := 2.3e-308;         (* Minimum argum
ent of decadic logarithm for 64- bit IEEE 754 arithmetic. *)
    nChannels             : UDINT := 1;                 (* Number of cha
nnels. *)
END_STRUCT
END_TYPE

```

- nFFT_Length はFFTの長さです。1より大きく、2の累乗である必要があります。
- nWindowLength は解析ウィンドウの長さ（サンプル数）です。長さは1より大きく、偶数である必要があります。
- fSampleRate 入力される時間信号のサンプリング・レート。この値は、Hz単位での結果のスケーリングに使用されます。
- fDecayTime は時間波形の減衰時間を定義します（スパイク関数）。算出されるスパイクの精度を向上させるため、Configure() 方式で値を設定できます。
- fLowerFrequencyLimit は、内部で使用するバンドパス（帯域幅フィルタ）の下限值です。この値は、予想されるマシンの振動よりも高くする必要があります。
- fUpperFrequencyLimit は、内部で使用するバンドパス（帯域幅フィルタ）の上限値です。高周波の衝撃を識別できるようにするため、できるだけ高い値を選択します。
- nOrder は使用するIIRフィルタの次数を定義します。
- eWindowType 使用するウィンドウ関数（E_CM_WindowType [▶ 274]型）を定義します。良いデフォルト値はウィンドウタイプeCM_HannWindow です。
- aWindowParameters には、選択したウィンドウ関数のフリーパラメータが含まれます。eCM_KaiserWindowを使用する場合、最初のエントリはパラメータβを定義します。eCM_FlatTopWindowを使用する場合、全てのパラメータが使用されます。ウィンドウ関数 [▶ 19]を参照してください。
- nOverlap はオーバーラップするサンプル数を定義します。これはゼロ以上でなければなりません。値cCM_OverlapRecommendedが選択された場合、推奨されるオーバーラップが内部で計算されます（F_CM_CalculateRecommendedOverlap [▶ 270]を参照）。値cCM_OverlapInactiveは、内部バッファを無効化し、値をゼロに設定します。

- eScalingTypeは、絶対スケーリングが必要な場合に、使用するスケーリング方式（E_CM_ScalingType [▶ 273]型）を選択できるようにします。デフォルト値はeCM_DiracScalingです。スケーリングを選択する際には、信号が決定論的な信号なのか、確率的な部分を持つ広帯域信号なのか、信号の種類を考慮する必要があります。それぞれ異なるスケーリングを必要とします。
- bTransformToDecibel は、FFTの結果をデシベル単位に変換するかどうかを示すブール値で、変換 $x \rightarrow 20 \cdot \log_{10}(x)$ に従って処理されます。
- fDecibelThreshold は非常に小さな浮動小数点値で、ゼロより大きな値です。この数値より小さい値は、デシベルスケールへの変換前にこの値に置き換えられます。（その目的は、値域エラーの回避です。ゼロの対数は定義されておらず、小さな引数の極限では値が負の無限大に発散します。数字のゼロの引数、arg(0)も同様です。可能な最小値は2.3e-308で、これは定数cCM_MinArgLog10 と同じ値です。）
- nChannels は、独立したチャンネルの数を定義します。これはゼロより大きくなければなりません。

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM_Base

5.3.14.33 ST_CM_VibrationAssessment_InitPars

ファンクションブロック固有の構造体で、ファンクションブロックの初期化時に解析される初期化パラメータを保持します。

```

TYPE ST_CM_VibrationAssessment_InitPars EXTENDS ST_CM_Object_InitPars :
STRUCT
    nFFT_Length          : UDINT := 512;                (* Length of FFT. *)
    nWindowLength        : UDINT := 400;                (* Length of analysis window. *)
    *)
    fSampleRate          : LREAL := 20000;              (* Sample rate in Hertz. *)
    nChannels            : UDINT := 2;                  (* Number of channels. *)
    nOrder               : UDINT := 2;                  (* Maximum order of integrati
on: 0 = acceleration, 1 = velocity, 2 = place. *)
    nMaxBands            : UDINT := 1;                  (* Maximum number of frequenc
y bands. *)
    nMaxClasses          : UDINT := 3;                  (* Number of configurable thr
eshold classes. *)
    bMemorize            : BOOL := TRUE;                (* Flag if results are memori
zed. *)
    eWindowType          : E_CM_WindowType := eCM_HannWindow; (* Window type. *)
    aWindowParameters    : T_CM_WindowParameters := [2.5,1,1,1,1]; (* Window parameters for spec
ific windows, e.g. FlatTop. *)
    nOverlap             : UDINT := -1;                 (* Size of overlap in samples
. *)
    bTransformToDecibel   : BOOL := TRUE;                (* Transform to decibel. *)
    fDecibelThreshold     : LREAL := cCM_MinArgLog10;    (* Minimum argument of decadi
c logarithm for 64-bit IEEE 754 arithmetic. *)
END_STRUCT
END_TYPE

```

- nFFT_Length はFFTの長さです。1より大きく、2の累乗である必要があります。
- nWindowLength は解析ウィンドウの長さ（サンプル数）です。長さは1より大きく、偶数である必要があります。
- fSampleRate 入力される時間信号のサンプリング・レート。この値は、Hz単位での結果のスケーリングに使用されます。
- nChannels は、独立したチャンネルの数を定義します。これはゼロより大きくなければなりません。
- nOrder は積分の最大次数です。これは0から2までの整数でなければなりません。チャンネルごとに決定される値の数は(nOrder+1)です。
- nMaxBands は、RMS値を計算する周波数帯域の最大数を指定します。
- nMaxClasses は、設定されているクラスの最大数です。これは1以上である必要があります。
- bMemorize はブール型の変数です。FALSE の場合、ファンクションブロックは、各ステップについて、最上位カテゴリの番号と対応するチャンネルを再計算します。値がTRUE の場合、制限値を超えると、ResetData()メソッドが実行されるか、チャンネルが上位カテゴリに達するまで、結果値が保存されます。デフォルト値はTRUEです。

- `eWindowType` 使用するウィンドウ関数 (`E_CM_WindowType` [▶ 274]型) を定義します。良いデフォルト値はウィンドウタイプ `eCM_HannWindow` です。
- `aWindowParameters` には、選択したウィンドウ関数のフリーパラメータが含まれます。
`eCM_KaiserWindow`を使用する場合、最初のエントリはパラメータ β を定義します。`eCM_FlatTopWindow`を使用する場合、全てのパラメータが使用されます。ウィンドウ関数 [▶ 19]を参照してください。
- `nOverlap` はオーバーラップするサンプル数を定義します。これはゼロ以上でなければなりません。値 `cCM_OverlapRecommended`が選択された場合、推奨されるオーバーラップが内部で計算されます (`F_CM_CalculateRecommendedOverlap` [▶ 270]を参照)。値 `cCM_OverlapInactive`は、内部バッファを無効化し、値をゼロに設定します。
- `bTransformToDecibel` は、FFTの結果をデシベル単位に変換するかどうかを示すブール値で、変換 $x \rightarrow 20 * \log_{10}(x)$ に従って処理されます。
- `fDecibelThreshold` は非常に小さな浮動小数点値で、ゼロより大きな値です。この数値より小さい値は、デシベルスケールへの変換前にこの値に置き換えられます。(その目的は、値域エラーの回避です。ゼロの対数は定義されておらず、小さな引数の極限では値が負の無限大に発散します。数字のゼロの引数、`arg(0)`も同様です。可能な最小値は $2.3e-308$ で、これは定数 `cCM_MinArgLog10` と同じ値です。)

● ウィンドウの長さに注意

1 `nWindowLength` の値は `nFFT_Length` の値以下になるようにしてください。FFTの長さは、必要な周波数分解能に合わせるすることができます。一般的には、FFT長の約4/5の値がウィンドウ長としてよく使用されます。

`nFFT_Length` が `nWindowLength` よりも大きい場合、FFTの周波数分解能が（したがって、戻り値ベクトルの長さも）増加します。長さの差は、フーリエ変換の前にゼロで満たされます。これは、より高い周波数分解能を得るためや、時間領域での逆変換による計算で円形エイリアシングを避けるために有用です。ただし、より高い周波数分解能にもかかわらず、結果にはそれ以上の情報は含まれていません。

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM_Base

5.3.14.34 ST_CM_WatchUpperThresholds_InitPars

ファンクションブロック固有の構造体で、ファンクションブロックの初期化時に解析される初期化パラメータを保持します。

```

TYPE ST_CM_WatchUpperThresholds_InitPars EXTENDS ST_CM_Object_InitPars :
STRUCT
    nChannels      : UDINT := 10;      (* Number of channels. *)
    nSubChannels   : UDINT := 0;      (* Number of subchannels. *)
    nMaxClasses    : UDINT := 3;      (* Number of configurable threshold classes. *)
    bMemorize      : BOOL := TRUE;    (* Flag if results are memorized. *)
END_STRUCT
END_TYPE

```

- `nChannels` は、独立したチャンネルの数を定義します。これはゼロより大きくなければなりません。
- `nSubChannels` は、独立したサブチャンネルの数を定義します。これはゼロ以上でなければなりません。
- `nMaxClasses` は、設定されているクラスの最大数です。これは1以上である必要があります。
- `bMemorize` はブール型の変数です。FALSE の場合、ファンクションブロックは、各ステップについて、最上位カテゴリの番号と対応するチャンネルを再計算します。値がTRUE の場合、制限値を超えると、`ResetData()`メソッドが実行されるか、チャンネルが上位カテゴリに達するまで、結果値が保存されます。デフォルト値はTRUEです。

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM_Base

5.3.14.35 ST_CM_ZoomFFT_InitPars

ファンクションブロック固有の構造体で、ファンクションブロックの初期化時に解析される初期化パラメータを保持します。

```
TYPE ST_CM_ZoomFFT_InitPars EXTENDS ST_CM_Object_InitPars :
STRUCT
    nFFT_Length      : UDINT := 2048;          (* Length of FFT. *)
    fSampleRate      : LREAL := 10000.0;       (* Sample rate in Hertz. *)
    fCenter          : LREAL := 200.0;         (* Center frequency of band in Hertz. *)
    nDecimationFactor : UDINT := 16;           (* FFT decimation factor. *)
    nOrder           : UDINT := 4;             (* Order of lowpass filter in range [1 .. 20]. *)
    nChannels        : UDINT := 1;             (* Number of channels. *)
END_STRUCT
END_TYPE
```

- nFFT_Length はFFTの長さです。1より大きく、2の累乗である必要があります。
- fSampleRate 入力される時間信号のサンプリング・レート。この値は、Hz単位での結果のスケールングに使用されます。
- fCenter は使用する周波数帯域の中心です。帯域幅はデシメーション係数に依存します。
- nDecimationFactor は、入力信号をセンタリングした後に適用されるローパスフィルタのデシメーション係数です。ダウンサンプリング時に整数分割を確実にするため、この値は2の累乗でなければなりません。
- nOrder は使用するIIRフィルタの次数を定義します。
- nChannels は、独立したチャンネルの数を定義します。これはゼロより大きくなければなりません。

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM_Base

5.3.14.36 ST_MA_MultiArray_InitPars

ファンクションブロック固有の構造体で、ファンクションブロックの初期化時に解析される初期化パラメータを保持します。

```
TYPE ST_MA_MultiArray_InitPars :
STRUCT
    eTypeCode : E_MA_ElementTypeCode := eMA_TypeCode_LREAL;          (* Used datatype. *)
)
    nDims      : UDINT := 1;          (* Number of dimensions the array will have. *)
    aDimSizes : ARRAY[0.. 15] OF UDINT := [1,1,1,1,1,1,1,1,1,1,1,1,1,1,1]; (* Size for each dimension. *)
END_STRUCT
END_TYPE
```

- eTypeCode このパラメータは、マルチアレイバッファの要素型 ([E_MA_ElementTypeCode](#) [[▶ 274](#)]) を指定します。
- nDims このパラメータは、マルチアレイ バッファの次元数を指定します。
- aDimSizes 各次元のサイズはこの配列で指定します。以下のアルゴリズムの入力バッファの形状が（入力ストリームにおいて）'m x n'で定義されている場合、マルチアレイバッファはaDimSizes := [m,n]で作成する必要があります。

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4013	PCまたはCX (x86、x64)	Tc3_MultiArray

5.4 グローバル定数

5.4.1 GVL_CM

解析ファンクションブロック定数：

```
cCMA_MaxDest : UDINT := 20;
cCMA_MaxID   : UDINT := 1000;
```

- cCMA_MaxDest: 解析ファンクションブロックのデスティネーションの最大数。
- cCMA_MaxID: 解析ファンクションブロックに発行可能な最大ID（解析ファンクションブロックの最大可能数に対応）。

転送トレイパラメータ：

内部の転送トレイは、解析ファンクションブロック間のデータ転送のために以下の定数で初期化されます。

```
cCMA_InitParsTransferTray : ST_MA_TransferTray_InitPars := (
    nStreams      := 2048,
    nMaxEntries   := 10,
    nQueueSize    := 64,
    bLockFree     := TRUE,
    nUpdatePeriod := 2 );
```

- nStreams このパラメータは、独立して機能するキューの数を指定します。各タスクにまたがるデータストリームには、個別のキューが必要です。追加のキューはシステムリソースを必要としません。
- nMaxEntries このパラメータは、キューが含むことができる要素の最大数を示します。データバッファの通信では、バッファオーバーランを防ぐため、通常は対象となるすべてのバッファがキュー内に収まるだけの空き容量を確保するのが合理的です。1の値を選択することもできます。
- nQueueSize キューの予約長。この値はnMaxEntries より大きく、さらに2の累乗でなければなりません。
- bLockFree このパラメータがTRUEの場合、キューには最新のロックフリー実装が使用されます。これがプリセットの状態です。それ以外の場合は、割り込みブロックを使った従来の実装が使用されます。ロックフリー実装は、システム全体の時間的挙動を改善することができますが、極端に高付加な状況ではレイテンシーが増加する可能性があります。
- nUpdatePeriod このパラメータは、内部の中間結果の更新頻度を指定します。複雑な操作の頻度は、1を超える値で低減できます。通常、2（プリセット）か3の値が実用的です。

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86, x64)	Tc3_CM, Tc3_CM_Base, Tc3_MultiArray

5.4.2 GVL_CM_Base

制限値定数：

```
cCM_MinArgLog10 : LREAL := 2.3E-308; (* approximate minimum argument of decadic logarithm *)
cCM_MinArgLogN  : LREAL := 2.3E-308; (* approximate minimum argument of natural logarithm *)
cCM_MinArgDiv   : LREAL := 2.3E-308; (* minimum argument of division *)
```

これらの定数の目的は、値域エラーを回避することです。負の数とゼロの場合には、実数の対数は定義されません。このため、定数はゼロに近い値で定義されています。ゼロによる除算も同様です。

オーバーラップを推奨する定数：

```
cCM_OverlapRecommended : UDINT := TO_UDINT(-1); (* setting for recommended overlap (e.g. HannWindow: 50% overlapping) *)
cCM_OverlapInactive     : UDINT := TO_UDINT(-2); (* setting in order to deactivate overlap *)
```

これらの定数は、ウィンドウ関数を使用する際、内部データバッファサイズを設定するためのものです。これらは、それぞれのファンクションブロックのInitPars構造体 [▶ 280]において、パラメータnOverlap [▶ 285]の値として機能します。

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM_Base

5.4.3 Global_Version

このグローバル定数には、ライブラリのバージョン情報が含まれています。

全てのライブラリは、固有のバージョンをもっています。このバージョンは、PLCライブラリレポジトリにも表示されています。

```
VAR_GLOBAL CONSTANT
    stLibVersion_Tc3_CM : ST_LibVersion;
END_VAR
```

このグローバル定数構造体の型定義：ST_LibVersion

既存のバージョンと必要なバージョンを比較するには、ファンクションF_CmpLibVersion(Tc2_Systemライブラリで定義)を使用します。

5.4.4 Param

実行前に設定可能なパラメータのリスト

```
VAR_GLOBAL CONSTANT
    eEventTraceLevel : TcEventSeverity := TcEventSeverity.Critical;
END_VAR
```

- eEventTraceLevel: このパラメータは、Condition Monitoringライブラリ内で TcEventLogger（を参照）を使用する際のイベントに関する一括設定を可能にします。

設定

パラメータは、ライブラリマネージャで変更します：GVLsフォルダのTc3_CM_BaseライブラリのParamの下にある "Value (editable)" 列で変更します。

The screenshot shows the Library Manager interface. The top pane lists libraries with columns for Name, Namespace, and Effective version. The Tc3_CM_Base library is highlighted. The bottom pane shows the library parameters for Tc3_CM_Base, 2.1.17.17. The parameter eEventTraceLevel is visible with its value set to TcEventSeverity.Critical.

Name	Namespace	Effective version
Tc2_Standard = Tc2_Standard, * (Beckhoff Automation GmbH)	Tc2_Standard	3.3.2.0
Tc2_System = Tc2_System, * (Beckhoff Automation GmbH)	Tc2_System	3.4.21.0
Tc2_Uilities = Tc2_Uilities, * (Beckhoff Automation GmbH)	Tc2_Uilities	3.3.35.0
Tc3_CM = Tc3_CM, * (Beckhoff Automation GmbH)	Tc3_CM	2.0.29.15
Tc3_CM_Base = Tc3_CM_Base, * (Beckhoff Automation GmbH)	Tc3_CM_Base	2.1.17.17
Tc3_Module = Tc3_Module, * (Beckhoff Automation GmbH)	Tc3_Module	3.3.21.0
Tc3_MultiArray = Tc3_MultiArray, * (Beckhoff Automation GmbH)	Tc3_MultiArray	2.0.13.8

Name	Type	Value (editable)	Comment
eEventTraceLevel	TcEventSeverity	TcEventSeverity.Critical	

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM_Base

6 サンプル

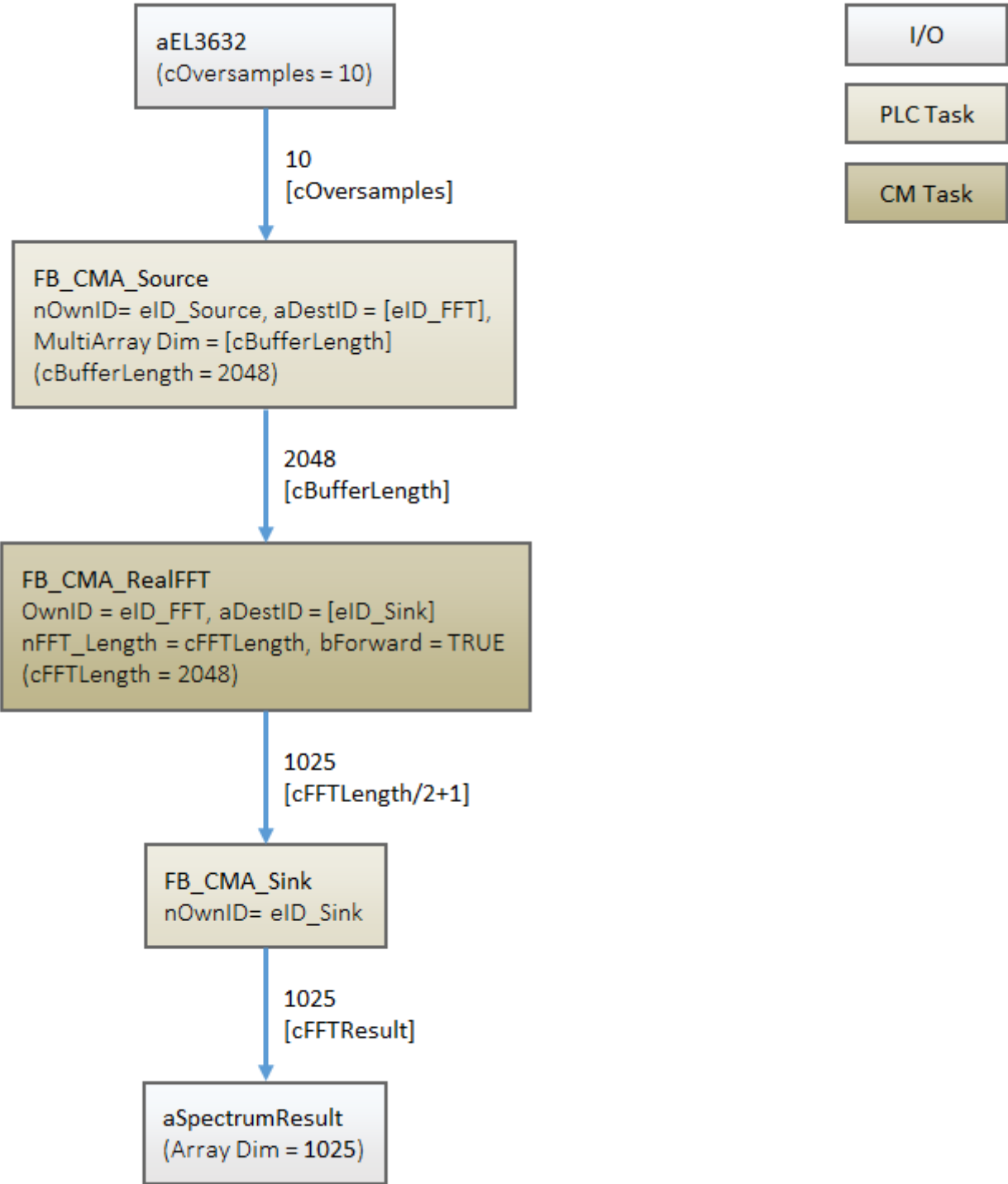
6.1 実数値入力信号によるFFT

このサンプルは、ファンクションブロック `FB_CMA_RealFFT` [▶ 203] を使ったスペクトル計算の実装例を示しています。

サンプルはこちらからダウンロードできます：

https://infosys.beckhoff.com/content/1033/TF3600_TC3_ConditionMonitoring/Resources/3394507531.zip

ブロック図



プログラムのパラメータ

次のファンクションブロックの設定に関する重要パラメータの一覧を以下の表に示します。

RealFFTファンクションブロック

FFTの長さ	2048
フォワード計算	TRUE

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4018	PCまたはCX (x86、x64)	Tc3_CM, Tc3_CM_Base, Tc3_MultiArray

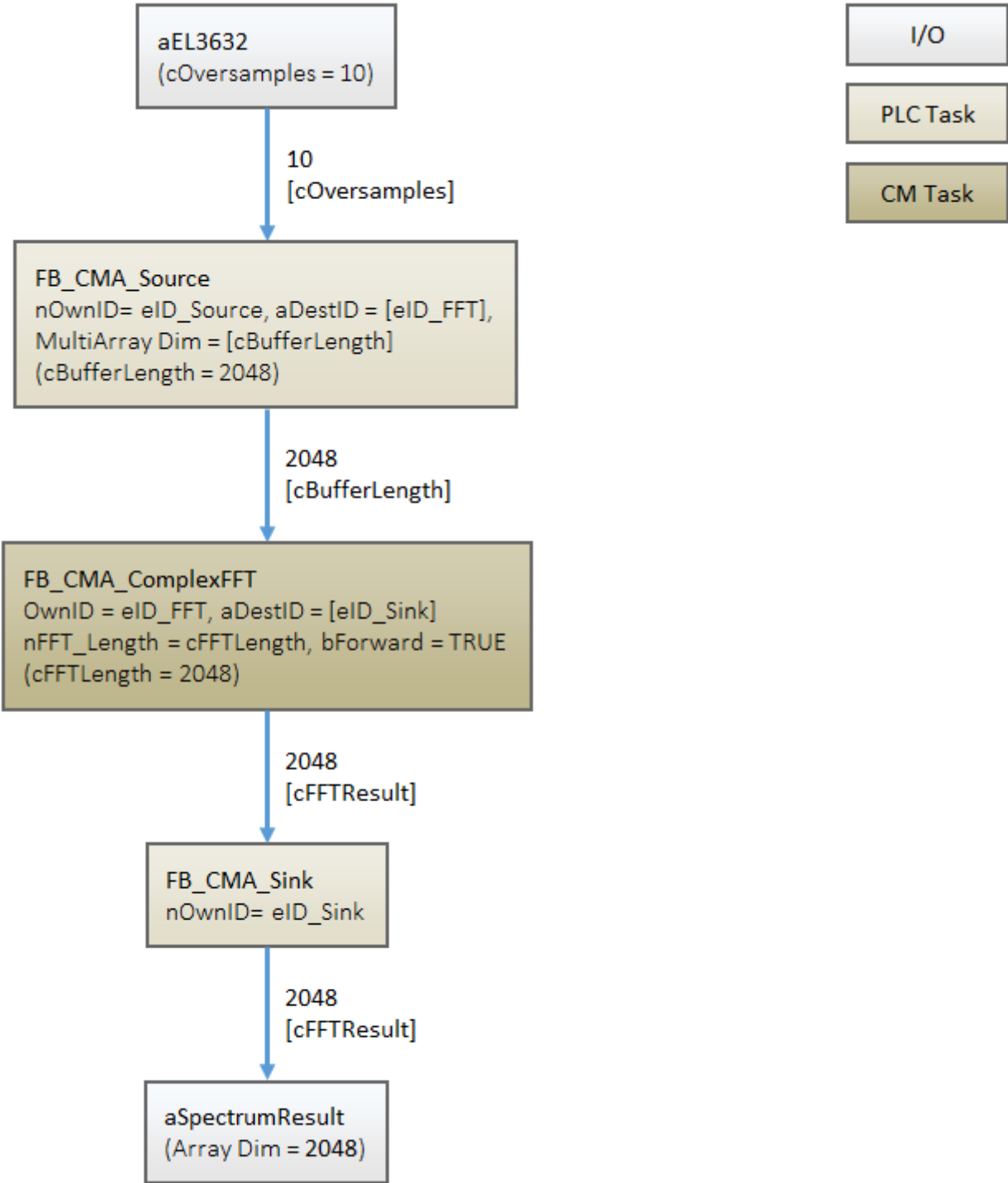
6.2 複素数値入力信号によるFFT

このサンプルは、ファンクションブロック `FB_CMA_ComplexFFT` [▶ 104] を使ったスペクトル計算の実装例を示しています。ファンクションブロック `FB_CMA_RealFFT` [▶ 203] とは対照的に、必要なマルチアレイにデータ型 `LCOMPLEX` が使用されます。

サンプルはこちらからダウンロードできます：

https://infosys.beckhoff.com/content/1033/TF3600_TC3_ConditionMonitoring/Resources/3394479755.zip

ブロック図



プログラムのパラメータ

次のファンクションブロックの設定に関する重要パラメータの一覧を以下の表に示します。

ComplexFFTファンクションブロック

タイプコード	eMA_TypeCode_LCOMPLEX
FFTの長さ	2048

フォワード計算

TRUE

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4018	PCまたはCX (x86、x64)	Tc3_CM, Tc3_CM_Base, Tc3_MultiArray

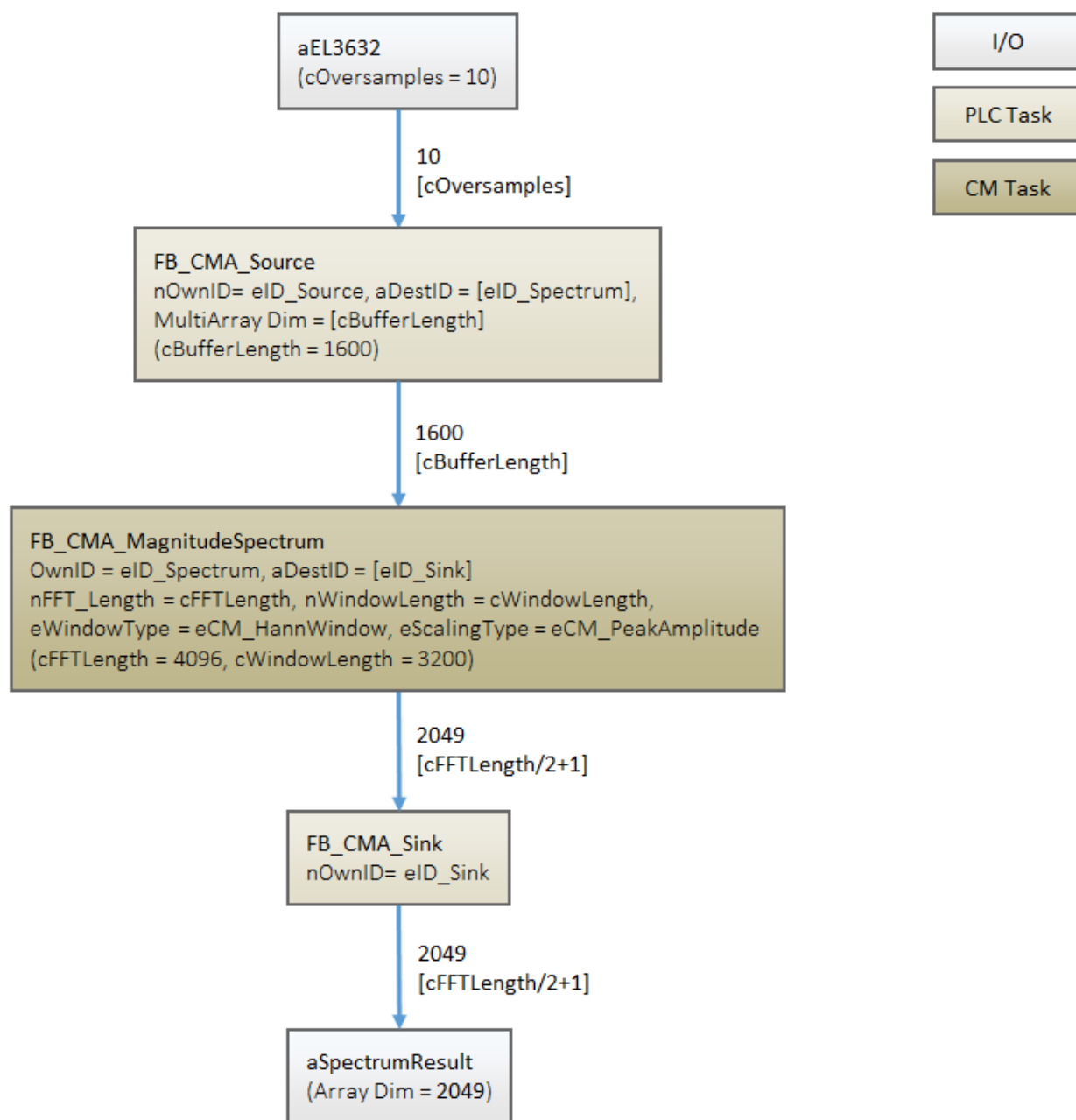
6.3 マグニチュードスペクトル：

このサンプルは、シングルチャンネルのマグニチュードスペクトルを実装しています。すなわち、コードは、EL3632などのハードウェアモジュールの離散入力信号を収集する制御タスクと、スペクトルを計算するCMタスクの2つのタスクに分割されます。以下のブロック図は、サンプルに実装されている解析チェーンを示しています。

サンプルのソースコードはこちらからダウンロードできます：

https://infosys.beckhoff.com/content/1033/TF3600_TC3_ConditionMonitoring/Resources/3394574859.zip

ブロック図



プログラムのパラメータ

以下の表は、マグニチュードスペクトルのファンクションブロック設定に重要なパラメータの一覧です。

FFTの長さ	4096
ウィンドウサイズ	3200
バッファサイズ	1600
ウィンドウタイプ	eCM_HannWindow
スケーリングタイプ	eCM_ROOT_POWER_SUM
デシベル (dB) 単位のスケーリング	FALSE

グローバル定数

これらのパラメータは、グローバル変数のリストに定数として定義されています。

```
VAR_GLOBAL CONSTANT
    cOversamples      : UDINT := 10;      // oversampling factor
    cBufferLength     : UDINT := 1600;    // size of buffer for spectrum
    cWindowLength     : UDINT := 3200;    // size of window
    cFFTResult        : UDINT := 2049;    // size of spectrum result
    cFFTLengh        : UDINT := 4096;    // spectrum lines
END_VAR
```

制御タスクのコード

次のコードスニペットは、MAINプログラムでの宣言を示しています：

```
PROGRAM MAIN

VAR CONSTANT
    cInitSource      : ST_MA_MultiArray_InitPars := ( eTypeCode := eMA_TypeCode_LREAL, nDims := 1, aDim
    Sizes := [cBufferLength]);
END_VAR

VAR
    nInputSelection  : UDINT := 1; // Switch between hardware and function generator
    nSample          : UDINT;
    aEl3632 AT %I*    : ARRAY[1..cOversamples] OF INT; // Input from hardware e.g. EL3632
    aBuffer          : ARRAY[1..cOversamples] OF LREAL;

    fbSource         : FB_CMA_Source :=( stInitPars := cInitSource, nOwnID := eID_Source, aDestIDs :=
[eID_Spectrum]); // Initialize source buffers
    fbSink           : FB_CMA_Sink := (nOwnID := eID_Sink);
    aSpectrumResult  : ARRAY[1..cFFTResult] OF LREAL; // Copy result
END_VAR
```

メソッドはMAINプログラムを呼び出します：

```
fbSource.Input1D(pDataIn := ADR(aBuffer),
    nDataInSize := SIZEOF(aBuffer),
    eElementType := eMA_TypeCode_LREAL,
    nWorkDim := 0,
    pStartIndex := 0,
    nOptionPars := 0);

fbSink.Output1D(pDataOut := ADR(aSpectrumResult),
    nDataOutSize := SIZEOF(aSpectrumResult),
    eElementType := eMA_TypeCode_LREAL,
    nWorkDim := 0,
    nElements := 0,
    pStartIndex := 0,
    nOptionPars := 0,
    bNewResult => bCalculate);
```

CMタスクのコード

MAIN_CMプログラムでの宣言：

```
PROGRAM MAIN_CM

VAR CONSTANT
    cInitSpectrum : ST_CM_MagnitudeSpectrum_InitPars := (nFFT_Length := cFFTLengh,
    nWindowLength := cWindowLength,
    bTransformToDecibel:= FALSE,
    eWindowType := eCM_HannWindow,
    eScalingType := eCM_RMS);
END_VAR
VAR
    fbSpectrum : FB_CMA_MagnitudeSpectrum :=(stInitPars := cInitSpectrum,
    nOwnID := eID_Spectrum,
    aDestIDs := [eID_Sink]);
END_VAR
```

MAIN_CMプログラムのメソッド呼び出し：

```
fbSpectrum.Call();
```

サンプルコードの結果は、入力信号として任意の振幅と周波数を持つ正弦波信号を使用して確認できます。上記の変数 fRmsValue は、amplitude/SQRT(2) と完全に一致する必要があります。

各周波数値は、スペクトル結果の対応する配列インデックスに割り当てることができます。計算式：
 サンプルレート = オーバーサンプリング係数 / サンプリングタスクのサイクルタイム
 インデックス = 周波数 * (FFT 長 / サンプルレート)

要件

開発環境	ターゲットシステム	必要なPLCライブラリ
TwinCAT v3.1.4013	PCまたはCX (x86、x64)	Tc3_CM, Tc3_CM_Base, Tc3_MultiArray

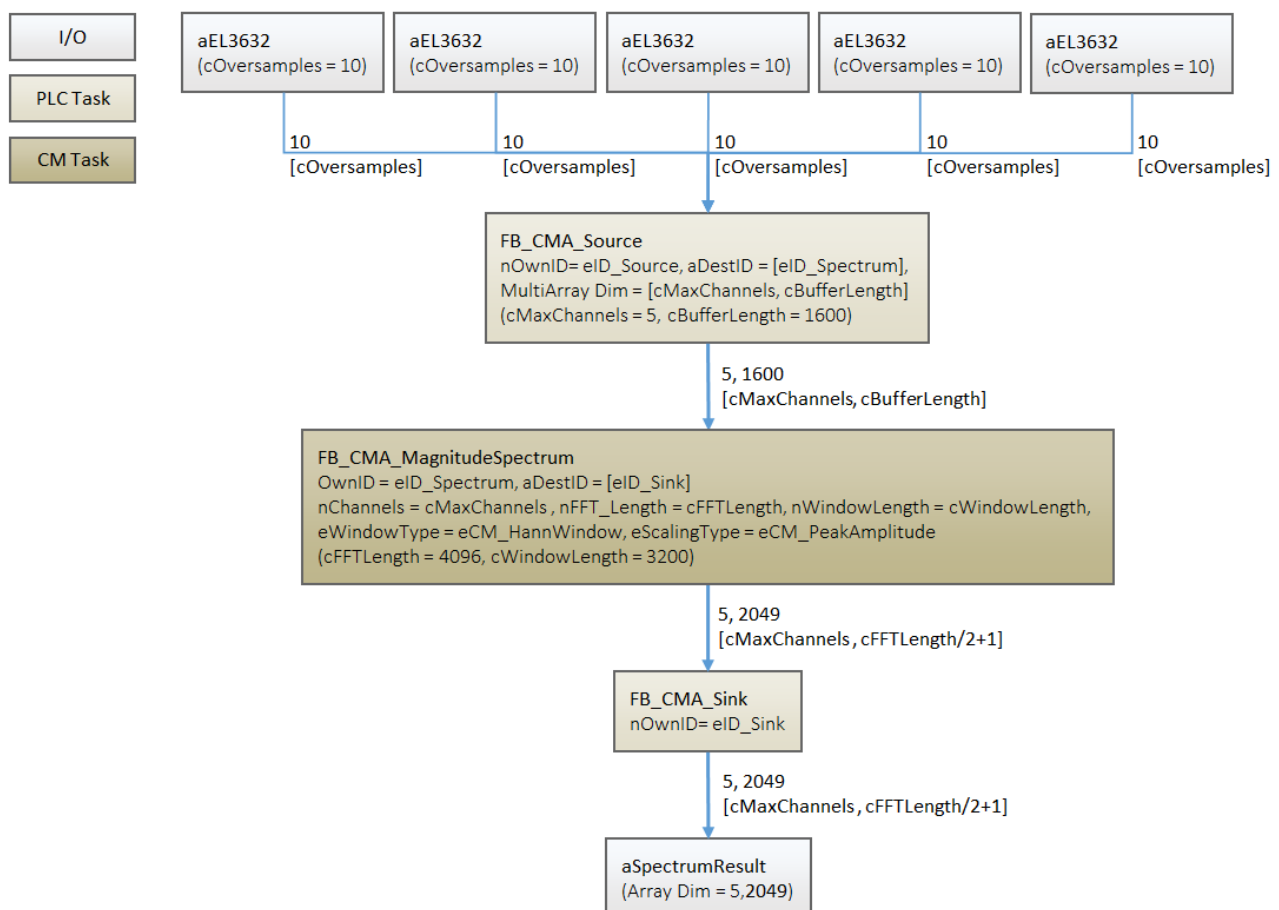
6.4 マルチチャンネルのマグニチュードスペクトル

このサンプルは、5つの入力チャンネルのマグニチュードスペクトルを同時に実装します。すなわち、コードは、EL3632などのハードウェアモジュールの離散入力信号を収集する制御タスクと、スペクトルを計算するCMタスクの2つのタスクに分割されます。以下のブロック図は、プログラムに実装された解析チェーンを示しています。

サンプルのソースコードはこちらからダウンロードできます：

https://infosys.beckhoff.com/content/1033/TF3600_TC3_ConditionMonitoring/Resources/3394576523.zip

ブロック図



プログラムのパラメータ

以下の表は、マグニチュードスペクトルのファンクションブロック設定に重要なパラメータの一覧です。

チャンネル

5

FFTの長さ	4096
ウィンドウサイズ	3200
バッファサイズ	1600
ウィンドウタイプ	eCM_HannWindow
スケーリングタイプ	eCM_RootPowerSum
デシベル換算	FALSE

グローバル定数

これらのパラメータは、グローバル変数のリストに定数として定義されています。

```
VAR_GLOBAL CONSTANT
    cOversamples : UDINT := 20; // number of oversamples
    cMaxChannels : UDINT := 5; // Number of data channels
    cWindowType : E_CM_WindowType := E_CM_WindowType.eCM_HannWindow; // window type for analysis
    cWindowLength : UDINT := 3200; // length of signal window.
    cOverlap : UDINT := 1600; // recommended buffer overlap
    cBufferLength : UDINT := cWindowLength -
    cOverlap; // internal buffer size with 50% overlapping
    cFFTLenght : UDINT := 4096; // length of FFT for mag.
    spectrum
    cFFTResult : UDINT := 2049; // result of mag. spectrum
    (cFFTLenght/2+1)
END_VAR
```

グローバル変数

このパラメータはグローバル変数のリストに定義されています。

```
VAR_GLOBAL
    bInvalidateCh4 : BOOL := FALSE; // Invalidate input signal on channel 4
END_VAR
```

解説

サンプルコードの結果は、任意の振幅と周波数の正弦波信号に対してテストすることができます。RMS値は、各チャンネル番号にしたがって配列aRmsValue に保存されます。その結果は、全ての正弦波信号のピーク振幅をSQRT(2) で割ったものに完全に一致する必要があります。サンプルコードは、要件とターゲットシステムのリソースに応じて、5チャンネル以上に拡張できます。

グローバル変数bInvalidateCh4 := TRUEの設定は、不正な入力データに対するエラー処理の一例を示しています。

要件

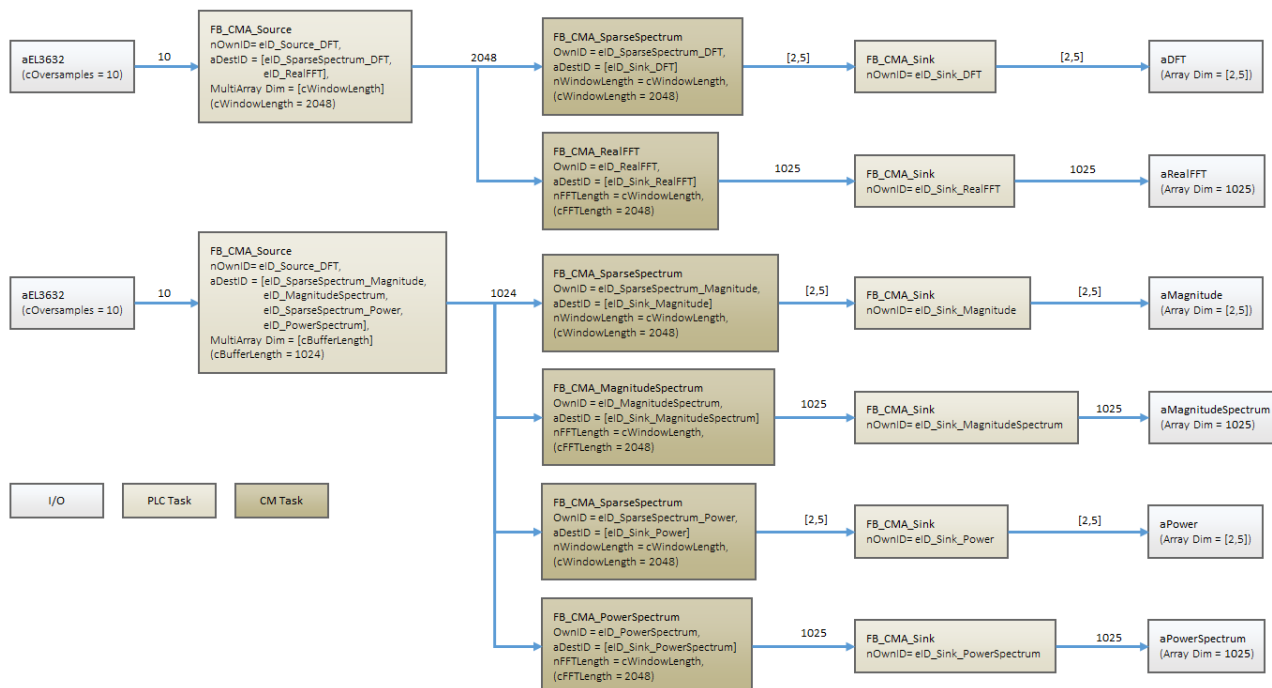
開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM, Tc3_CM_Base, Tc3_MultiArray

6.5 個々のスペクトル値の計算

このサンプルは、ファンクションブロックFB_CMA_SparseSpectrum の使用例です。DFT、マグニチュード、パワー値の計算のために、様々なウィンドウ関数とスケーリング方法が示されています。検出における数値的な影響は、生成された信号の2つの周波数成分で示されます。1つ目の周波数はスペクトル範囲内の数値分解能の倍数に対応し、2つ目の周波数はその倍数の間に位置します。

サンプルのソースコードはこちらからダウンロードできます：https://infosys.beckhoff.com/content/1033/TF3600_TC3_ConditionMonitoring/Resources/9066662027.zip

ブロック図



プログラムのパラメータ

以下の表は、ファンクションブロックを設定するための重要なパラメータの一覧です。

FFTの長さ	2048 / 2048 / 2048
ウィンドウサイズ	2048 / 2048 / 2048
バッファサイズ	2048 / 1024 / 1024
ウィンドウタイプ	eCM_RectangularWindow / eCM_HannWindow / eCM_HannWindow
スケーリング方法	eCM_NoScaling / eCM_PeakAmplitude / eCM_PeakAmplitude
スペクトル値の種類	eCM_DFT / eCM_Magnitude / eCM_Power

周波数帯域の設定

GVL_Constantsは、アルゴリズムを初期化するための中心的なパラメータと、生成される信号の特性を定義します。

```

VAR_GLOBAL CONSTANT
    cSampleRate : LREAL := 10000; // Sample rate of input signal.
    cWindowLength : UDINT := 2048; // Internal buffer size with 50% overlapping
    cResolution : LREAL := cSampleRate / cWindowLength; // Frequency resolution
    cBands : UDINT := 2; // Number of bands
    cSetFrequency : ARRAY[1..cBands] OF LREAL := [ 41*cResolution, 413 ]; // Frequency in Hz; [ exact, intermediate ]
    cSetAmplitude : ARRAY[1..cBands] OF LREAL := [ 1.0, 2.0 ]; // Peak amplitudes of sine signals
    cBandWidth : UDINT := 5; // Computed bins per frequency.
    cDFTBins : UDINT := cBandWidth * cBands; // Number of spectral bins, cBandWidth bins per frequency
END_VAR

```

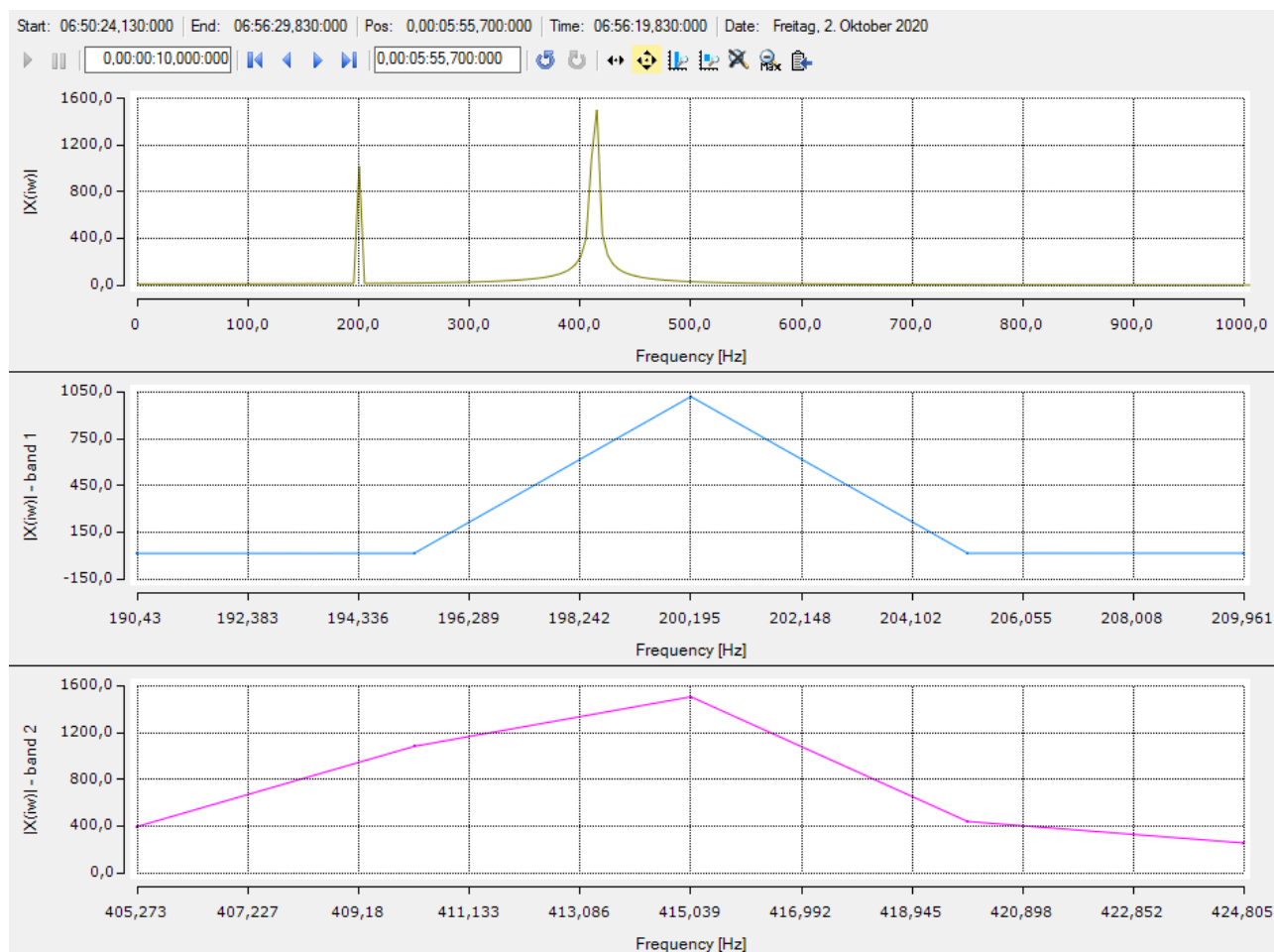
生成された信号(MAIN.aBuffer)は2つの周波数成分で構成されます。1つ目は数値分解能に基づいて選択され、つまり $f = 10000 \text{ Hz} / 2048 = 4.8828125 \text{ Hz}$ の倍数になります。もう1つは、ピークが2つのスペクトル値の間に位置し、正確に表現できないように選択されます。数値的な効果を示すため、求めたスペクトル値の周辺でさらに4つの値を計算します。設定はMAIN_CMで行われます。


```
// Compute parameters, adjust if cDFTBins is changed.
FOR i := 1 TO cBands DO
  k := LREAL_TO_DINT(cSetFrequency[i] / cResolution);
  aDFTBins[(i-1) * cBandWidth + 1] := DINT_TO_UDINT(MAX(k-2,1));
  aDFTBins[(i-1) * cBandWidth + 2] := DINT_TO_UDINT(MAX(k-1,1));
  aDFTBins[(i-1) * cBandWidth + 3] := DINT_TO_UDINT(MIN(k+0,nyquist));
  aDFTBins[(i-1) * cBandWidth + 4] := DINT_TO_UDINT(MIN(k+1,nyquist));
  aDFTBins[(i-1) * cBandWidth + 5] := DINT_TO_UDINT(MIN(k+2,nyquist));
END_FOR
```

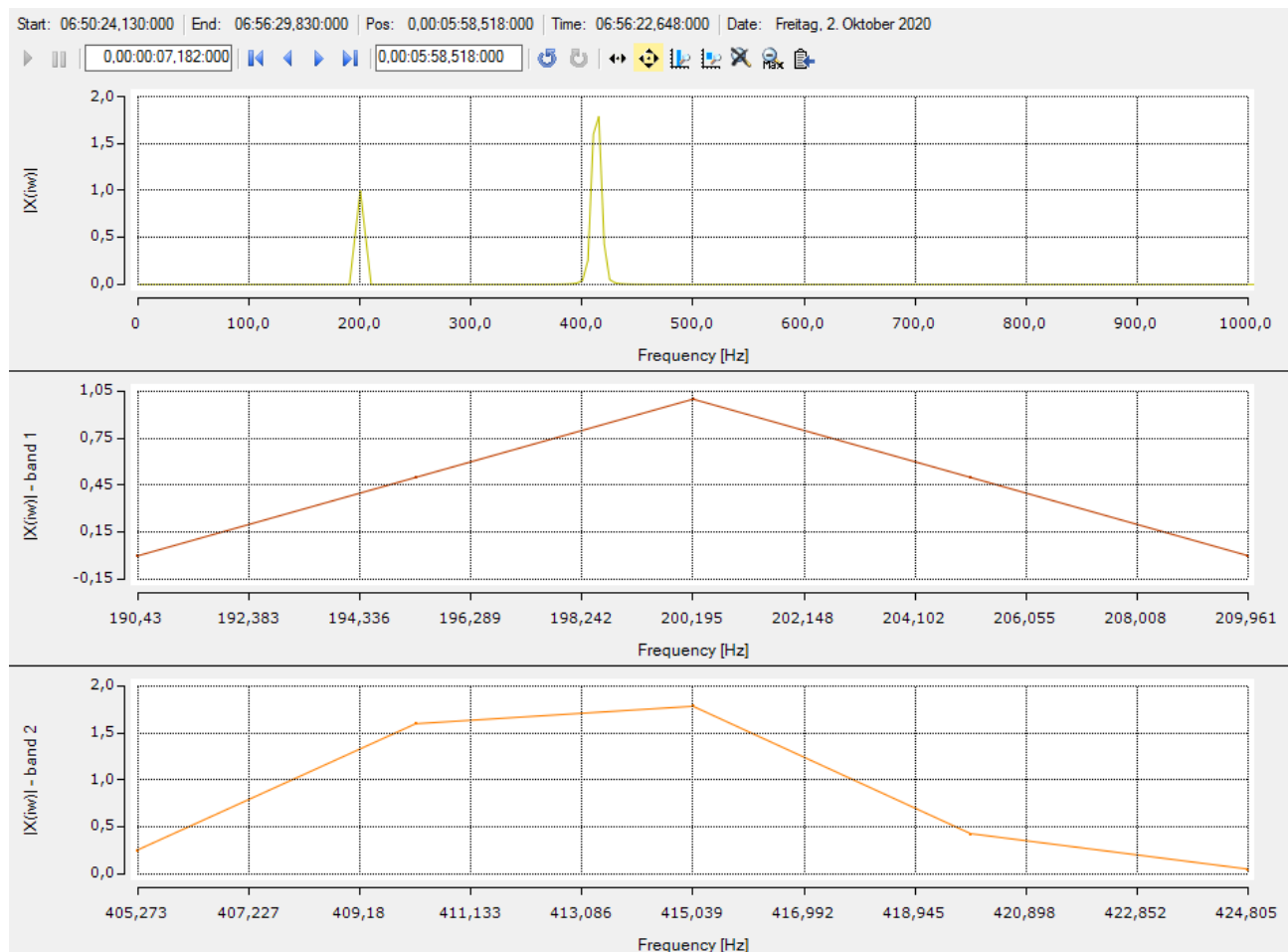
結果の可視化

このサンプルには、ファンクションブロックFB_CMA_SparseSpectrumから計算されたスペクトル値を参照アルゴリズム（FB_CMA_RealFFT、FB_CMA_MagnitudeSpectrum、FB_CMA_PowerSpectrum）と比較する、広範なTwinCAT計測プロジェクトが含まれています。数値周波数分解能の倍数となる励振と、その中間に位置する値との違いが示されています。可視化された周波数帯域は、対応する領域の「ズーム（拡大）」とみなすことができます。

次の図は、ファンクションブロック FB_CMA_RealFFT と FB_CMA_MagnitudeSpectrum の比較結果を示しています。



FB_RealFFTのスペクトル（上）とバンド1（中央）とバンド2（下）のスペクトル値。



ファンクションブロックFB_MagnitudeSpectrumのマグニチュードスペクトル（上）と、バンド1（中央）とバンド2（下）のスペクトル値。

要件

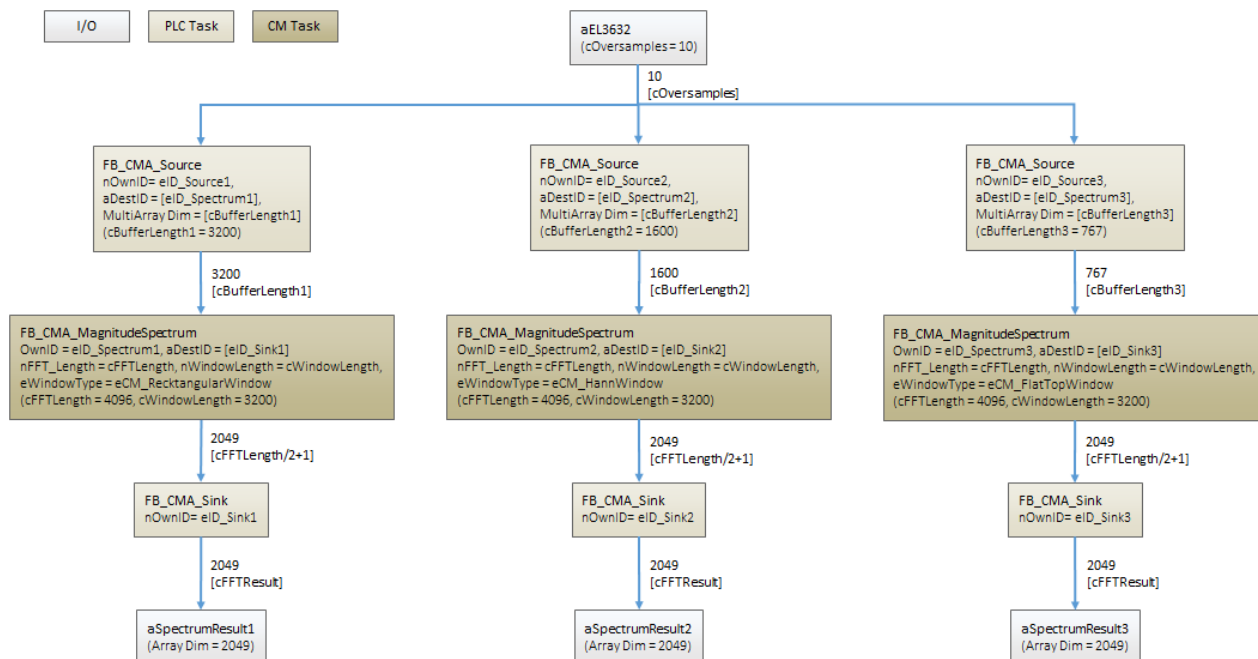
開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM, Tc3_CM_Base, Tc3_MultiArray

6.6 ウィンドウ関数

このサンプルは、複数のシングルチャンネルのマグニチュードスペクトルを実装し、異なるウィンドウ関数の使用を比較します。3つのバリエーションが示されています：オーバーラップ機構を無効にしたもの、推奨オーバーラップの自動計算、適切なファームウェア機能を使った手動設定の3つです。

サンプルのソースコードはこちらからダウンロードできます：https://infosys.beckhoff.com/content/1033/TF3600_TC3_ConditionMonitoring/Resources/5261536139.zip

ブロック図



プログラムのパラメータ

以下の表は、マグニチュードスペクトルのファンクションブロック設定に重要なパラメータの一覧です。

FFTの長さ	4096
ウィンドウサイズ	3200
バッファサイズ	3200 / 1600 / 767
ウィンドウタイプ	eCM_RectangularWindow / eCM_HannWindow / eCM_FlatTopWindow
デシベル変換	False

オーバーラップの計算／仕様

バージョン1：ファンクションの使用 F_CM_CalculateRecommendedOverlap [▶ 270]

指定されたファンクションによって、推奨されるオーバーラップを実行時に計算できます。ダウンロード／ログイン後、必要な値がMAIN_CMの最初のサイクルで呼び出されます。

```
IF bCalculateOverlap THEN
  // recommended overlap for window 1
  nOverlap1 := F_CM_CalculateRecommendedOverlap(eWindowType := cWindowType1, aWindowParameters :=
cWindowParameters1, nWindowLength := cWindowLength);

  // recommended overlap for window 2
  nOverlap2 := F_CM_CalculateRecommendedOverlap(eWindowType := cWindowType2, aWindowParameters :=
cWindowParameters2, nWindowLength := cWindowLength);

  // recommended overlap for window 3
  nOverlap3 := F_CM_CalculateRecommendedOverlap(eWindowType := cWindowType3, aWindowParameters :=
cWindowParameters3, nWindowLength := cWindowLength);

  bCalculateOverlap := FALSE;
END_IF
```

このようにして得られた長さは、その後、解析チェーンの設定に入力します（GVL_Constants のパラメータ cOverlap1, cOverlap2, cOverlap3）。ここでは、バッファの長さの依存関係に注意してください。再ダウンロード/再ログイン後、アプリケーションでオーバーラップに関しての設定を行います。

バージョン2：初期パラメータの読み込み オンライン表示 [▶ 80]

オーバーラップの初期化構造で値 `nOverlap := cCM_OverlapRecommended` が使用される場合、最適なオーバーラップは、ウィンドウパラメータに応じて、ファンクションブロックの初期化フェーズで計算されます。この値は、ダウンロードとログイン後、ノード `stInitPars` のファンクションブロックのオンライン表示で読み取ることができます。このバージョンでは、アプリケーションを起動する必要はありません。

Expression	Type	Value
 <code>stInitPars</code>	ST_CM_MagnitudeS...	
 <code>eObjState</code>	TCOM_STATE	TCOM_STATE_OP
 <code>sObjName</code>	STRING(127)	'CMA_MagnitudeSpectrum'
 <code>eTraceLevel</code>	TCEVENTSEVERITY	Critical
 <code>nFFT_Length</code>	UDINT	4096
 <code>nWindowLength</code>	UDINT	3200
 <code>eWindowType</code>	E_CM_WINDOWTYPE	eCM_HannWindow
  <code>aWindowParameters</code>	T_CM_WindowPara...	
 <code>nOverlap</code>	UDINT	1600
 <code>eScalingType</code>	E_CM_SCALINGTYPE	eCM_DiracScaling
 <code>bTransformToDecibel</code>	BOOL	FALSE
 <code>fDecibelThreshold</code>	LREAL	2.3E-308
 <code>nChannels</code>	UDINT	1

スペクトル2のオンライン表示（Hannウィンドウ）

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM, Tc3_CM_Base, Tc3_MultiArray

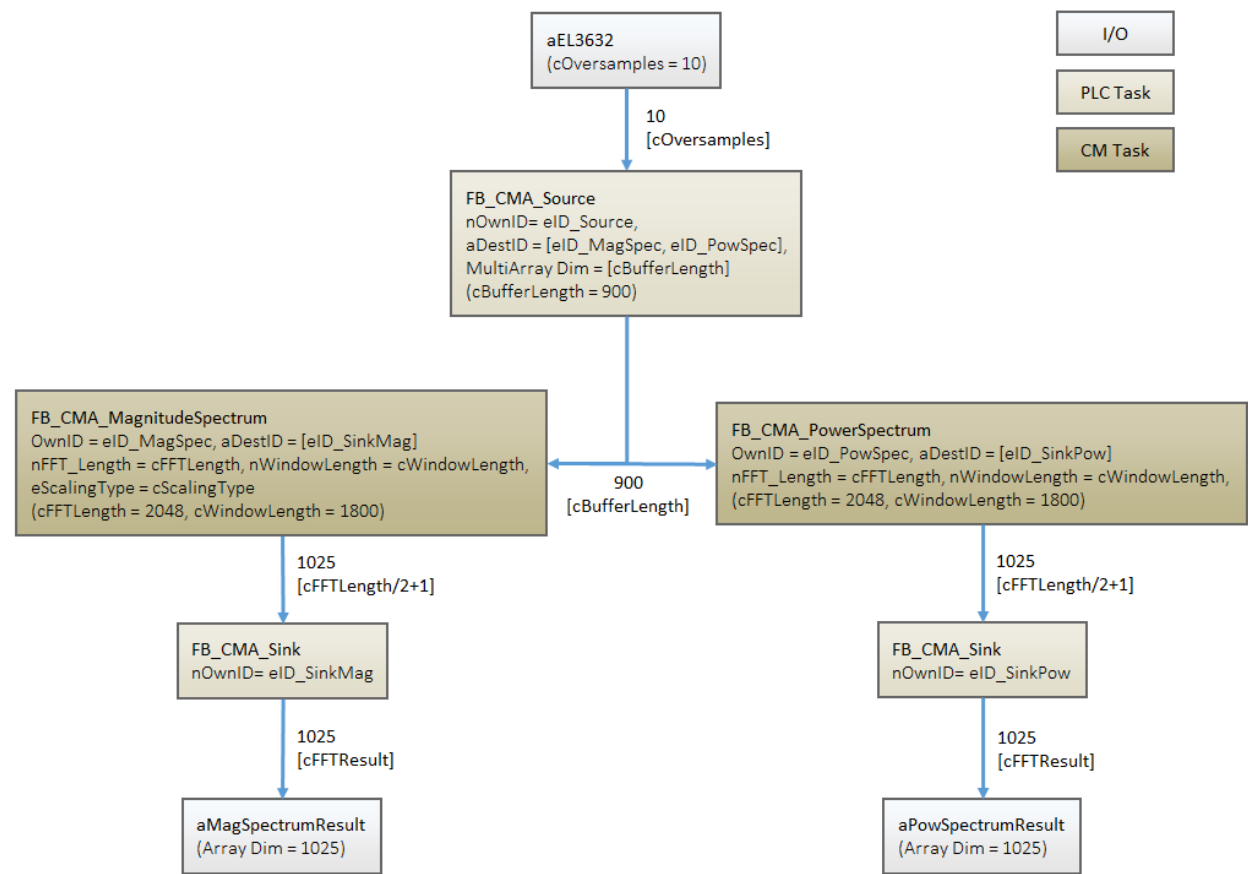
6.7 スペクトルのスケーリング

スペクトルのスケーリング [▶ 25] で説明されているように、Condition Monitoringライブラリには、スペクトルのスケーリングに関する様々なオプションが用意されています。このチュートリアルでは、簡単な正弦波を使って様々なスケーリングを検討し、理論的な理解を深めることができます。違いをより明確に示すため、スコープは0Hzから400Hzの範囲に限定されています。

チュートリアルはこちらからダウンロードできます：

https://infosys.beckhoff.com/content/1033/TF3600_TC3_ConditionMonitoring/Resources/3394510859.zip

ブロック図



プログラムのパラメータ

以下の表は、ファンクションブロック**FB_CMA_MagnitudeSpectrum** [▶ 161]および**FB_CMA_PowerSpectrum** [▶ 189]を設定するための重要なパラメータの一覧です。

GVL_Constantには3つのシナリオがあります。選択したコードセグメントをコメントアウトまたはコメントアウト解除し、設定を有効にすることでテストできます。シナリオの予想される動作は、コメントとしてGVLに文書化されています。

FFTの長さ	2048
ウィンドウサイズ	1800
デシベル換算	TRUE / FALSE
ウィンドウタイプ	eCM_HannWindow
スケーリングタイプ	eCM_PeakAmplitude / eCM_RMS

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4018	PCまたはCX (x86、x64)	Tc3_CM, Tc3_CM_Base, Tc3_MultiArray

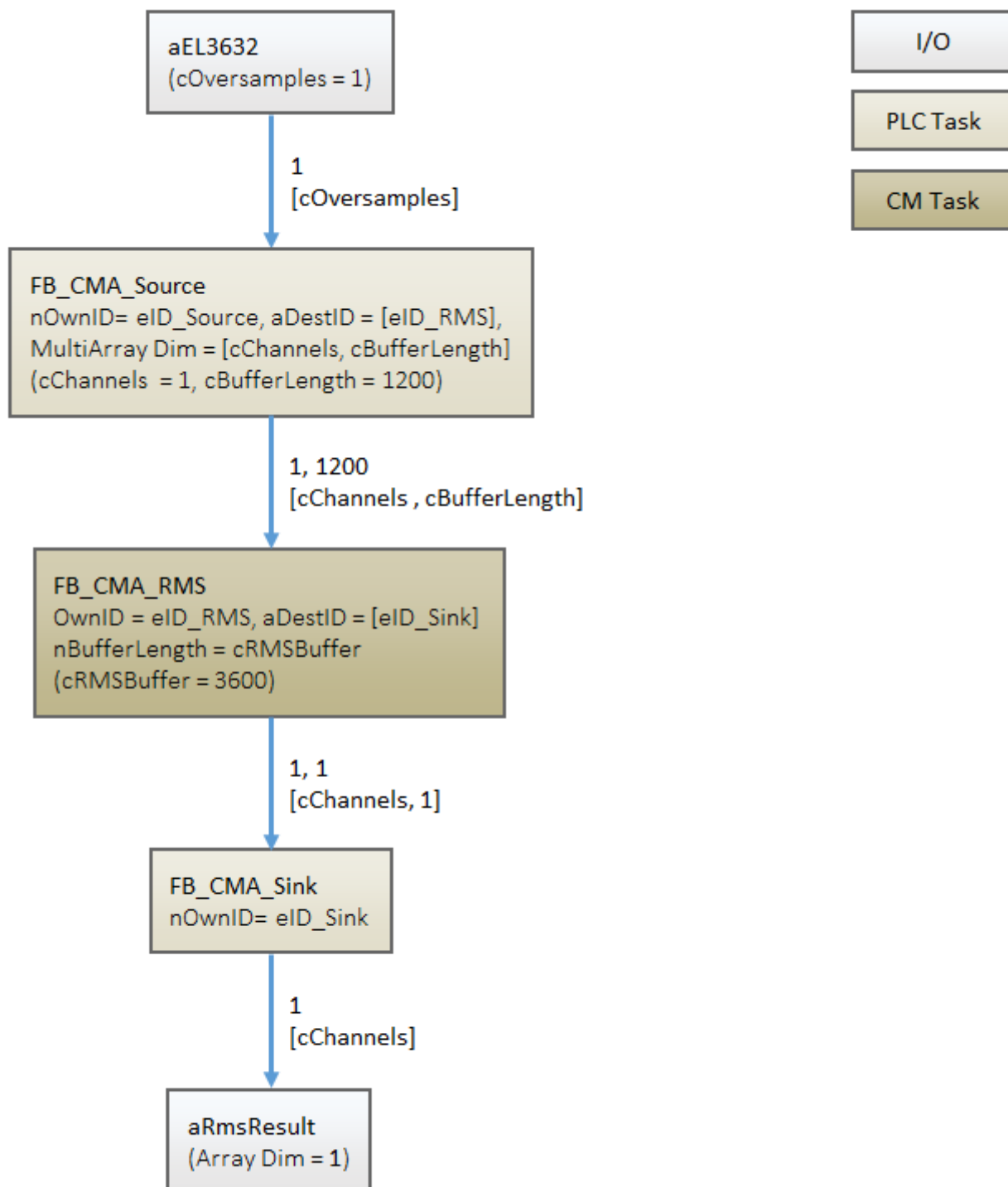
6.8 時間ベースのRMS値

このサンプルは、ファンクションブロック **FB_CMA_RMS** [▶ 207]を使った信号に対し時間ベースのRMS値計算を実装する方法を示しています。

サンプルはこちらからダウンロードできます：

https://infosys.beckhoff.com/content/1033/TF3600_TC3_ConditionMonitoring/Resources/3394509195.zip

ブロック図



プログラムのパラメータ

以下の表は、時間ベースのRMS値を計算するプログラムブロック設定のために重要なパラメータの一覧です。

チャンネル	1
バッファサイズ	1200
デシベル換算	FALSE

要件

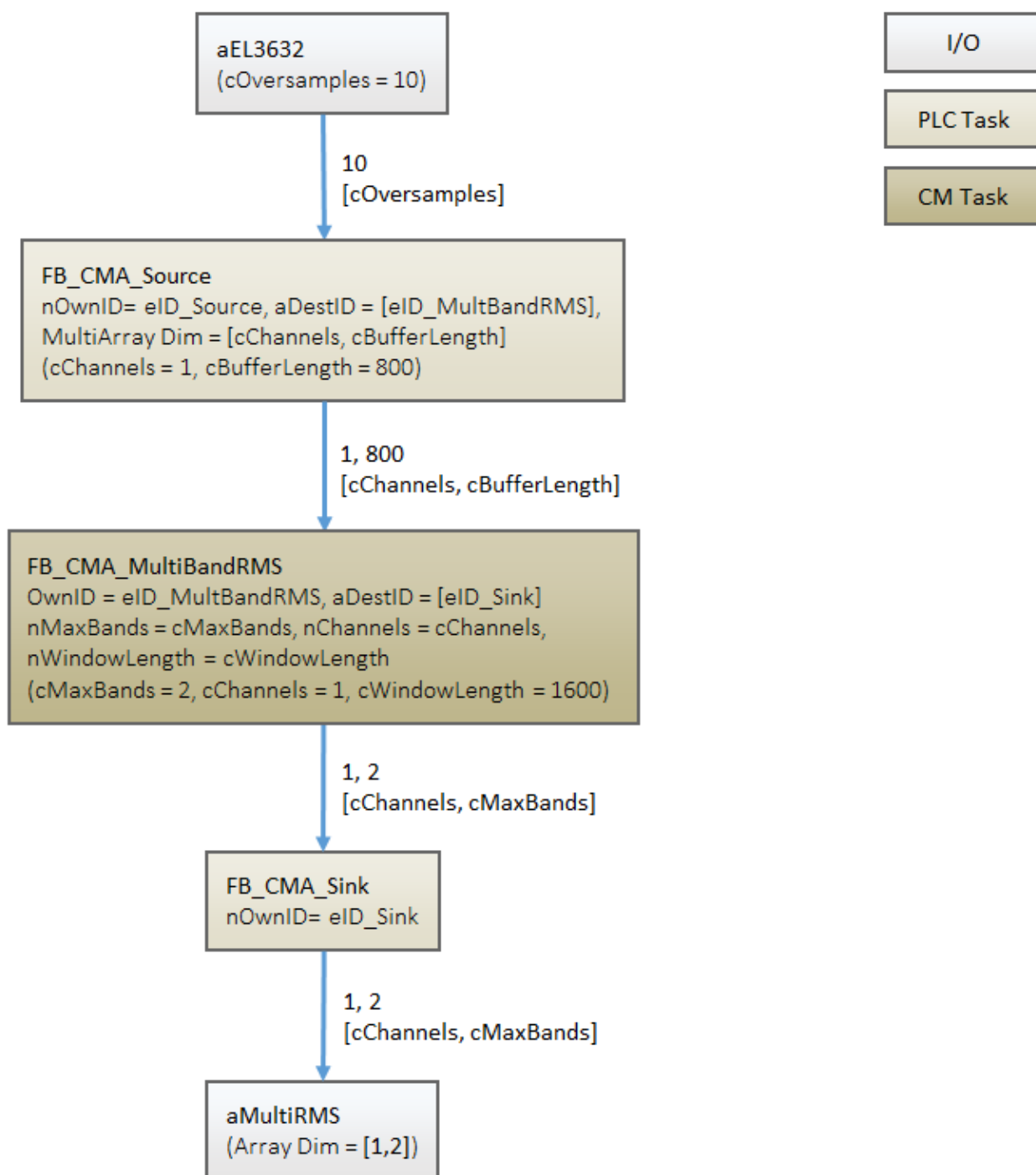
開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4016.12	PCまたはCX (x86、x64)	Tc3_CM, Tc3_CM_Base, Tc3_MultiArray

6.9 マルチバンドRMS

このサンプルは、ファンクションブロックFB_CMA_MultiBandRMS [▶ 177]を使用して、複数の周波数帯域で制限された信号のRMS値を計算する実装例を示しています。

サンプルはこちらからダウンロードできます：
https://infosys.beckhoff.com/content/1033/TF3600_TC3_ConditionMonitoring/Resources/3394505867.zip

ブロック図



プログラムのパラメータ

以下の表は、複数の周波数帯域で制限された信号のRMS値を計算するためのファンクションブロック設定に重要なパラメータの一覧です。

FFTのサイズ	2048
ウィンドウサイズ	1600
サンプリングレート	10000
周波数帯域	2

チャンネル	1
ウィンドウタイプ	eCM_HannWindow
デシベル換算	FALSE

要件

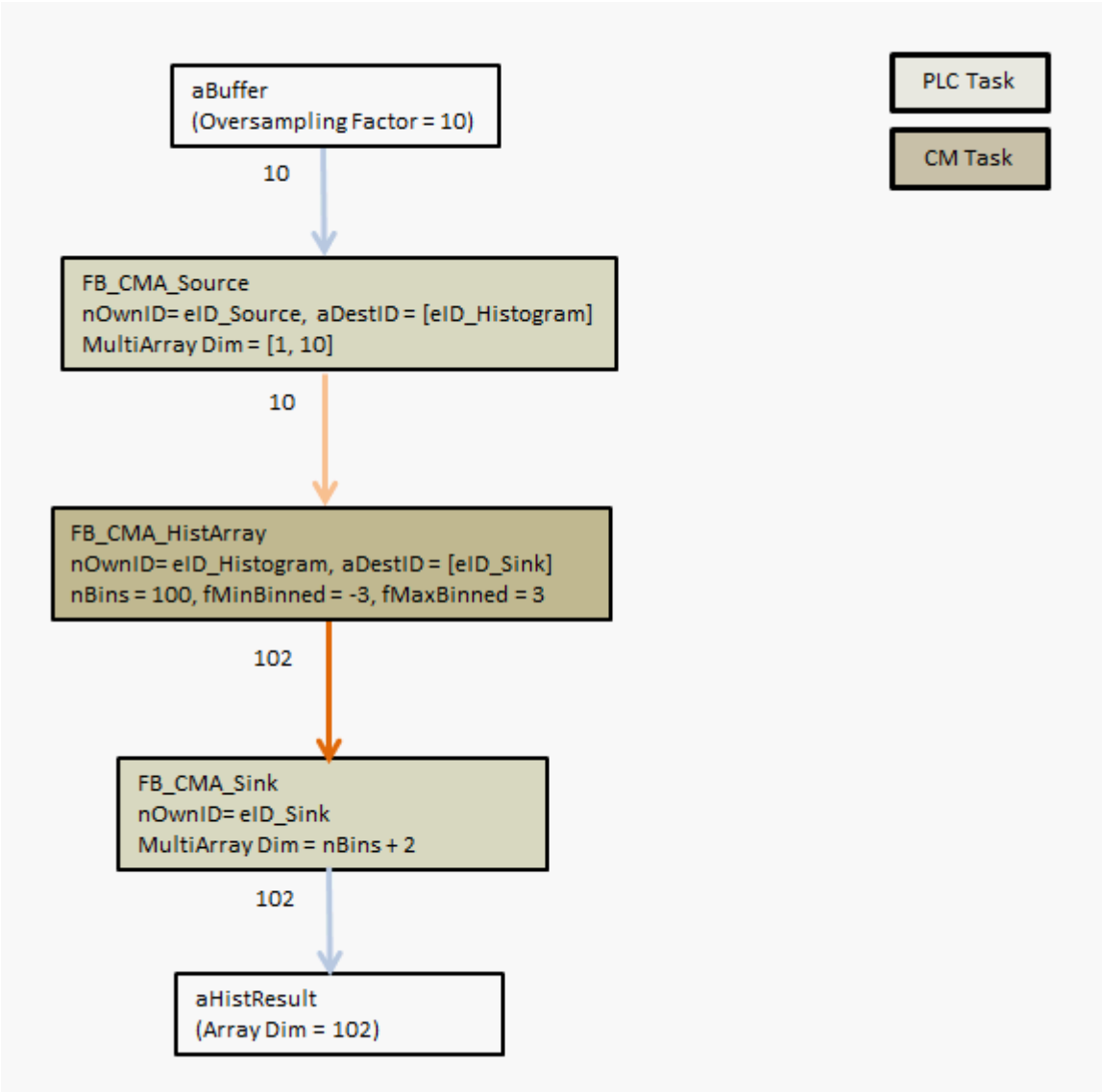
開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4016.12	PCまたはCX (x86、x64)	Tc3_CM, Tc3_CM_Base, Tc3_MultiArray

6.10 ヒストグラム

このサンプルではヒストグラムを実装しています。すなわち、コードは、EL3632などのハードウェアモジュールの入力信号を収集する制御タスクと、ヒストグラムを計算するいわゆるCMタスクの2つのタスクに分割されます。下のブロック図は解析チェーンを示しています。

サンプルのソースコードはこちらからダウンロードできます：
https://infosys.beckhoff.com/content/1033/TF3600_TC3_ConditionMonitoring/Resources/3394573195.zip

ブロック図



プログラムのパラメータ

以下の表は、ヒストグラムのファンクションブロックの設定に最も重要なパラメータを示しています：

ヒストグラムのビン	100
追加データセット	10
オーバーサンプル	10
ビンの上限（最大値）	+3 または +5
ビンの下限（最小値）	-3 または -5
チャンネル	1
バッファの長さ	100

グローバル定数

上記のパラメータは、グローバル定数として定義できます：

```
VAR_GLOBAL CONSTANT
    cBufferLength : UDINT := 100;
    cChannels     : UDINT := 1;
    cOversamples  : UDINT := 10;
    cMaxBins      : UDINT := 100;
    cAppendedData : UDINT := 10;
    cBinLimit_1   : LREAL := 3;
    cBinLimit_2   : LREAL := 5;
END_VAR
```

MAINタスクのコード

次のコードスニペットは、MAINプログラムでの宣言を示しています：

```
PROGRAM MAIN
VAR CONSTANT
    cInitSource : ST_MA_MultiArray_InitPars
    := (eTypeCode := eMA_TypeCode_LREAL, nDims := 2, aDimSizes := [1, cBufferLength]);
END_VAR
VAR
    nInputSelection : UDINT := 1;
    nSample         : UDINT;
    aEl3632 AT %I* : ARRAY [1..cOversamples] OF INT;
    aBuffer         : ARRAY [1..cOversamples] OF LREAL;
    fbSource        : FB_CMA_Source := (stInitPars := cInitSource, nOwnId
:= eID_Source, aDestIDs := [eID_Histogram]);
    fbSink          : FB_CMA_Sink := (nOwnID := eID_Sink);
    aHistReultst    : ARRAY [1..cMaxBins+2];
END_VAR
```

次のコード・スニペットは、MAINプログラムのメソッド呼び出しを示しています：

```
fbSource.Input2D(pDataIn := ADR(aBuffer),
    aDataInSize := SIZEOF(aBuffer),
    eElementType := eMA_TypeCode_LREAL,
    nWorkDim0 := 0,
    nWorkDim1 := 1,
    pStartIndex := 0,
    nOptionPars := 0);

fbSink(pDataOut := ADR(aHistResult),
    nDataOutSize := SIZEOF(aHistResult),
    eElementType := eMA_TypeCode_UINT64,
    nWorkDim0 := 0,
    nWorkDim1 := 1,
    nElements := 0,
    pStartIndex := 0,
    nOptionPars := 0);
```

CMタスクのコード

MAIN_CM プログラムの変数宣言：


```
VAR CONSTANT
    cInitHistArray : ST_CM_HistArray_InitPars := (nChannels := cChannels, nBins := cMaxBins,
    fMinBinned := -cBinLimit_1, fMaxBinned := cBinLimit_1);
END_VAR
```

MAIN_CMプログラムのメソッド呼び出し：

```
fbHistArray.CallEx(nAppendData := cAppendData, bReset := );

IF bConfig then
    fbHistArray.Configure(pArg := ADR(aHistArrayConfig), nArgSize := SIZEOF(aHistArrayConfig))
END_IF
```

Configureメソッドはオプションですが、実行時にパラメータfMinBinnedとfMaxBinnedを詳細設定できます。

乱数ジェネレータ

ヒストグラムは、振動信号のピークなど、全計測値の基本的な分布を理解するための視覚的な助けとして、非常に頻繁に使用されます。サンプルコードに含まれるファンクションジェネレータは、この目的のために拡張されています。ファンクションジェネレータを使用すると、通常の乱数や実用的な乱数と、その分布をシミュレーションできます。変数E_DistributionTypeを使用すると、指数分布、正規分布（またはガウス分布）、カイ2乗分布、ガンマ分布などを選択できます。デフォルトでは、乱数は一様分布に基づいて生成されます。



どの分布も、乱数の伝播やその範囲を決定するために、1つ以上のパラメータを必要とすることに注意してください。これは、入力変数aRangeを使用して行うことができます。

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4016.12	PCまたはCX (x86、x64)	Tc3_CM (v1.0.19), Tc3_CM_Base, Tc3_MultiArray

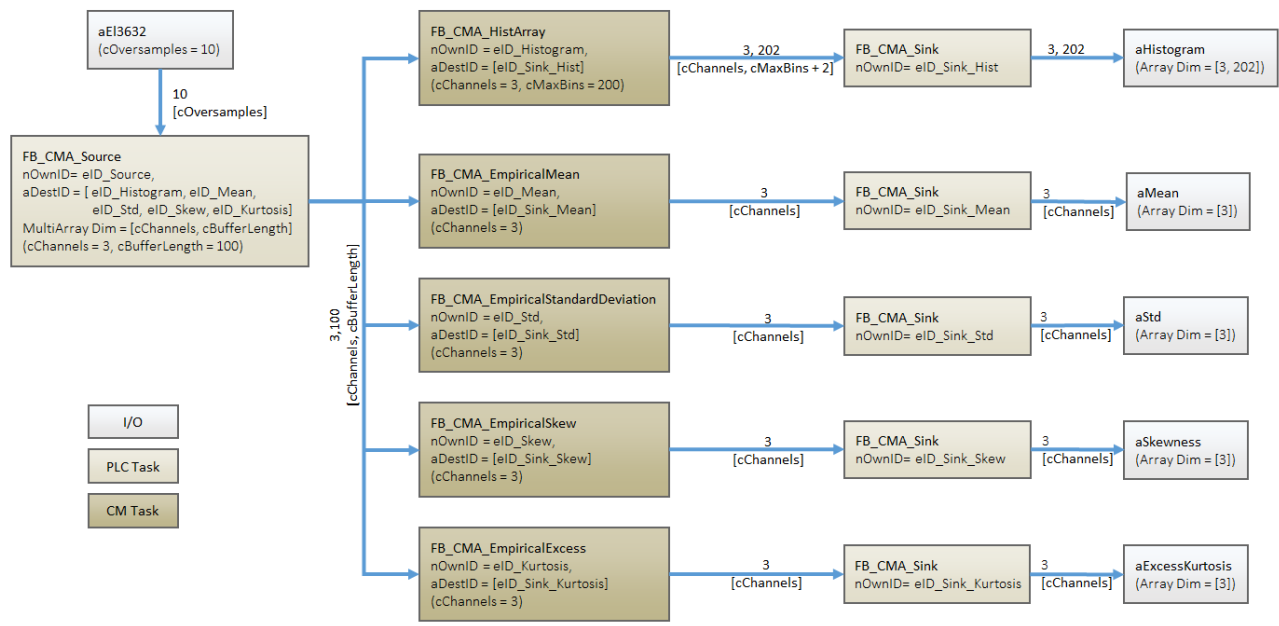
6.11 統計的手法

このサンプルは、Condition Monitoringライブラリデータの統計的評価のオプションを示しています。標準的な正規分布、ガンマ分布の信号データと正弦信号の統計的評価を行います。ファンクションブロックFB_CMA_HistArray [▶ 145]、FB_CMA_EmpiricalMean [▶ 123]、FB_CMA_EmpiricalStandardDeviation [▶ 133]、FB_CMA_EmpiricalSkew [▶ 128]、FB_CMA_EmpiricalExcess [▶ 118] を使用します。

サンプルはこちらからダウンロードできます：

https://infosys.beckhoff.com/content/1033/TF3600_TC3_ConditionMonitoring/Resources/5261532811.zip

ブロック図



プログラムのパラメータ

以下の表は、使用するファンクションブロックの設定に重要なパラメータの一覧です。

バッファサイズ	100
チャンネル	3
周波数ビン	200

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022	PCまたはCX (x86、x64)	Tc3_CM ($\geq 1.0.22$)、 Tc3_CM_Base ($\geq 1.1.10$)、 Tc3_MultiArray

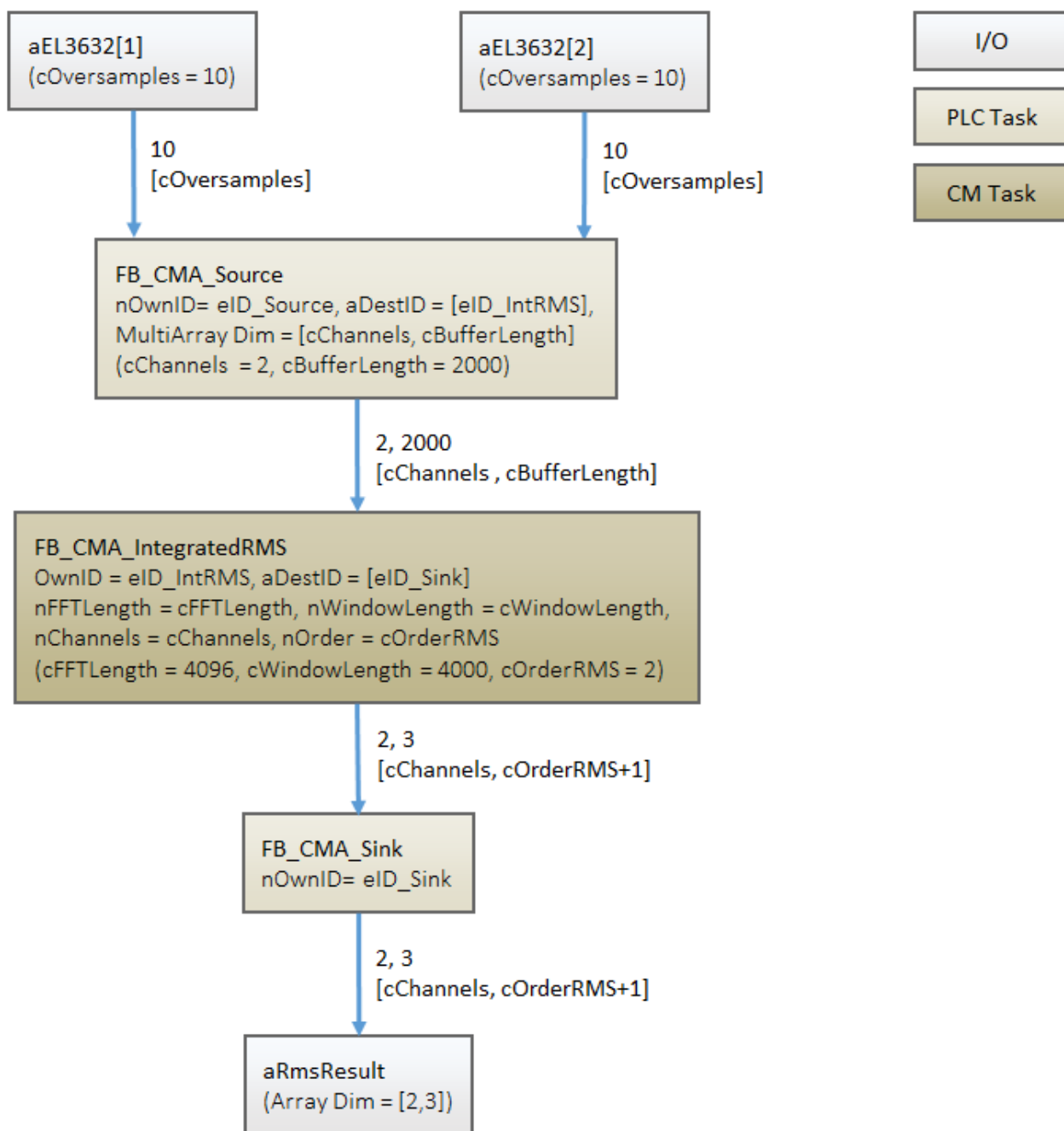
6.12 ISO 10816-3に準拠した振動評価

ISO 10816-3に基づく振動評価については、[振動評価 \[▶ 33\]](#)の「アプリケーションの概念」の項で詳しく説明しています。計算されたRMS値に基づく分類は、MAINプログラムで直接行われます。あるいは、ファンクションブロックFB_CMA_WatchUpperThresholds [\[▶ 246\]](#)またはFB_CMA_DiscreteClassification [\[▶ 113\]](#)を使用することもできます。

サンプルはこちらからダウンロードできます：
https://infosys.beckhoff.com/content/1033/TF3600_TC3_ConditionMonitoring/Resources/3394512523.zip

別の実施方法は、ISO 10816-3に準拠した振動評価（コンパクト版） [\[▶ 331\]](#)サンプルと、ISO 10816-3に準拠した振動評価（拡張版） [\[▶ 333\]](#)のサンプルに記載されています。

ブロック図



プログラムのパラメータ

以下の表は、使用するファンクションブロックの設定に重要なパラメータの一覧です。

バッファサイズ	2000
チャンネル	2
FFTの長さ	4096
ウィンドウサイズ	4000
サンプリングレート	10000
下限周波数	10

上限周波数	1000
オーダー (RMS)	2
ウィンドウタイプ	eCM_HannWindow
デシベル換算	FALSE

データ入力

サンプルでは、オーバーサンプリングが10に設定され、I/OにリンクしたPLCタスクが1msに設定されています。この結果、データ入力のサンプリングレートは10kHzになります。サンプリングの定理によれば、I/Oターミナルでアンチエイリアシングフィルタが正しく設定されていれば（[デジタル化 \[▶ 12\]](#)を参照）、5kHzまでのスペクトル信号を正確に解析できます。

データストリームのバッファリング

2つのチャンネルの入力データは、ソースファンクションブロックを含むMAINルーチン内でバッファリングされます。したがって、サイズ[cChannels, cBufferLength]の2次元配列が確立されます。DIN ISO 10816-3によると、回転速度が 600 min^{-1} 以上の場合、周波数領域は10 Hz～1000 Hzで評価する必要があります。そのため、(IntegratedRMSファンクションブロックの内部で計算される) 周波数解析の周波数分解能は、10 Hzを十分に下回る必要があります。サンプリングレート10kHzで4000サンプルのバッファを使用した場合、周波数分解能は2.5Hzとなります。Hannウィンドウを使用する場合、正式には $2.5 \text{ Hz} * 1.5 = 3.75 \text{ Hz}$ に減少します。さらに、FFT長は2の累乗で、WindowLengthより大きくする必要があります。BufferLengthは、ウィンドウが50%重なったときの長さです。内部FFTのパラメータ設定は、GVL_Constantで以下のように定義されます：

```
cFFTLenght      : UDINT := 4096;           // length of FFT
cWindowLength   : UDINT := 4000;           // 96 samples Zero padding
cBufferLength    : UDINT := cWindowLength/2; // buffer due to 50% overlap
```

したがって、上図に示すように、ファンクションブロックFB_CMA_IntegratedRMSへの転送用に、サイズ[2,2000]の配列が生成されます。

周波数選択型のRMS値計算

ファンクションブロック FB_CMA_IntegratedRMS では、FFTを実行し、転送された周波数範囲（ここでは10 Hz ～ 1000 Hz）のRMS値を計算します（正式には複数の範囲を指定可能です）。直接入力信号（加速度計が接続されている場合、通常は加速度信号）のRMS値に加えて、ファンクションブロックは各積分パラメータ、つまり振動速度のRMS 値と振動変位のRMS値も計算します。したがって、ファンクションブロックの出力は、[2,3]（2チャンネル、各チャンネルあたり3つのRMS値）の2次元配列となります。

```
// define frequency interval according to ISO 10816-3
// e.g. 10 .. 1000 Hz for rotating speed over 600r/min
cfLowerFrequencyLimit : UDINT := 10;
cfUpperFrequencyLimit : UDINT := 1000;

// Parameters for RMS calculation
cOrderRMS      : UDINT := 2;           // acceleration, velocity, and displacement
cChannels      : UDINT := 2;           // ISO 10816-3 says 2 orthogonal sensors
cResult_Length : UDINT := cOrderRMS+1; // nOrder+1 (see InfoSys)
```

上記の設定において、ソースファンクションブロックは、バッファの充填に $2000/10 = 200$ サイクル、1msを必要とします。PlcTask_CM のサイクルタイムは $0.5 * 200 \text{ ms}$ 以下にする必要があります（[タスク設定 \[▶ 69\]](#)を参照）。ファンクションブロックはわずかな計算時間しか必要としないため、PlcTask_CM のサイクルタイムは10msに設定されています。ソースファンクションブロックからタスク境界を越えたFB_CMA_IntegratedRMSへのデータ転送は、Condition Monitoringライブラリによって内部的に処理されます。

結果の解析

RMS値の計算結果は、シンク関数ブロックを介して1 msサイクルで高速PLCタスクに転送されます。この目的のため、MAINルーチンでFB_CMA_IntegratedRMS の出力の配列サイズと一致する配列を指定することが必要です（変数aRMSResult を参照）。

シンクファンクションブロックは、有効な結果が計算された場合に、フラグをTRUEに設定します（変数 `bCalculate` を参照）。

```
(* Push results to sink *)
fbSink.Output2D( pDataOut      := ADR(aRMSResult),
                 nDataOutSize  := SIZEOF(aRMSResult),
                 eElementType  := eMA_TypeCode_LREAL,
                 nWorkDim0     := 0,
                 nWorkDim1     := 1,
                 nElementsDim0 := 0,
                 nElementsDim1 := 0,
                 pStartIndex   := 0,
                 nOptionPars   := 0,
                 bNewResult    => bCalculate );
```

このフラグに基づき、RMS値の計算結果をMAINルーチンで使用できます。この場合、振動速度と振動変位のRMS値は、ISO規格で定義された制限値でチェックされます。サンプルをシンプルに保つため、単純なIFクエリを使用しています。

ISO 10816-3 に基づくクラスは、各2チャンネルごとに決定され、変数 `ISOClassIs_Vel`（振動速度の分類用）および `ISOClassIs_Displ`（振動変位の分類用）に格納されます。このサンプルの結果は、4つに分類されます。ISO 10816-3 によると、センサーが直交配置されている場合は、2つの値のうち大きい方を使用する必要があります。また、振動変位と振動速度の両方を用いる場合は、より厳密な評価を行う必要があります。したがって、4つの評価のうち最悪のケースがソースコード内で判定され、出力変数 `ISO_10816_Classification` として定義されます。

サンプルについての相互作用とコメント

サンプルでは、振幅は同じ（ 4m/s^2 ）だが周波数の異なる2つの調和振動（400Hzと35Hz）が入力変数として定義されています。この加速度振幅は、振動変位と振動速度に基づく評価では周波数400HzでクラスAに分類されることを意味しますが、振動変位に基づく評価では周波数35HzでクラスC、振動速度ではクラスDに分類されます。したがって、出力変数 `ISO_10816_Classification` はクラスDに該当します。

35Hzの振動の振幅を 1m/s^2 に変えると、分類はB（振動速度）またはA（振動変位）に変わります。したがって、変数 `ISO_10816_Classification` はBに設定されます。

あるいは、振幅は 4m/s^2 のままで、周波数を、例えば800Hzに上げることもできます。この結果、全体がクラスAに分類され、変数 `ISO_10816_Classification` はAに設定されます。

要件

開発環境	ターゲットシステム	必要なPLCライブラリ
TwinCAT v3.1.4013	PCまたはCX (x86、x64)	Tc3_CM, Tc3_CM_Base, Tc3_MultiArray

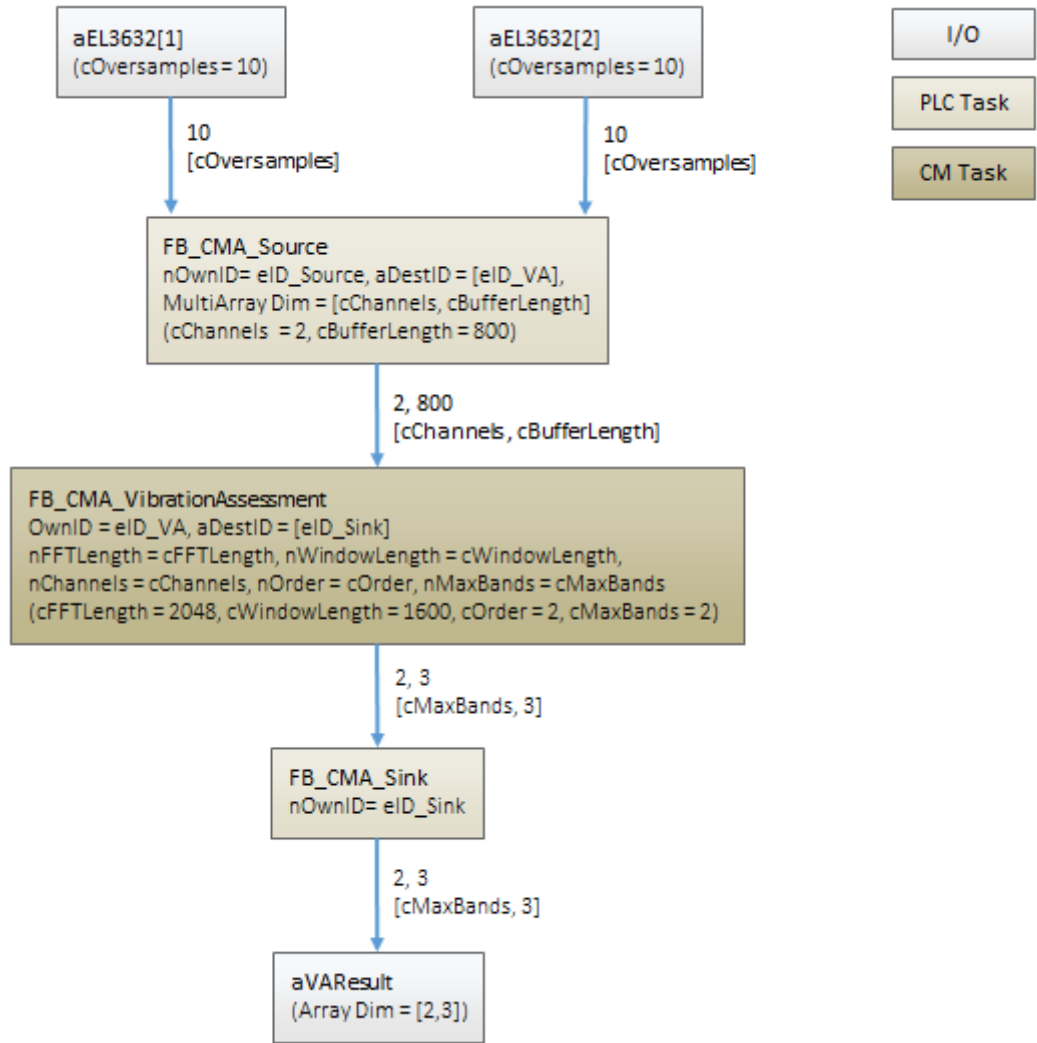
6.13 ISO 10816-3に準拠した振動評価（コンパクト版）

ISO 10816-3 に基づく振動評価については、アプリケーションの概念の項（振動評価 [▶ 33]）を参照してください。分類はファンクションブロック `FB_CMA_VibrationAssessment` [▶ 242] を使用して行われます。

サンプルはこちらからダウンロードできます：https://infosys.beckhoff.com/content/1033/TF3600_TC3_ConditionMonitoring/Resources/8236478987.zip

別の実装例は、サンプル `ISO 10816-3に準拠した振動評価` [▶ 328] とサンプル `ISO 10816-3に準拠した振動評価（拡張版）` [▶ 333] にあります。

ブロック図



プログラムのパラメータ

以下の表は、使用するファンクションブロックの設定に重要なパラメータの一覧です。

バッファサイズ	800
チャンネル	2
FFTの長さ	2048
ウィンドウサイズ	1600
サンプリングレート	10000
周波数帯域数	2
下限周波数	[10, 200]
上限周波数	[1000, 2000]
オーダー (RMS)	2
ウィンドウタイプ	eCM_HannWindow
デシベル換算	FALSE

設定

計測データの速度と変位は、ISO 10816-3 に基づく機械監視に使用されます。加速度データに基づく分類は行われません。このため、制限値を十分に高い値に設定することで、分類は暗黙的に無効化されます（GVL_Constants を参照）。

```
cISOCClassDef_Vibration : ARRAY[1..cMaxClasses] OF LREAL := [1E6, 1E6, 1E6];
cISOCClassDef_Velocity : ARRAY[1..cMaxClasses] OF LREAL := [2.3E-3, 4.5E-3, 7.1E-3];
cISOCClassDef_Displ : ARRAY[1..cMaxClasses] OF LREAL := [29E-6, 57E-6, 90E-6];
```

評価

ISO 10816-3に基づく分類の定義は、全チャンネルの速度と変位に基づき、制御タスクの中で行われます。新しい結果が得られた場合、次のように評価されます：

```
IF bCalculate THEN
  // Highscore in classification according to ISO 10816-3.
  ISO_10816_HighscoreClass := aVAResult[ISO_10816_nSelectedBand][1]; // class
  ISO_10816_HighscoreOrder := aVAResult[ISO_10816_nSelectedBand][2]; // order
  ISO_10816_HighscoreChannel := aVAResult[ISO_10816_nSelectedBand][3]; // channel

  IF NOT (ISO_10816_HighscoreClass = E_IsoClass.Error) THEN
    nCountResults := fbSink.nCntResults;
    // ToDo: if succeeded
  ELSE
    // ToDo: if error; RMS result is NaN. Code here what to do.
  END_IF
END_IF
```

結果データaVAResult には、設定した全ての周波数帯域について、以下の情報が含まれます：

- ISO_10816_HighscoreClass:設定した制限値に基づくマシン状態の分類（A-D）。
- ISO_10816_HighscoreOrder:計算された評価の積分次数（加速度0、速度1、変位2）。
- ISO_10816_HighscoreChannel:計算された評価の基本チャンネル。

このサンプルでは、評価用の周波数帯域は変数ISO_10816_nSelectedBand によって選択されます。

メモリ特性

アルゴリズムのメモリ特性は、初期パラメータbMemorize で変更できます。パラメータがTRUE に設定されている場合、メソッドResetData() が呼び出されるまでは、最も高い分類がより低い評価によって再び上書きされることはありません。ここでファンクションブロックFB_CMA_WatchUpperThresholds [▶ 246] の動作を比較します。

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM, Tc3_CM_Base



CM 3.1 では一部の機能がすでに使用可能です。[互換性 \[▶ 61\]](#)の項を参照してください。

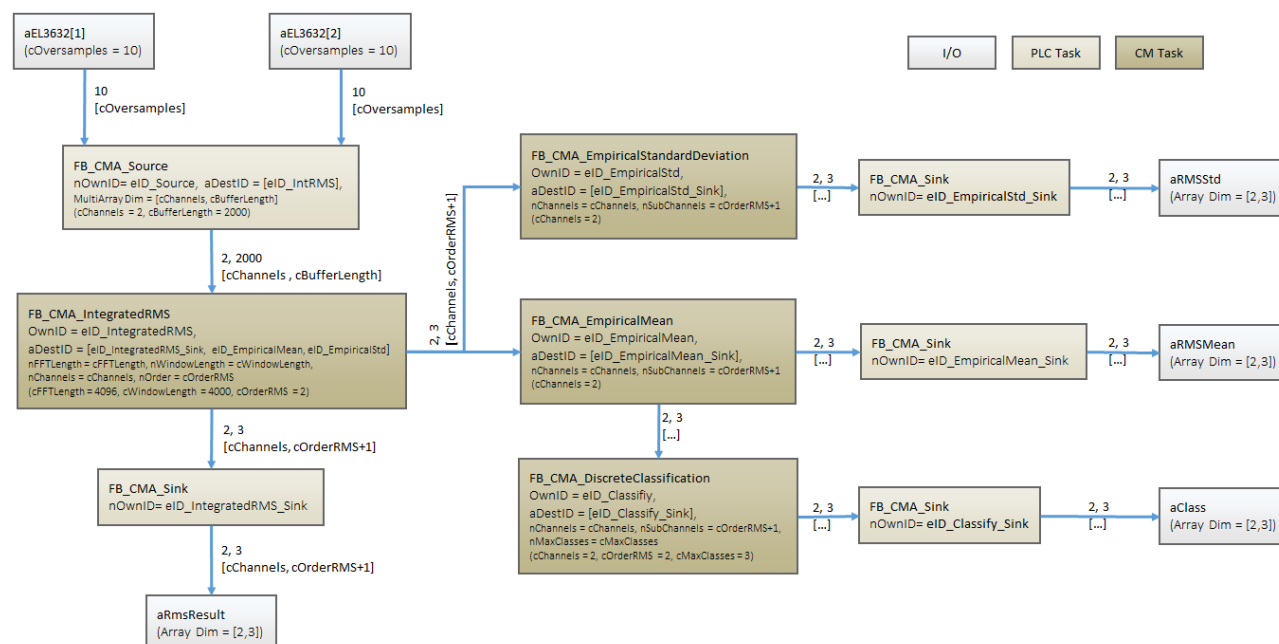
6.14 ISO 10816-3に準拠した振動評価（拡張版）

ISO 10816-3 に基づく振動評価については、アプリケーションの概念の項（[振動評価 \[▶ 33\]](#)）を参照してください。

サンプルISO 10816-3に準拠した振動評価 [▶ 328] との比較において、計算された（積分された）RMS値は、まずファンクションブロックFB_CMA_EmpiricalMean [▶ 123] を用いて平均化されます。ここでは、FB_CMA_DiscreteClassification [▶ 113] のインスタンスによる平均値に基づいた分類が行われます。統計データを使用することで、RMS値を直接処理するよりも機械評価が安定します。上記のサンプルでも、ウィンドウの長さを適宜調整することで同じ効果が得られますが、計算量は大幅に増大します。

サンプルはこちらからダウンロードできます：https://infosys.beckhoff.com/content/1033/TF3600_TC3_ConditionMonitoring/Resources/5261534475.zip

ブロック図



プログラムのパラメータ

以下の表は、使用するファンクションブロックの設定に重要なパラメータの一覧です。

バッファサイズ	2000
チャンネル	2
FFTの長さ	4096
ウィンドウサイズ	4000
サンプリングレート	10000
下限周波数	10
上限周波数	1000
オーダー (RMS)	2
クラス (分類)	3
ウィンドウタイプ	eCM_HannWindow
データセット (統計)	100
計算後リセット (統計)	TRUE

設定

計測データの速度と変位は、ISO 10816-3 に基づく機械監視に使用されます。加速度データに基づく分類は行われませんが、風力タービンのISO 10816-21など、関連規格では加速度データに基づく分類が使用されています。ここで使用される制限値は、GVL_Constants で定義されています：

```
cISOCClassDef_Acc      : ARRAY[1..3] OF LREAL := [1E6, 1E6, 1E6];
cISOCClassDef_Velocity : ARRAY[1..3] OF LREAL := [2.3E-3, 4.5E-3, 7.1E-3];
cISOCClassDef_Displ    : ARRAY[1..3] OF LREAL := [29E-6, 57E-6, 90E-6];
```

評価

算出された速度と変位のRMS値、および関連する平均値と分散（標準偏差）は、ST_Channel のインスタンスにチャンネルごとに格納されます。分類には平均値のみを使用します。分散は平均値の信頼性の尺度として使用できます。分類の結果は、設定された制限値に基づいて、E_IsoClass によって表示されます。結果のデータはスコーププロジェクトで可視化されます。



要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM, Tc3_CM_Base



CM 3.1 では一部の機能がすでに使用可能です。[互換性 \[▶ 61\]](#)の項を参照してください。

6.15 周波数解析による状態監視

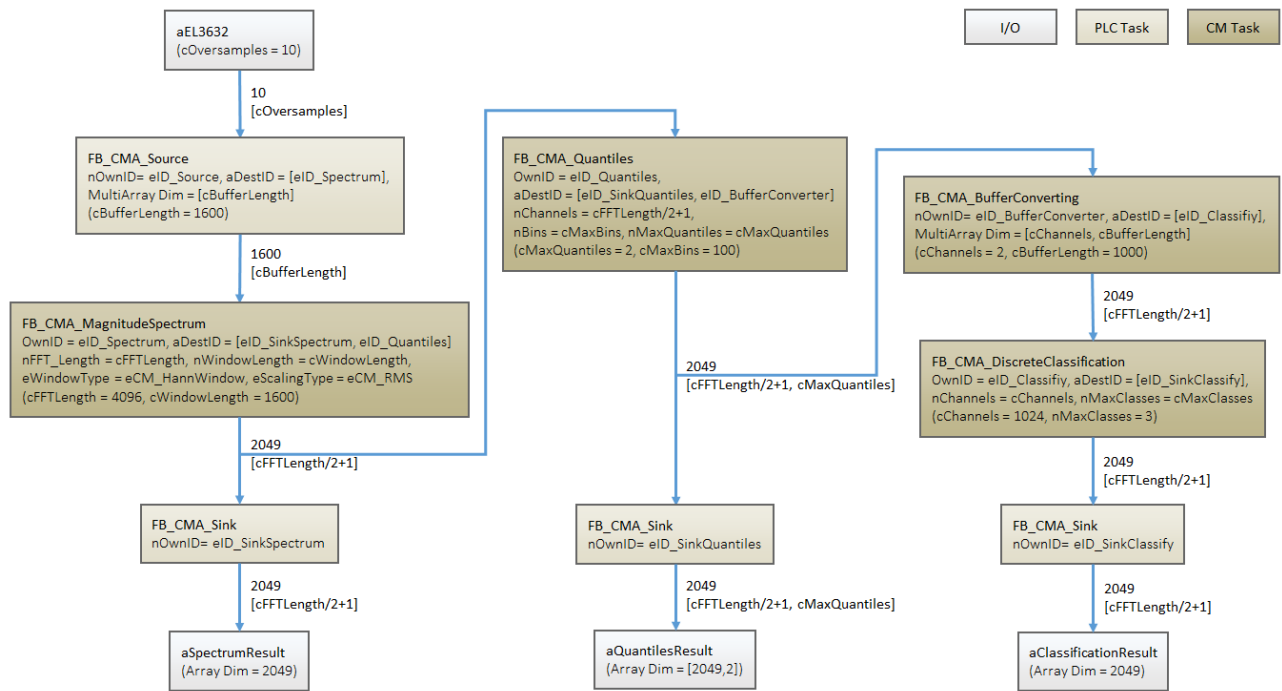
このチュートリアルでは、TwinCAT 3 Condition Monitoring APIに基づいて、包括的な監視アプリケーションを構成します。データ収集や高性能解析アルゴリズムの追加など、Condition Monitoringアプリケーションのワークフローを容易に作成できます。以下のブロック図は、アプリケーションの配置を示しています。プログラミングタスクへの理解を深めるために、この文書は設計段階ごとに細分化されています。

サンプルはこちらからダウンロードできます：

https://infosys.beckhoff.com/content/1033/TF3600_TC3_ConditionMonitoring/Resources/3394571531.zip

必要に応じて変更や拡張も可能です。

ブロック図



ステップ1 : アプリケーションの仕様

状態監視アプリケーションの設計の最初のステップは、アプリケーションの主な目的を決定することです。例えば、周波数解析に基づいて、過度の振動が発生した場合やベアリングに異常が発生した場合に自動的に警告を発するなどです。また、計測センサ、コントローラのサンプリングレート、期待される精度など、その他の技術的要因を考慮することも重要です。このチュートリアルは、マグニチュードスペクトルとその分位分布を利用して、入力信号の大小の誤差を検出することです。さらに、「正常状態」、「警告状態」、「アラーム状態」といった一般的な状態を予測するために分類器が使用されます。以下の表は、このチュートリアルで使用するファンクションブロックの一覧です。

ファンクションブロック
FB_CMA_Souce
FB_CMA_Sink
FB_CMA_MagnitudeSpectrum
FB_CMA_Quantiles
FB_CMA_DiscreteClassification

ベアリング解析 [▶ 38]、ギアユニット解析 [▶ 45]、周波数解析 [▶ 34]など、特定の問題に対するアルゴリズム選択のより詳細な説明は、別の箇所で説明されているソリューションを参照してください。チュートリアルは入力信号の全体的な変化を検出することなので、4096ラインの分解能を持つマグニチュードスペクトルで十分です。スペクトル値の50%および90%の分位値が計算され、その結果が「正常状態」、「警告状態」、「アラーム状態」に分類されます。

ステップ 2: PLCタスクの構成

状態監視と解析は、データ取得段階、計算段階、解析段階で構成されるため、各段階の計算要件に応じてタスクを構成する必要があります。このトピックに関する追加情報は、[タスク設定 \[▶ 69\]](#)にあります。このチュートリアルは、4096ラインのマグニチュードスペクトルを計算することで、これには約3200のデータサンプルが必要です。つまり、データ収集の段階で、ソースマルチアレイはオーバーラッピングを考慮して1600個のデータサンプルを処理する必要があります。10倍のオーバーサンプリングでは、1つのマルチアレイを埋めるのに160サイクル（1msのトリガーで160ms）が必要です。したがって、計算タスクには以下の設定を推奨します：

計算サイクルタイム < (データ収集サイクルタイム * バッファサイズ / オーバーサンプリング係数)*0.5

チュートリアルでは、計算サイクルタイムは10msに設定されています。実際のアプリケーションでは、ビジュアライゼーションやネットワーク通信など、同じコントローラ上で同時に実行される他のタスクの計算負荷を考慮することが重要です。タスク設定に関する詳細情報は、[タスクサイクルタイム \[▶ 69\]](#)の項にあります。Condition Monitoringアプリケーションの構築を開始する前に、[ルータのメモリ容量 \[▶ 68\]](#)が十分に確保されていることを確認してください。このチュートリアルは、ルータのメモリ容量を32MBに調整して作業するように設定されています。

ステップ3：ファンクションブロックの設定

このステップでは、アプリケーションの要件にしたがって上記のファンクションブロックを設定します。前述のように、ソースマルチアレイはスペクトルを計算するために1600のデータサンプルを収集します。したがって、aDimSizesパラメータは1600に設定します。チュートリアルでは1つのチャンネルしか考慮しないので、nDimsは1に設定します。

```
PROGRAM CM_Worker

VAR CONSTANT
    cInitSourceSpectrum      : ST_MA_MultiArray_InitPars := ( eTypeCode := eMA_TypeCode_LREAL, nDims := 1, aDimSizes := [1600]);
END_VAR
```

計算タスクでは、4096ラインのスペクトルを計算するためのマグニチュードスペクトルが設定され、cFFTLenghtで示されます。スペクトル計算は、連続信号の離散セグメントを周期的に処理するため、いわゆるウィンドウ関数が使用されます。正しく選択したウィンドウ関数は、信号変換効率を向上し、オーバーラップ加算法によって揺らぎを減らし、スペクトル分解能を向上します。実用的なアプリケーションでは、ウィンドウ関数も臨界周波数付近でのリーク効果を低減します。チュートリアルではHannウィンドウを選択しました。マグニチュードスペクトルのファンクションブロックには、幅広いスケーリングオプション [\[▶ 371\]](#)があり、その中からRMS値が選択されました。その理由は、時間的に変化する物理信号では、ピーク値と対照的に、RMS 値の方が平均信号パワーの指標として適しているためです。振動加速度スペクトルにおいて、個々の線は対応する周波数における振動のRMS値を示し、mm/s²やgなどの対応する単位で示すことができます。

```
PROGRAM MAIN_CM

VAR CONSTANT
    cInitSpectrum : ST_CM_MagnitudeSpectrum_InitPars := (      nFFT_Length := 4096,
                                                                nWindowLength := 2*1600,
                                                                bTransformToDecibel:= FALSE,
                                                                eWindowType := eCM_HannWindow,
                                                                eScalingType := eCM_RMS);
END_VAR
```

マグニチュードスペクトルの結果は、シンクファンクションブロックを介して配列にコピーされます。指定された配列長はnFFT_Length/2+1です。解析チェーンの次のステップでは、スペクトル値の50%と90%の分位を計算するための分位ファンクションブロックを構成します。多くの場合、スペクトル値は大きく変動するため、値が低すぎても高すぎても評価は困難になります。分位数を使用すると、指定した時間間隔の最大値、最小値、あるいは平均値を求めることができます。このような範囲に基づく評価は、より信頼性が高く、扱いやすい場合が多いです。50%分位値 $Q_{0.5}$ は、分布内の値の約 50% が $Q_{0.5}$ 以下となる値です。中央値とも呼ばれます。同様に、90% 分位数 $Q_{0.9}$ は、分布内の値の 90% が $Q_{0.9}$ 以下となる値を示します。

```
VAR CONSTANT
    cInitQuantiles : ST_CM_Quantiles_InitPars := ( nChannels := (4096/2+1),
                                                    fMinBinned := -10,
                                                    fMaxBinned := 10,
                                                    nBins := 100,
                                                    nMaxQuantiles := 2);
END_VAR
```

プログラムでは、分位数は以下のように設定されています：

```
(*----- Configure quantile args -----*)
IF bConfigureQuantile THEN
    FOR nChannel := 1 TO (cFFTLenght/2+1) DO
        aQuantilesArg[nChannel,1] := 0.50; // 50% quantile
        aQuantilesArg[nChannel,2] := 0.90; // 90% quantile
    END_FOR
    fbQuantiles.Configure( pArg := ADR(aQuantilesArg), nArgSize := SIZEOF(aQuantilesArg));
    bConfigureQuantile := FALSE;
END_IF
```

ここでは、ファンクションブロックFB_CMA_Quantiles [▶ 193] の詳細について説明します。パラメータ fMinBinned と fMaxBinned は予想される入力信号範囲を定義し、nBins はその信号範囲を分割するビンの数を示します。これらのパラメータは、それぞれの入力信号に依存します。信号の状態は分位数の情報に基づいて分類されます。離散ファンクションブロックは複数のチャンネルを同時に処理できるため、分位出力は直接ブロックに送信されます。分類器は3つの状態を区別し、nMaxClassesパラメータによって該当する状態を表示するように設定されています。

```
VAR CONSTANT
    cInitClassification : ST_CM_DiscreteClassification_InitPars := (nChannels:= (4096/2+1),
                                                                    nMaxClasses := 3);
END_VAR
```

分位数ファンクションブロックの出力は2次元配列で、この場合、行がスペクトル線の数、列が分位数の数を表します。しかし離散分類器では、入力チャンネル数を含む1次元配列しか使用できません。次元の衝突を避けるために、FB_CMA_BufferConverting のバッファコンバータを使用する必要があります。このファンクションブロックは、データを失うことなく2次元配列を1次元配列に変換します。コードスニペットは、対応するアプリケーションを説明しています。

```
VAR CONSTANT
    cInitBuffer : ST_MA_MultiArray_InitPars := ( eTypeCode := eMA_TypeCode_LREAL,
                                                nDims := 1,
                                                aDimSizes := [(4096/2+1)]);
END_VAR
VAR
    fbBufferConverter : FB_CMA_BufferConverting := (stInitPars := cInitBuffer, nOwnID := eID_Buffer
    Converter, aDestIDs := [eID_Classify]);
END_VAR
```

バッファコンバータはメソッドを呼び出します：

```
fbBufferConverter.Copy1D(nWorkDimIn := 0,
                        nWorkDimOut := 0,
                        nElements := 0,
                        pStartIndexIn := 0,
                        pStartIndexOut := 0,
                        nOptionPars := 0);
```

このファンクションブロックの詳細については、FB_CMA_BufferConverting [▶ 90] を参照してください。ファンクションブロックの設定を完了するには、各シンクファンクションブロックを正しい次元のPLC配列にリンクする必要があります。

ステップ4：アプリケーションパラメータの微調整

解析を開始する前に、その制限値に関して離散分類器を設定することが重要です。離散分類器は、分類制限値または閾値に基づいて入力チャンネルを連続的に監視し、閾値を超える入力チャンネルがないか判定します。閾値は、それぞれのアプリケーション、精度要件、許容される検出公差などに依存します。このチュートリアル目的は、ランダムノイズに匹敵する小さなエラーや、特定の周波数（例えば 200 Hz）で発生する大きなエラーを検出することです。閾値fWarningとfAlarmを決定します。入力チャンネルの振幅がfWarningを超えると、状態は警告状態に切り替わります。fAlarmを超えると、アラームメッセージが発信されます。閾値を超えていなければ、チャンネルの状態は正常範囲にあります。

```
(*----- Configure classifier args -----*)
IF bConfigureClassifier THEN

    fWarning := (fMonitoringLevel/100)*1.5;
    fAlarm := (fMonitoringLevel/100)*2.5;

    fbTeachTimer(IN := TRUE, PT := T#15S);
    IF fbTeachTimer.Q THEN

        fbTeachTimer(IN := FALSE);

        FOR nChannel := 1 TO (cFFTLLength/2+1) DO
            aClassArgs[nChannel, 1] := (fMonitoringLevel/100)*aQuantilesCopy[nChannel,1];
            aClassArgs[nChannel, 2] := fWarning*aQuantilesCopy[nChannel,1];
            aClassArgs[nChannel, 3] := fAlarm*aQuantilesCopy[nChannel,1];
        END_FOR

        fbClassification.Configure(pArg := ADR(aClassArgs), nArgSize := SIZEOF(aClassArgs));
        bConfigureClassifier := FALSE;

    END_IF
END_IF
```

上のコードスニペットは、この離散分類器の設定を記述したもので、タイマーブロックによって、通常の動作範囲がいわゆるティーチングフェーズを通過し、その間に離散分類器が設定されます。この間、入力信号は正常に動作する、すなわち許容範囲内にあると仮定されます。警告閾値は「正常状態」の 50% 分位値の 150%、アラーム閾値は 250% です。50%の分位値は平均的な挙動を表すため、この閾値設定は、入力に外れ値が僅かなアプリケーションに適しています。また、入力信号が大きく変動することが想定される場合には、閾値として90%の分位値を設定することもできます。また、別の変数fMonitoringLevelを設定することも可能です。この変数を使用すると、誤報の数を制御するために、許容値の周囲に一定の許容範囲を適用することができます。このパラメータは、閾値の微調整に使用できます。離散分類器の閾値は、各入力チャンネルごとに個別に指定できることに注意してください。

ステップ5 : アプリケーションの起動

コードをコンパイルしてターゲットシステムにダウンロードし、PLCを起動してチュートリアルプログラムを実行します。Dashboardという名称の簡単なビジュアライゼーション設定は、Solution ExplorerのVISUノードの下にあり、簡易テストに使用できます。シミュレーションでは、入力信号がファンクションジェネレータにリンクされており、振幅 5 の 50 Hz 正弦波信号を生成するように設定されています。また、パルス、三角波、鋸歯状波などの他の信号や、EL3632 のようなハードウェアモジュールを入力に適用することもできます。アプリケーションが起動すると、表示フィールドにPLCの最大振幅、RMS振幅、および最大振幅での周波数がリアルタイムで表示されます。

The screenshot shows the Beckhoff Dashboard application interface. It is divided into three main sections:

- Program Control:** Contains input fields for 'Set Amplitude' (5.0) and 'Set Frequency [Hz]' (50.0), and a 'Classifier Configuration' button labeled 'Reconfigure'.
- Fault Simulation:** Contains a 'Fault Amplitude' input field (5.0), two radio buttons for 'Small Fault' (selected) and 'Large Fault', and an 'Activate Disturbance' button.
- Analysis Results:** Contains three data fields: 'Max. Amplitude' (2.3368), 'Frequency [Hz]' (48.83), and 'Fault Freq. [Hz]' (2.44). Below these, there is an 'RMS Value' field (3.5584), a status indicator 'eState_Warning' (highlighted in grey), and three circular status lights (white, orange, and white).

この図では、関連する表示フィールドにアプリケーションの状態が表示されています。小さなエラーは、Add Faultボタンを押すことでシミュレーションできます。入力信号のRMS値が閾値を超えて徐々に増加し、それに応じて状態が変化する様子がわかります。大きなエラーをシミュレーションするには、Small/Largeボタンを押します。前のエラーと同様、RMS値は増加しますが、今回は「Fault Frequency」フィールドに故障信号の周波数（この場合は200 Hz）が表示されています。

ステップ6 : 監視

PLCが起動すると、表示フィールドに現在の値が表示されます。初期状態ではReconfigureボタンは押された状態で表示され、右隅のシグナルは無効になっています。これは、離散分類器に対して制限値を特定している最中であることを意味します。コンフィギュレーションが完了すると、"Reconfigure"ボタンはリセットされ、機械の状態は「正常状態 (Normal state)」と表示されます。信号は緑に切り替わります。これも同様に正常状態を意味します。

エラーをシミュレーションするには、オプションフィールドを「Small Fault」のままにして、Activate Disturbanceボタンを押してください。機械は「正常（Normal）」と「警告（Warning）」の間で切り替わり、信号は緑とオレンジの間で切り替わります。オプションフィールドの切り替えによって大きなエラーがシミュレーションされると、機械の状態は「アラーム（Alarm）」状態に切り替わり、信号は赤に切り替わります。故障を防ぐには、Activate Disturbance ボタンを解除します。信号の状態が緑に戻ります。信号振幅の変化もエラー状態になることに注意してください。これを望まない場合は、Reconfigureボタンをもう一度押して、離散分類器を新しい信号状態に合わせます。

要件

開発環境	ターゲットシステム	必要なPLCライブラリ
TwinCAT v3.1.4013	PCまたはCX (x86、x64)	Tc3_CM, Tc3_CM_Base, Tc3_MultiArray

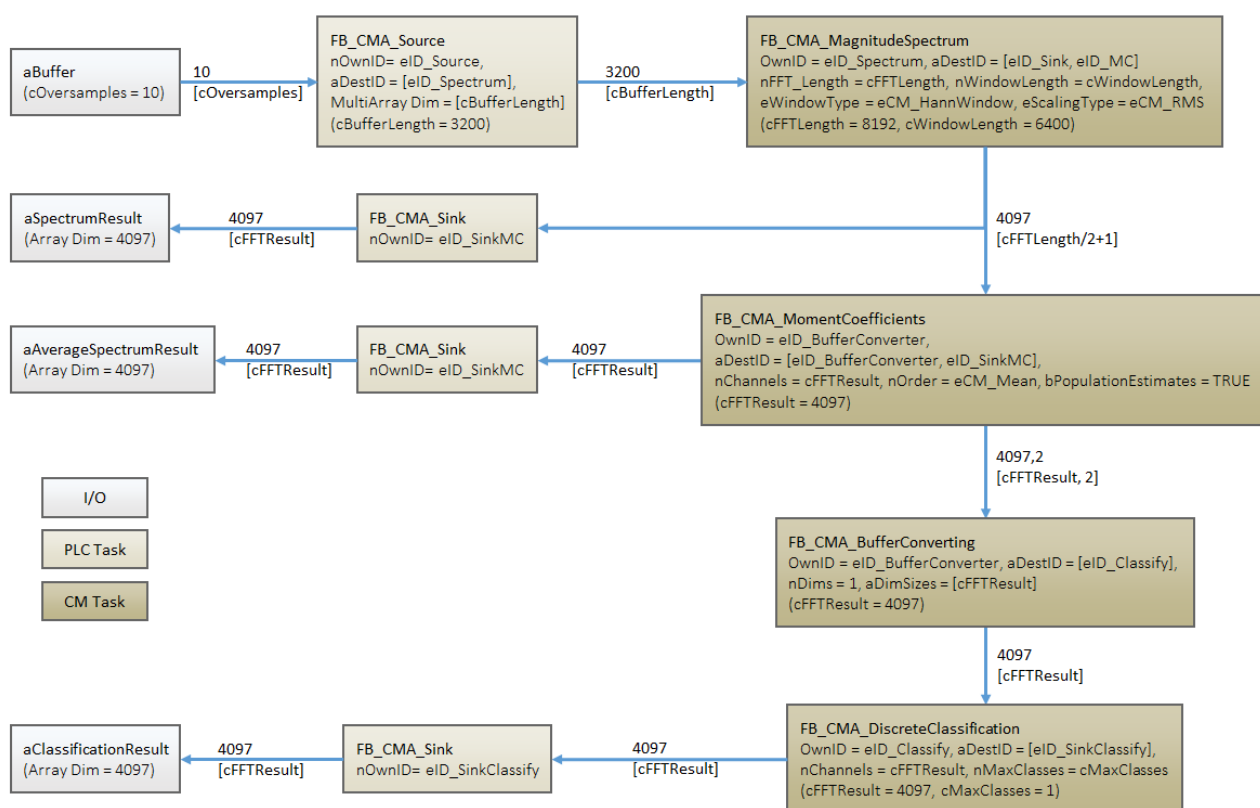
6.16 平均化したマグニチュードスペクトルに対する閾値の考慮

このサンプルは、閾値を考慮するための解析チェーンを示しています。詳細はアプリケーションの概念周波数解析 [▶ 34] で説明しています。解析チェーンは、マグニチュードスペクトルの計算、複数のマグニチュードスペクトルの平均化、およびそれに続く例示的な周波数帯域に対する閾値の検討をします。50Hz付近での閾値の考慮について、より分かりやすく説明するため、スコープは0Hzから100Hzの周波数領域に限定されています。

サンプルはこちらからダウンロードできます：

https://infosys.beckhoff.com/content/1033/TF3600_TC3_ConditionMonitoring/Resources/3394495883.zip

解析チェーンのブロック図：



プログラムのパラメータ

以下の表は、使用するファンクションブロックの設定に重要なパラメータの一覧です。

FFTの長さ	8192
ウィンドウサイズ	6400
バッファサイズ	3200
ウィンドウタイプ	eCM_HannWindow
スケーリングタイプ	eCM_RMS
係数オーダー	eCM_Mean
最大クラス数	1

要件

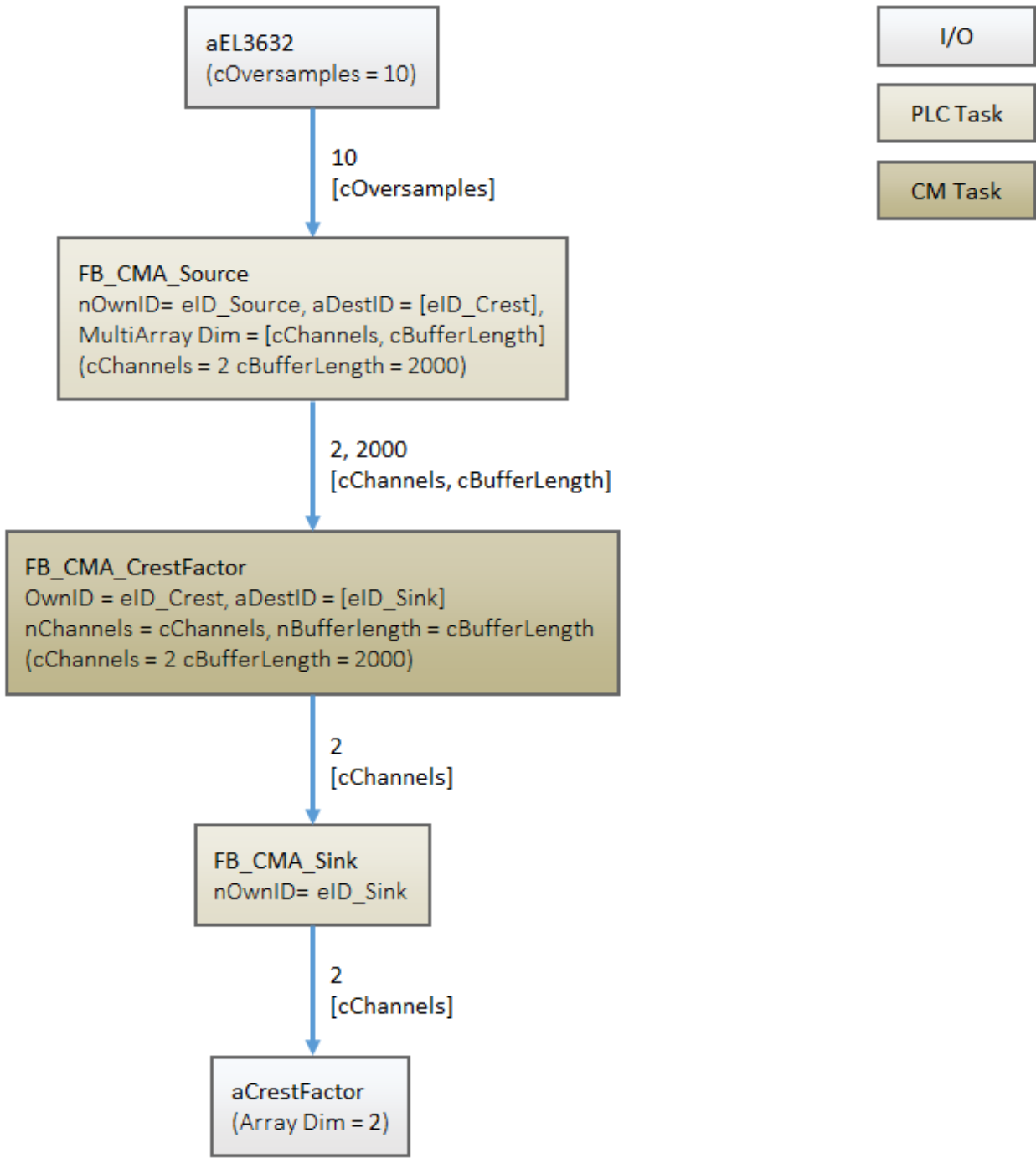
開発環境	ターゲットシステム	必要なPLCライブラリ
TwinCAT v3.1.4013	PCまたはCX (x86、x64)	Tc3_CM, Tc3_CM_Base, Tc3_MultiArray

6.17 クレストファクタ

このサンプルは入力信号のクレストファクタを計算します。ファンクションブロック `FB_CMA_CrestFactor` [▶ 93] は複数のチャンネルを処理できますが、説明のためにシングルチャンネルのみを対象とします。以下のブロック図は、プログラムに実装された解析チェーンを示しています。

サンプルのソースコードはこちらからダウンロードできます：
https://infosys.beckhoff.com/content/1033/TF3600_TC3_ConditionMonitoring/Resources/3394544267.zip

ブロック図



プログラムのパラメータ

以下の表は、クレストファクタ計算ファンクションブロックの重要な設定パラメータの一覧です。

チャンネル	2
バッファサイズ	1600

グローバル定数

これらのパラメータは、グローバル変数リストに定数として定義されています。

```
VAR_GLOBAL CONSTANT
    cOversamples : UDINT := 10;    // oversampling factor
    cChannels    : UDINT := 2;    // number of channels
    cBufferLength : UDINT := 2000; // size of buffer
END_VAR
```

制御タスクのコード

以下のコードスニペットは、MAINプログラムでの宣言を示しています：

```
VAR CONSTANT
    cInitSource : ST_MA_MultiArray_InitPars := ( eTypeCode := eMA_TypeCode_LREAL, nDims := 2,
    aDimSizes := [cChannels, cBufferLength]);
END_VAR

VAR
    nInputSelection : UDINT := 1;
    aCrestFactor    : ARRAY[1..cChannels] OF LREAL;
    nSampleIdx      : UDINT;
    nChannelIdx     : UDINT;
    aEl3632 AT %I*  : ARRAY[1..cChannels] OF ARRAY[1..cOversamples] OF INT; // input from hardware e.g. EL3632
    aBuffer : ARRAY[1..cChannels] OF ARRAY[1..cOversamples] OF LREAL;
    fbSource : FB_CMA_Source := (stInitPars := cInitSource, nOwnID := eID_Source, aDestIDs := [eID_Crest]); // Initialize source
    fbSink   : FB_CMA_Sink := (nOwnID := eID_Sink);
END_VAR
```

メインプログラムのメソッド呼び出し

```
// Collect data in a source
fbSource.Input2D(pDataIn := ADR(aBuffer),
    nDataInSize := SIZEOF(aBuffer),
    eElementType := eMA_TypeCode_LREAL,
    nWorkDim0 := 0,
    nWorkDim1 := 1,
    pStartIndex := 0,
    nOptionPars := 0 );

// Push results to sink
fbSink.Output1D(pDataOut := ADR(aCrestFactor),
    nDataOutSize := SIZEOF(aCrestFactor),
    eElementType := eMA_TypeCode_LREAL,
    nWorkDim := 0,
    nElements := 0,
    pStartIndex := 0,
    nOptionPars := 0,
    bNewResult => bNewResult);
```

CMタスクのコード

MAIN_CMプログラムでの宣言：

```
VAR CONSTANT
    cInitCrest : ST_CM_CrestFactor_InitPars := ( nChannels := cChannels, nBufferLength := cBufferLength );
END_VAR

VAR
    fbCrest : FB_CMA_CrestFactor := (stInitPars := cInitCrest, nOwnID:= eID_Crest, aDestIDs:= [eID_Sink]); // Initialize crest
END_VAR
```

MAIN_CMプログラムのメソッド呼び出し：

```
fbCrest.Call();
```

サンプルコードの結果は、入力信号として任意の振幅と周波数を持つ正弦波信号を使用して確認できます。クレストファクタ（この場合はaCrestFactorの最初の要素）は、3.01dBと等しくする必要があります。

要件

開発環境	ターゲットシステム	必要なPLCライブラリ
TwinCAT v3.1.4013	PCまたはCX (x86、x64)	Tc3_CM, Tc3_CM_Base, Tc3_MultiArray

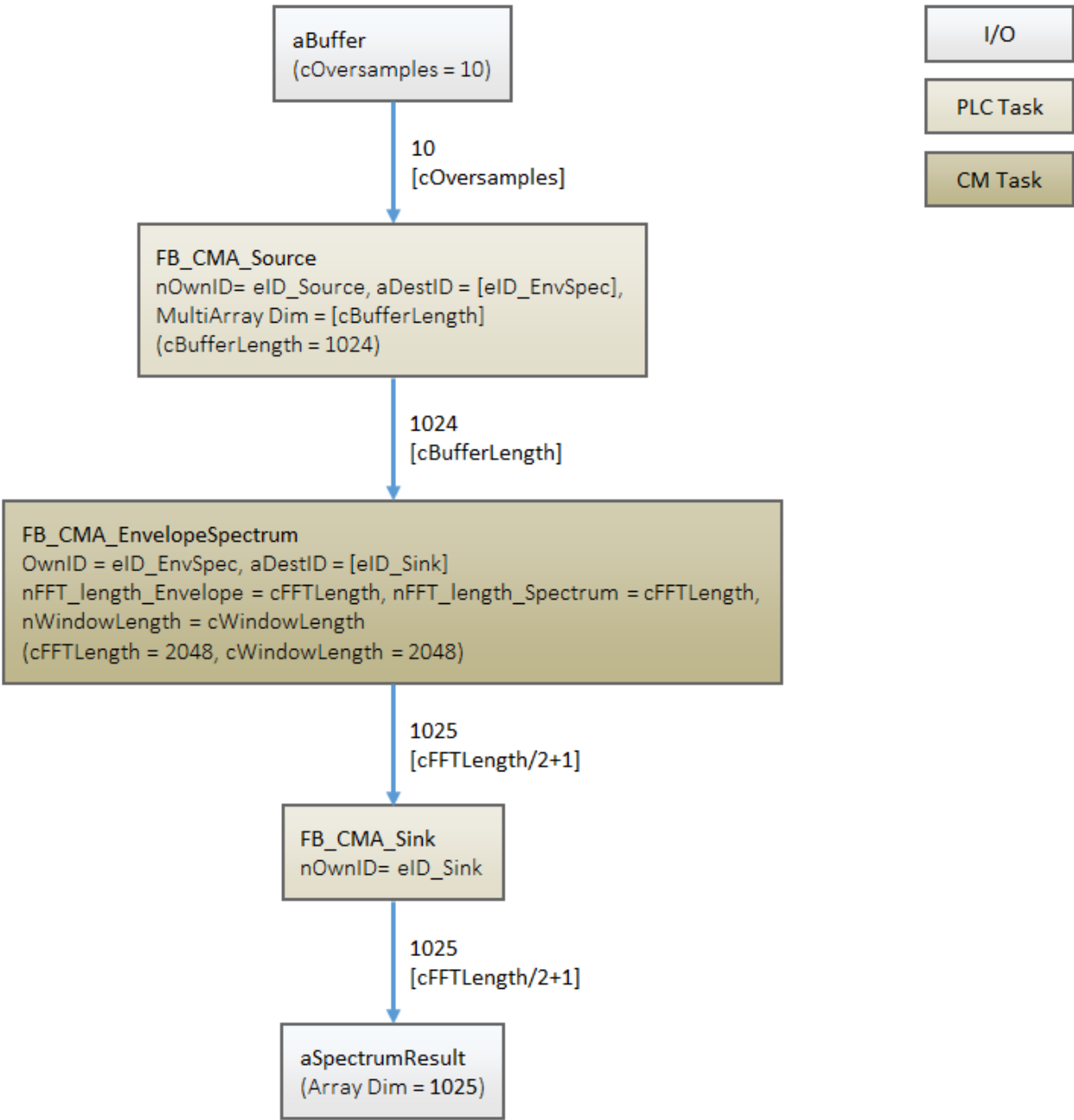
6.18 包絡線スペクトル

このサンプルは、ファンクションブロックFB_CMA_EnvelopeSpectrum [▶ 141] を使った包絡線スペクトルの計算実装を示しています。入力信号はファンクションジェネレータで生成されます。これは、120 Hz と230 Hzの2つの正弦波が重ね合わされたものに相当します。結果を分かりやすくするため、スコープは0Hzから300Hzの周波数領域に限定しています。

サンプルはこちらからダウンロードできます：

https://infosys.beckhoff.com/content/1033/TF3600_TC3_ConditionMonitoring/Resources/3394494219.zip

ブロック図



プログラムのパラメータ

以下の表は、包絡線スペクトル計算ファンクションブロックの重要な設定パラメータの一覧です。

FFT長の包絡線	2048
FFT長のスペクトル	2048
ウィンドウサイズ	2048
デシベル換算	FALSE
ウィンドウタイプ	eCM_HannWindow
スケーリングタイプ	eCM_RMS

要件

開発環境	ターゲットシステム	必要なPLCライブラリ
TwinCAT v3.1.4013	PCまたはCX (x86、x64)	Tc3_CM, Tc3_CM_Base, Tc3_MultiArray

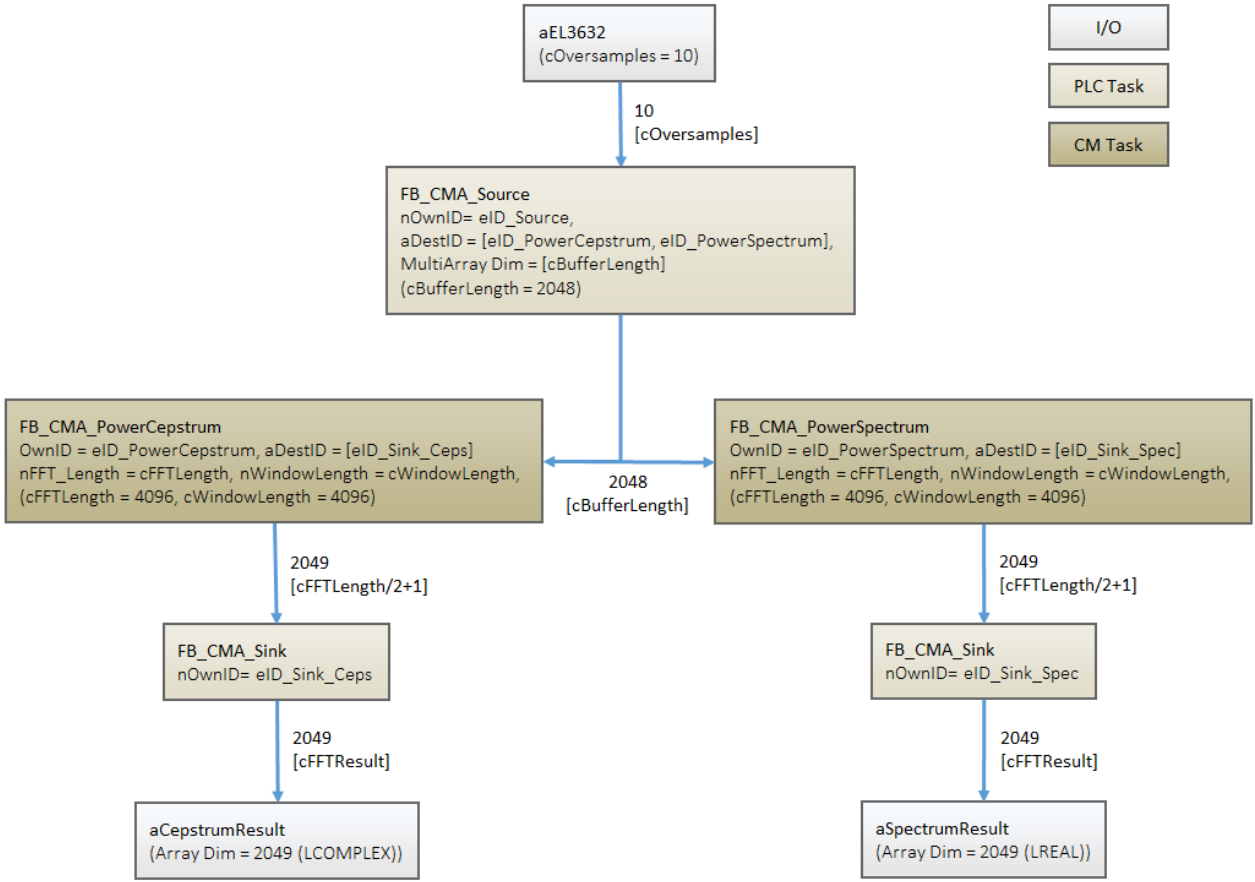
6.19 パワーケプストラム

このサンプルは、パワーケプストラムとパワースペクトルの計算実装例です。対象の信号は、搬送波周波数と変調周波数の2つの正弦波に基づく振幅変調によって生成されます。

サンプルのソースコードはこちらからダウンロードできます：

https://infosys.beckhoff.com/content/1033/TF3600_TC3_ConditionMonitoring/Resources/5261531147.zip

ブロック図



プログラムのパラメータ

以下の表は、使用するファンクションブロックの設定に重要なパラメータの一覧です。

FFTの長さ	4096
ウィンドウサイズ	4096
バッファサイズ	2048

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4018	PCまたはCX (x86、x64)	Tc3_CM, Tc3_CM_Base, Tc3_MultiArray

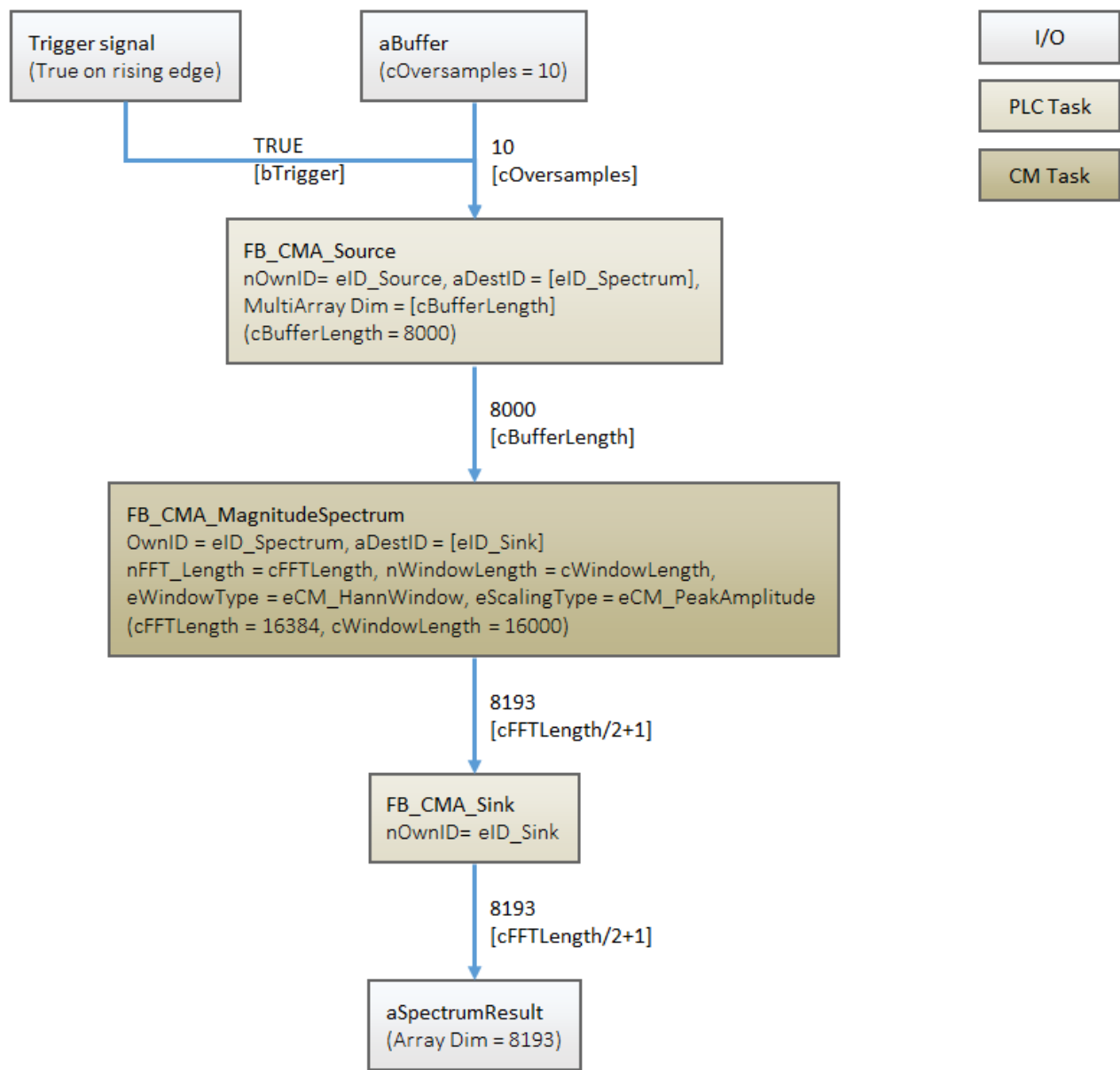
6.20 イベントベースの周波数解析

このサンプルはイベントベースの周波数解析の実装例です。生成された信号は、周波数200Hzのノイズの多い正弦信号と純ノイズからなり、2秒ごとに交互に繰り返されます。信号のバッファリングは、（生成された）入力信号の立ち上がりエッジが検出されたときに開始されます。収集されたデータは、FB_CMA_Source [▶ 230] 経由でファンクションブロック FB_CMA_MagnitudeSpectrum [▶ 161] に送信されます。

サンプルのソースコードはこちらからダウンロードできます：

https://infosys.beckhoff.com/content/1033/TF3600_TC3_ConditionMonitoring/Resources/5261425419.zip

ブロック図



プログラムのパラメータ

以下の表は、マグニチュードスペクトルのファンクションブロック設定に重要なパラメータの一覧です。

FFTの長さ	16384
ウィンドウサイズ	16000
バッファサイズ	8000
ウィンドウタイプ	eCM_HannWindow
スケーリングタイプ	ピーク振幅

入力信号のイベントベースのバッファリング

プログラムブロックCollectData は、入力信号のイベントベースのサンプリングを制御します。入力パラメータは以下のように定義されます：

```
PROGRAM CollectData
VAR_INPUT
    bTrigger      : BOOL;      // Trigger signal, start with rising edge
```

```

END_VAR
VAR_IN_OUT
    aInputSignal : ARRAY[1..cOversamples] OF LREAL; // input time signal
END_VAR

```

トリガー信号**bTrigger_**の逆数とバッファの現在の状態がローカルに保存されます。

```

VAR
    bTrigger_      : BOOL := FALSE;
    nSourceState   : UINT := 0;
    nActualBuffersSent : ULINT := 0;
    nBuffersToSent : ULINT := 2;

    // ...

END_VAR

```

イベント制御による信号のサンプリングは、トリガー信号が立ち上がりエッジを持ち、バッファがレディ状態、すなわち0状態の時に行われます。

```

IF (bTrigger AND NOT bTrigger_) AND nSourceState = 0 THEN
    nActualBuffersSent := fbSource.nCntResults; // check number of sent MultiArrays from fbSource
    fbSourceState := 1;
END_IF
bTrigger_ := bTrigger;

```

以下のコードは、ソースファンクションブロックを介した、実際のイベントベースの信号バッファリングを示しています。

```

CASE nSourceState OF

    1: // if <nBuffersToSent> MultiArrays has been sent, stop buffering

        fbSource.InputID( pDataIn      := ADDR(aInputSignal),
                           nDataInSize := SIZEOF(aInputSignal),
                           eElementType := eMA_TypeCode_LREAL,
                           nWorkDim    := 0,
                           pStartIndex := 0,
                           nOptionPars := 0);

        IF (fbSource.nCntResults-nActualBuffersSent) = nBuffersToSent THEN
            nSourceState := 2;
        END_IF

    2: // reset Source Buffer and wait for next trigger hit

        fbSource.ResetData();
        nSourceState := 0;

END_CASE;

```

バッファリングされた信号データは、その後、マグニチュードスペクトルファンクションブロックに渡されます。バッファリングされた信号は、マグニチュードスペクトル：[▶ 311] のサンプルと同様に処理されます。

要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4018	PCまたはCX (x86、x64)	Tc3_CM, Tc3_CM_Base, Tc3_MultiArray

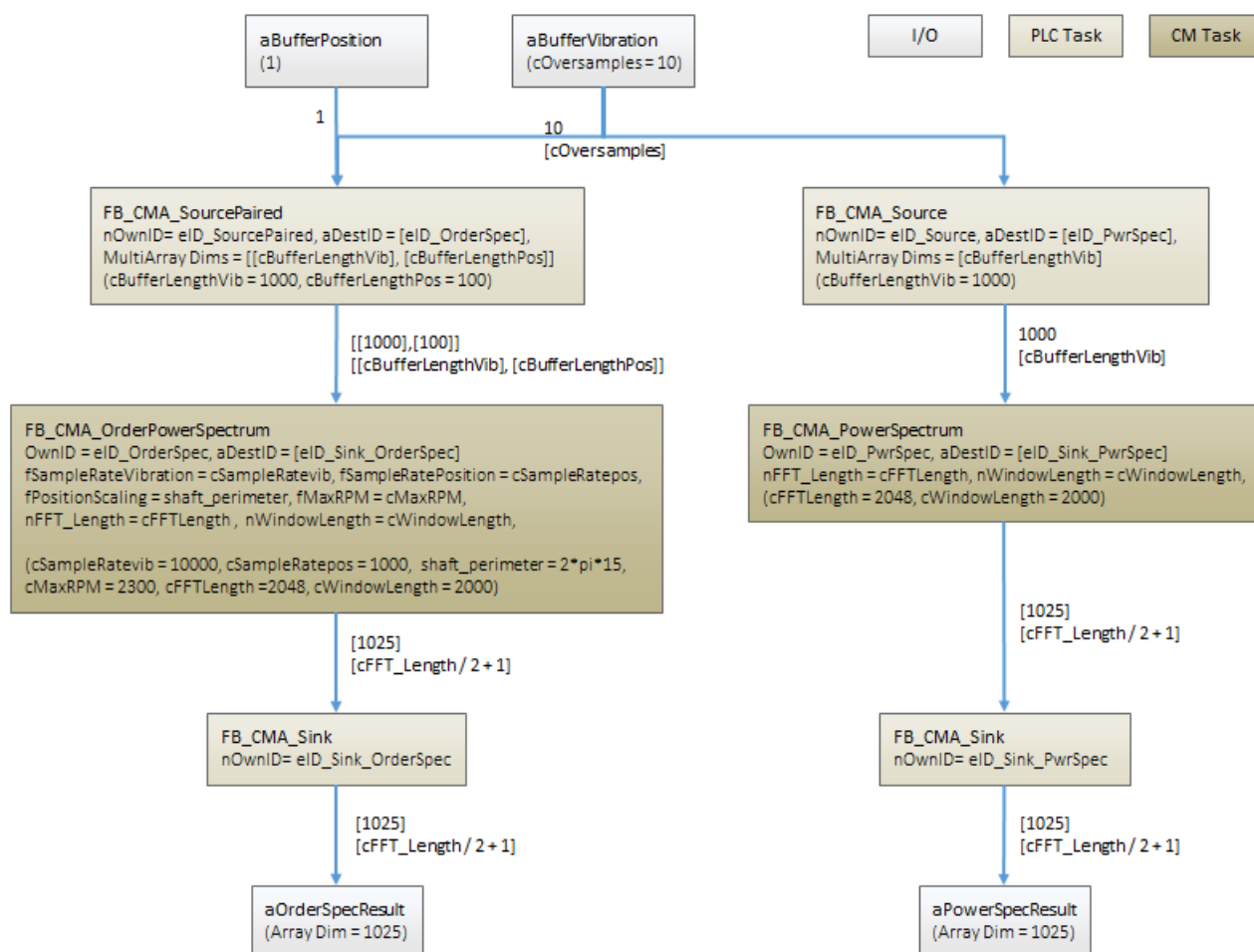
6.21 次数分析

FB_CMA_OrderPowerSpectrum [▶ 181] のサンプル実装では、シミュレーションされたNC軸と合成振動信号を使用して、OrderPowerSpectrumのパラメータ設定を例とし、通常のPowerSpectrumとの違いを示しています。

TwinCATソリューションでは、NCに500μsのサイクルタイムを使用しています。したがって、ベッコフの産業用PCまたは組み込み型PCをターゲットシステムとして使用することが望ましいです。特に他社製のPCでは、Autostart Boot Projectをオフにした方が良いです。

コードはここからダウンロードできます：https://infosys.beckhoff.com/content/1033/TF3600_TC3_ConditionMonitoring/Resources/9846479115.zip

ブロック図



プログラムのパラメータ

以下の表は、GVL_constants を参照した、次数スペクトル設定の重要パラメーター一覧です。

人工振動信号は、オーバーサンプリング係数10でPLCタスク（サイクルタイム1ms）内で生成されます。これは、10 kHzのサンプリングレートに相当します（cSampleRateVib）。位置信号は、PLCタスクによってNCから軸インターフェースを介して1ms周期で取得されます。これは、1 kHzのサンプリングレートに相当します（cSampleRatePos）。最高回転速度はcMaxRPM です。

この結果、最大でオーダー130.4まで分解可能となります。

オーダー軸の分解能はFFT長によって決まります。この場合、FFT長が2048で、分解能は0.1274となります。この値は、TwinCAT Scopeのサンプルプログラムで、ビン配列の表現をオーダーにスケーリングするために、シンボルaOrderSpecResultのScalefactor (i)で確認できます。

振動信号のウィンドウ長cWindowLength は、周波数領域への変換に使用される信号長を表します。2000サンプルが選択されました。FFT長はウィンドウ長より大きく、2の累乗で設定します。変換には、標準的なオーバーラップを持つHannウィンドウを使用し、入力バッファはcWindowLength/2、つまり1000に定義されます。振動バッファに1000要素が格納される時間内には、位置バッファには100要素が格納されます。

cSampleRateVib	10000
cSampleRatePos	1000
cShaftPerimeter	$2 \cdot \pi \cdot 15$
cFFTLength	2048
cWindowLength	2000

cBufferLengthVib	1000
cBufferLengthPos	100
cMaxRPM	2300

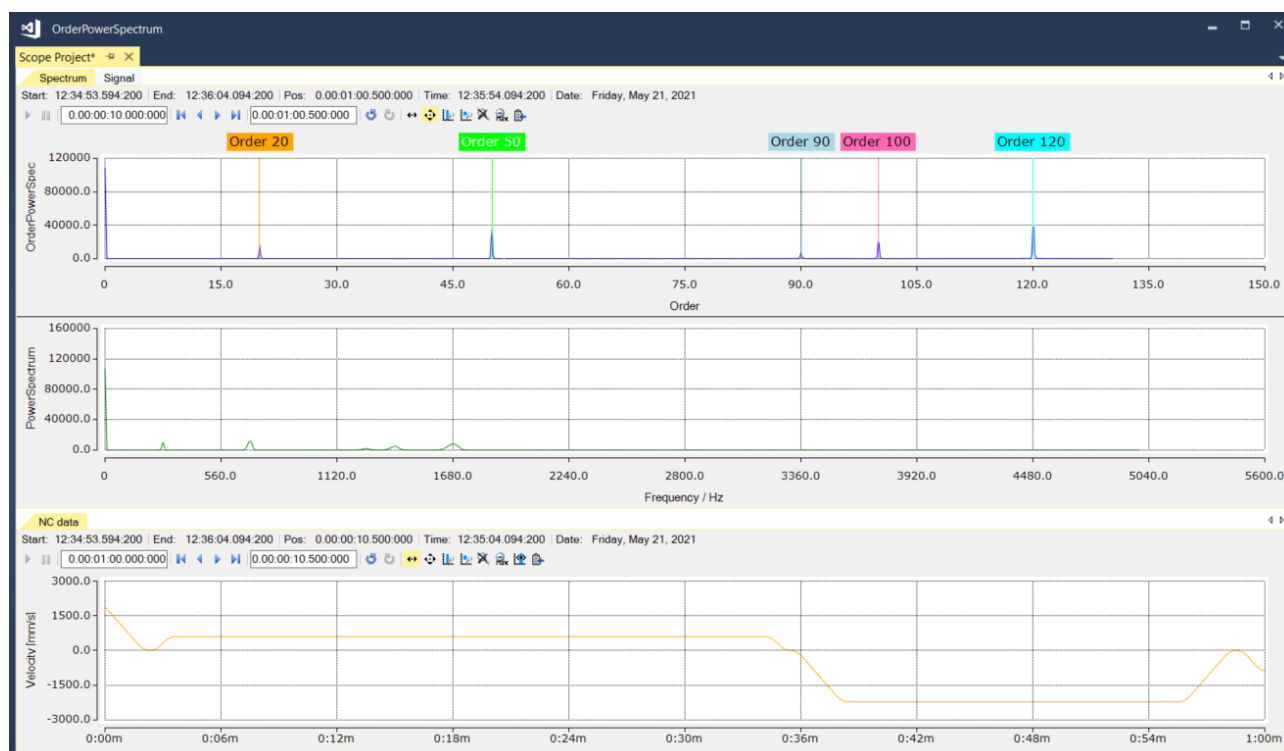
設定

ソースファンクションブロックは、シャフトの速度が最小値と最大値の間にある場合にのみ呼び出されます。上下限を超えると、解析チェーンがリセットされ、シャフト速度が有効な範囲に戻るまで一時停止します。

IF ABS(motorspeed) >= cMinRPM / 60 AND ABS(motorspeed) <= cMaxRPM / 60 THEN

定められたオーダーで特定の振幅を持つ合成信号が生成されます。配列からのオーダーは、参照用マーカーとともにスコープに表示されます。

仮想軸は自動的にその回転速度を変更しますMAIN (PRG)のnmovestate を参照してください。TwinCAT Scopeでは、OrderPowerSpectrumと通常のPowerSpectrumの結果が重なって表示されます。その下には、回転速度の時間曲線が参照用に描画されます。下のグラフのように、合成的に生成された振幅は、意図したオーダーで正確に示されています（MAIN.オーダー参照）。鋭いピークとしてオーダースペクトルに現れています。PowerSpectrumでは、シャフトの回転速度によって振幅がぼやけたピークとして現れ、速度が変動する範囲では明確に捉えにくくなります。



要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM, Tc3_CM_Base, Tc3_MultiArray

6.22 疲労分析

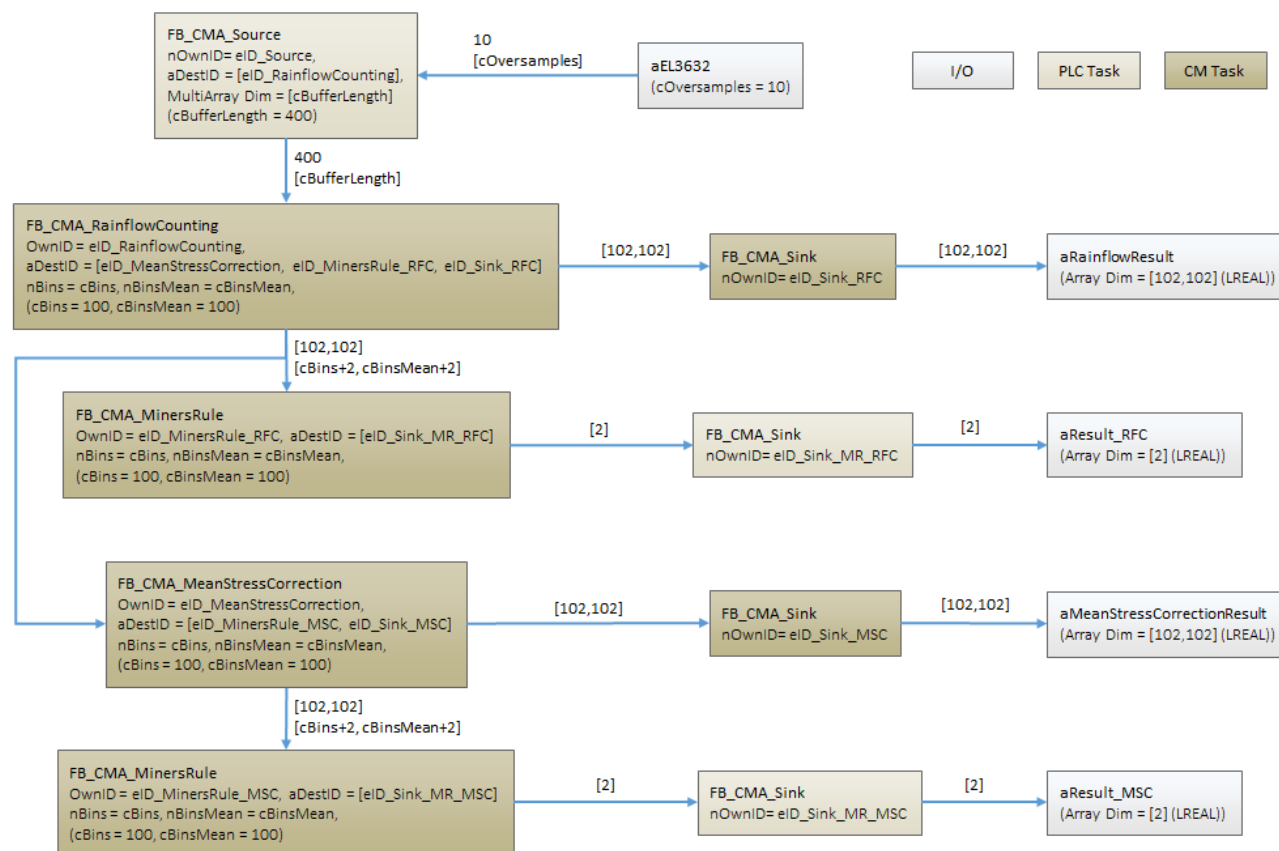
このサンプルでは、監視対象部品の疲労プロセスを推定するためのファンクションブロック [FB_CMA_RainflowCounting](#) [▶ 198] [FB_CMA_MeanStressCorrection](#) [▶ 165] および [FB_CMA_MinersRule](#) [▶ 168] のファンクションモードとアプリケーション例について説明します。

ウェラー曲線の計算には、[F_CM_CalculateWoehlerCurve](#) [▶ 270] ファンクションが架空の材料パラメータに基づいて実行されます。

詳しくは、[疲労寿命解析と損傷計算 \[▶ 49\]](#)のアプリケーションの概念 [▶ 32]の項で説明しています。

このサンプルのソースコードはこちらからダウンロードできます：https://infosys.beckhoff.com/content/1033/TF3600_TC3_ConditionMonitoring/Resources/9785658251.zip

ブロック図



プログラムのパラメータ

以下の表は、ファンクションブロックの設定に重要なパラメータ（GVL_Constants）の一覧です。

まず、機械的応力の予想計測範囲はcStressMin とcStressMax で定義されています。監視中に発生する全ての応力がこの間隔内にいることが重要です。応力サイクルの応力範囲をカウントする分解能は、cRfRangeBins およびcRfMeanBins で定義されます。デフォルトでは、最低でも64×64、通常は128×128で設定されます。ファンクションブロックのパラメータ設定をするための追加の定数は、これら4つの値で定義されます。

サンプルコードでは、平均応力補正の影響が考慮されており、TwinCAT Scopeの平均応力補正なしの計算と比較できます。サンプルはグッドマン補正を使用していますが、ガーバー補正に切り替えたり、cMSCType で無効化できます。

さらに、使用する材料固有のウェラー曲線を定義するために、材料パラメータが規定されています。ここで使用されているパラメータは架空のもので、短時間でTwinCAT Scopeに視覚的效果が現れるように選択されています。

cStressMin	-50 MPa	最小発生応力
cStressMax	50 MPa	最大発生応力
cRfRangeBins	100	応力範囲軸のビン数
cRfMeanBins	100	平均応力軸のビン数
cMSCType	eCM_Goodman	平均応力補正タイプ
cSRI	350	応力範囲切片（サイクル番号1との応力交点）
cUTS	700	極限引張強さ
cK1	3	N=1とcNC1の間のウェラー曲線の傾き

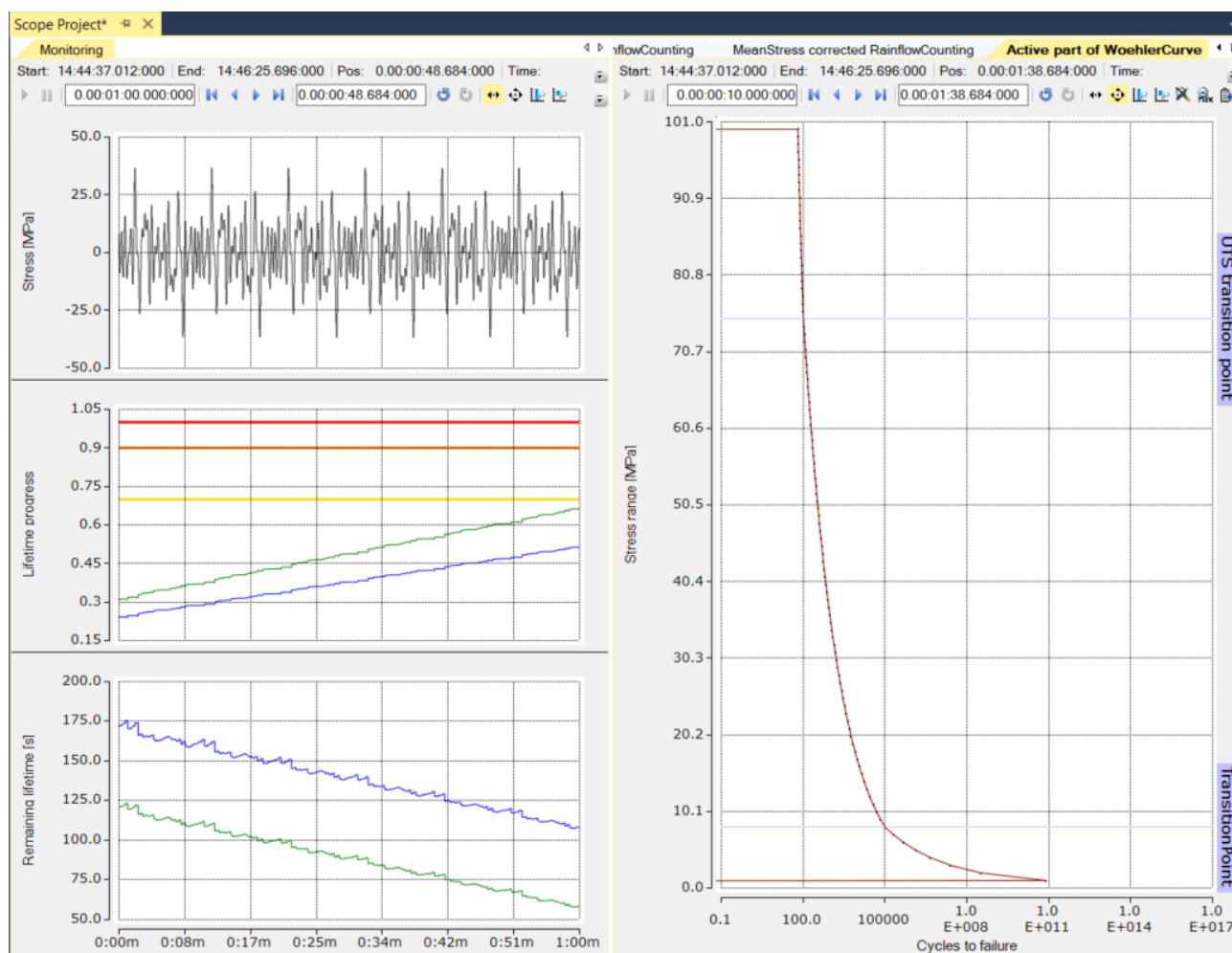
cK2	5	点cNC2から始まるウェラー曲線の傾き
cNC1	100	UTS補正の移行点
cNC2	100000	疲労遷移点

解説

TwinCATコンフィギュレーションをアクティベートしてTwinCAT Scopeを起動すると、複数の画面が表示されます。

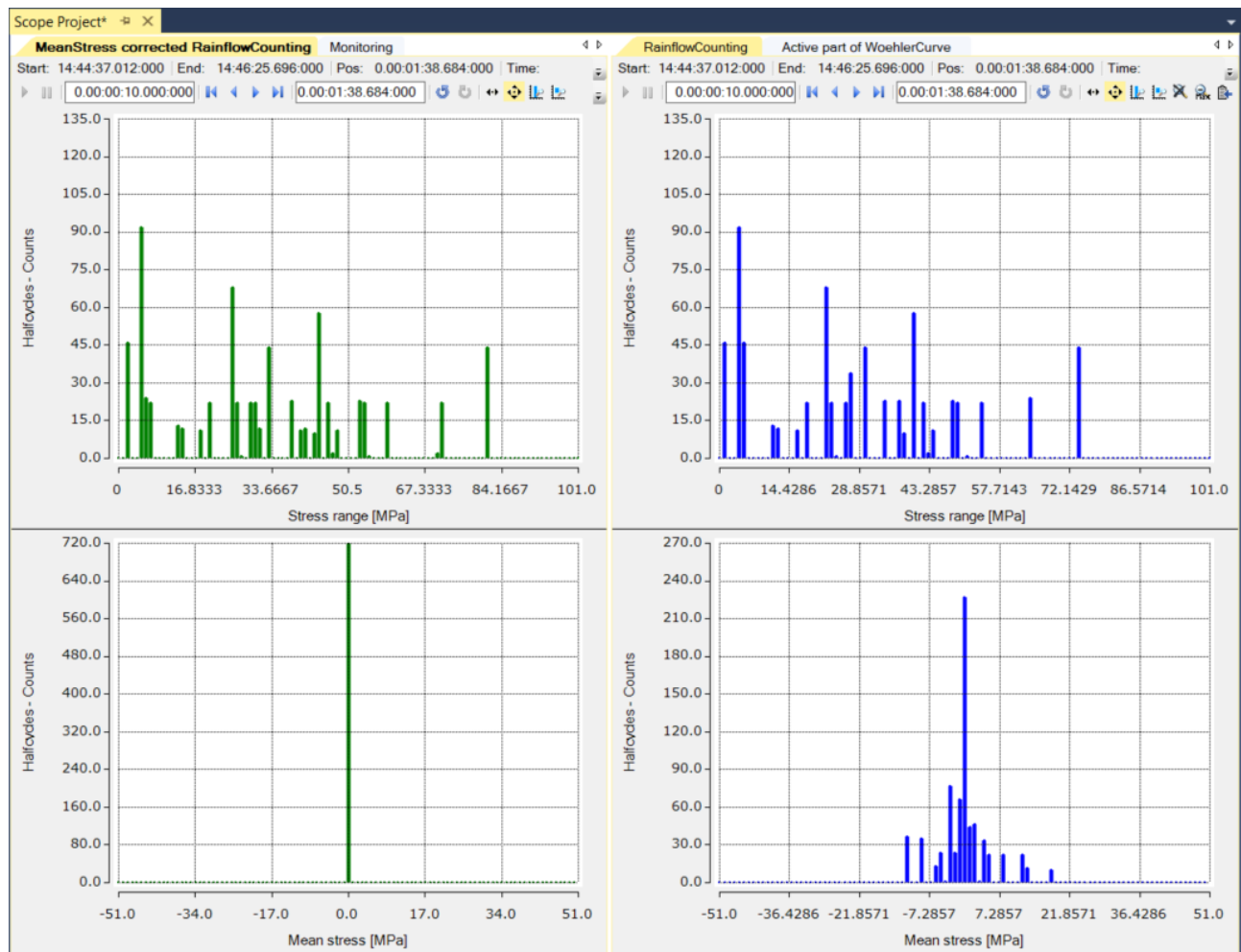
右下には、ウェラー曲線を示す静的な図が表示されます。これは、GVL_Constantのパラメータ設定に基づいて表現されます。

合成的に生成された応力信号は、左側に黒で経時的に描画されています。この例では、応力は常にメガパスカル (MPa) で表示されます。下の図では、平均応力補正を有効にした場合の損傷の合計 (緑) と、平均応力補正を無効にした場合の損傷の合計 (青) が表示されています。損傷の進行はゼロから大きくなり、1は提供された材料パラメータを使用した最大合計損傷 (計算上のコンポーネントの破損点) を表します。したがって、70%、90%、100%の位置に3本の水平線が引かれ、警告やアラームの閾値として使用できます。一番下のグラフは、推定疲労残存寿命 (秒) を示しています。したがって、この2つの曲線はシミュレーションを続けるにつれて下がっていきます。



同様に、レインフロー法ですでにカウントされたハーフサイクルは、平均応力補正ありの場合は緑色、なしの場合は青色で、配列棒グラフに示されています。平均応力補正を有効化すると、応力範囲の補正による損傷の増加が明確に確認できます。

図の左側では、平均応力補正によって平均応力軸が無効化 (ゼロに設定) され、補正が応力範囲に影響を与えていることがわかります。このため、左側のハーフサイクルカウントは、応力範囲軸上でわずかに右側にずれています。



要件

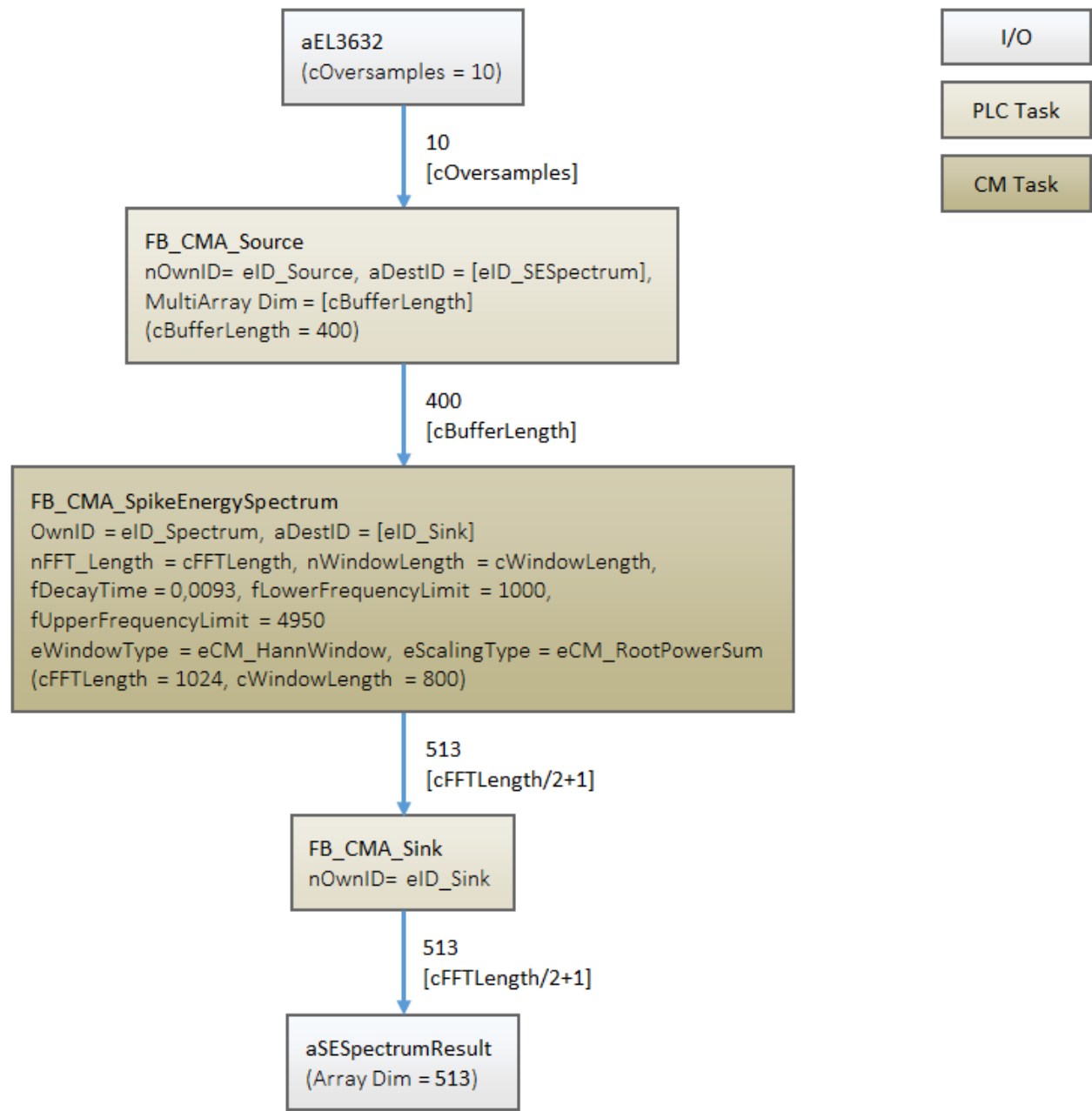
開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM, Tc3_CM_Base

6.23 スパイクエネルギースペクトル

このサンプルは、振動データの高周波成分を確認することで、ベアリング損傷を解析する手法を実装しています。実際の設定と比較してピークエネルギーの感度が高いため、模擬ショックのエネルギーは意図的に大きくされています。使用にあたっては、経時的なピーク値スペクトルの傾向を考慮する必要があります。

このサンプルのソースコードはこちらからダウンロードできます：https://infosys.beckhoff.com/content/1033/TF3600_TC3_ConditionMonitoring/Resources/9783190923.zip

ブロック図



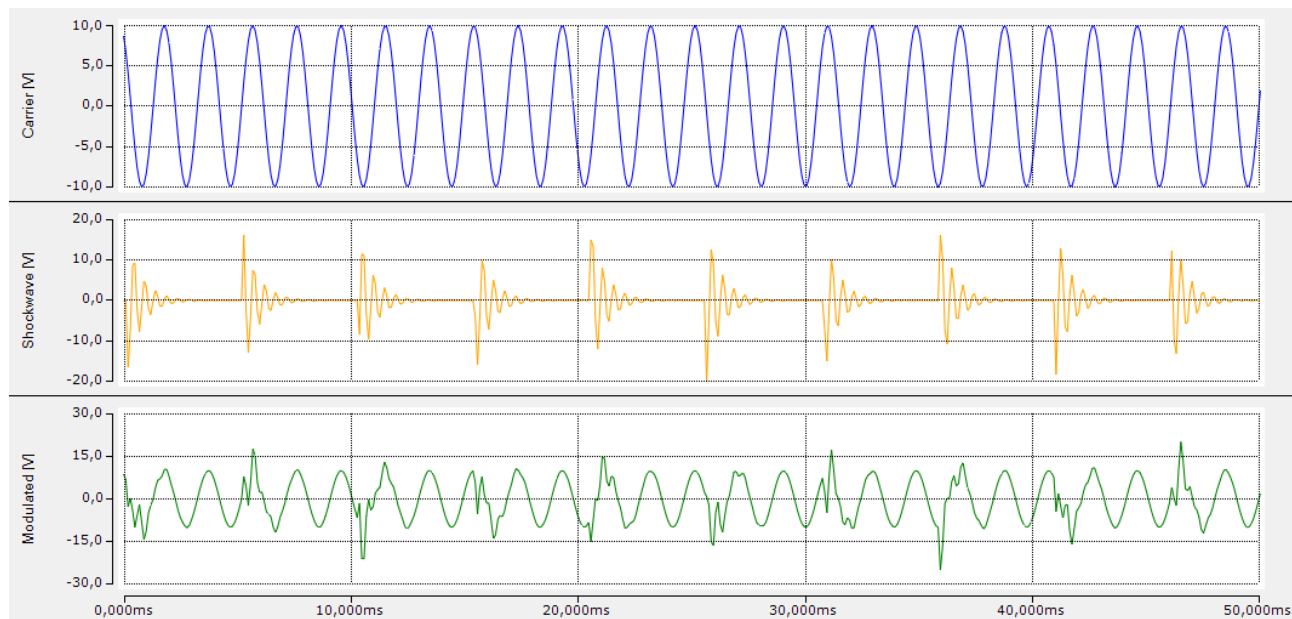
プログラムのパラメータ

以下の表は、ファンクションブロックを設定するための重要なパラメータの一覧です。

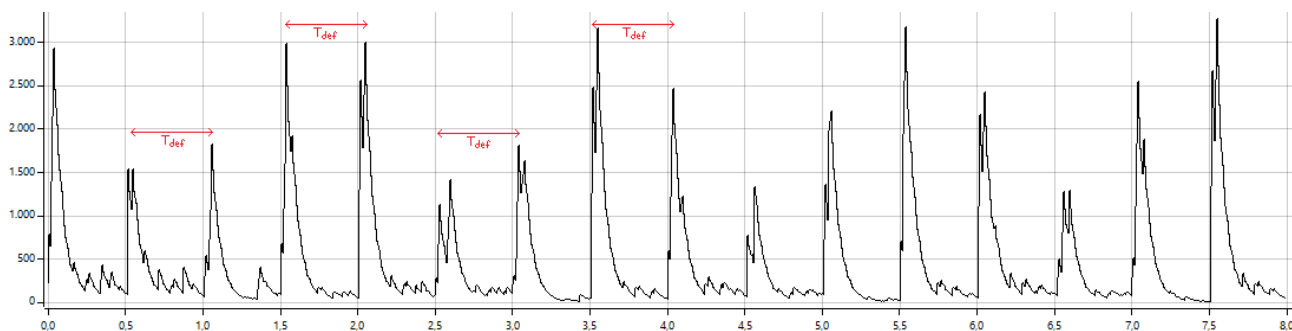
FFTの長さ	1024
ウィンドウサイズ	800
バッファサイズ	400
ピーク値スペクトルの長さ	513
バンドパスフィルタの下限 [Hz]	1000
バンドパスフィルタのカットオフ周波数 [Hz]	4950

解説

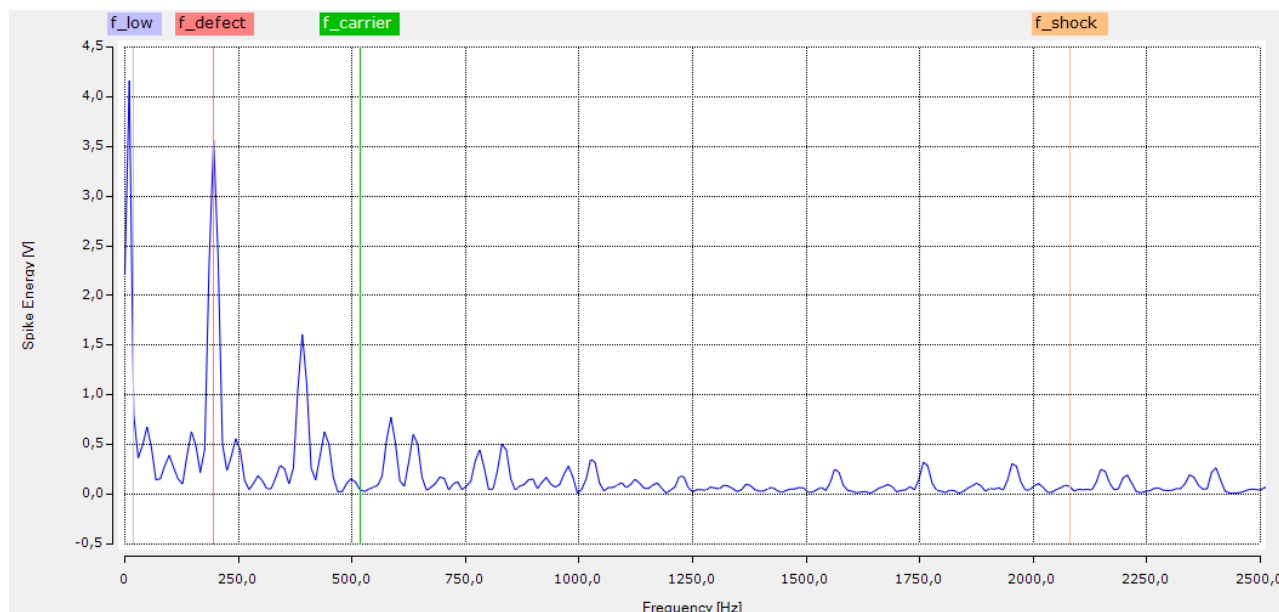
生成された振幅変調入力信号は次のように構成されています。基本振動（青）に、周波数200 Hzの衝撃（黄）が変調されています。したがって、ファンクションブロックの入力信号は以下の曲線（緑）に対応しています。



入力データは次のように処理されます：解析に不要な信号成分（1 kHz 以下）は、IIR バンドパスフィルタで除去されます。正確な算術の場合、これは衝撃（黄）に対応します。フィルタリングされた信号は、設定された減衰時間($\tau_{DecayTime}$) ("peak to peak") に応じてエネルギーピークが表示される時間波形に変換されます。次の画像は、上記の入力データの時間波形を示しています。



発生するピーク値の周波数、つまり元の信号の衝撃の形をした高周波成分の繰り返し率は、この表現からすでに認識できます。連続する2つのピークの間隔は0.5ms、すなわち $1 / 0.005 \text{ s} = 200 \text{ Hz}$ です。この周期性は、マグニチュードスペクトルの計算によって表されます。このため、サンプルのスコーププロジェクトでは、次のように可視化されます：



その結果、ピーク値のスペクトルでは、200Hzのエラー周波数(f_{defect})と、それに関連する高調波の成分がはっきりと認識できます。さらに、基本振動の周波数成分(f_{carrier})と衝撃の周波数成分 (f_{shock}) は、バンドパスフィルタリングと時間波形への変換によって除去されます。

f_{low} 以下のピーク値は、分解能の限界である $c_{\text{MinFrequency}} = c_{\text{SampleRate}}/c_{\text{WindowLength}} = 12,5 \text{ Hz}$ を下回っているため意味を持ちません。

要件

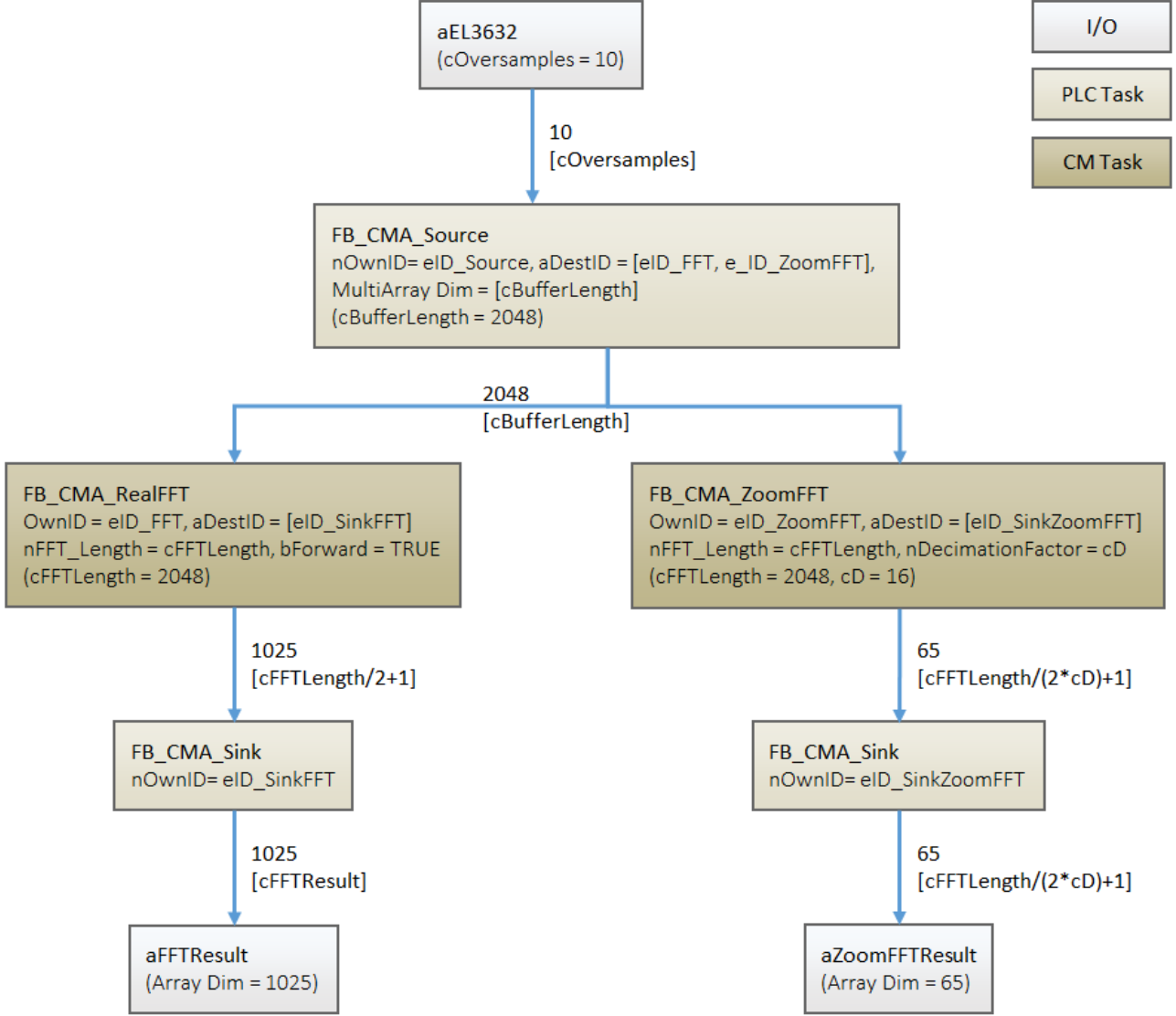
開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM, Tc3_CM_Base

6.24 ズームFFT

このサンプルは、ファンクションブロック FB_CMA_ZoomFFT の使用例を示しています。このファンクションブロックを使用すると、FB_CMA_RealFFT で計算されたスペクトルの一部を計算できます。

このサンプルのソースコードはこちらからダウンロードできます：https://infosys.beckhoff.com/content/1033/TF3600_TC3_ConditionMonitoring/Resources/9783192587.zip

ブロック図



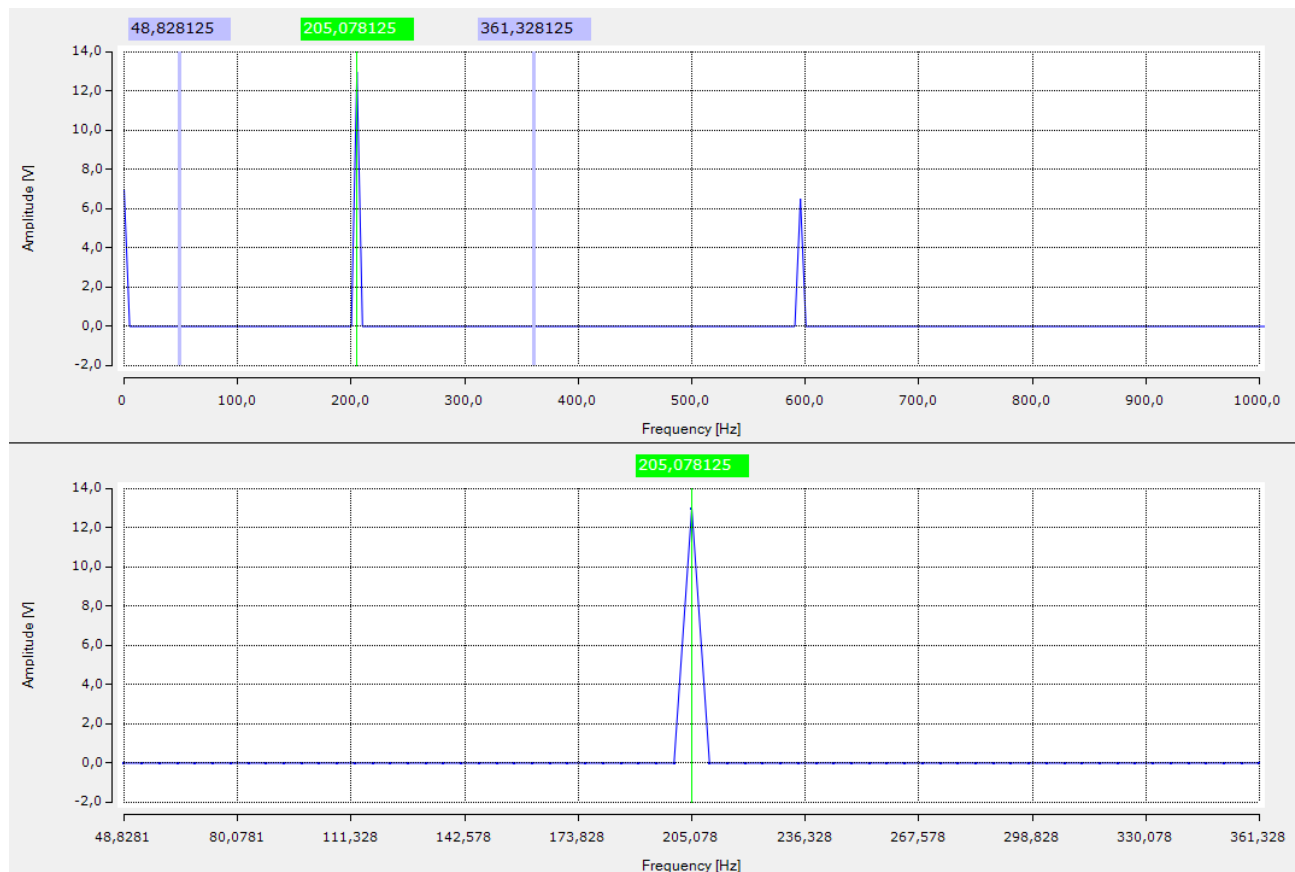
プログラムのパラメータ

以下の表は、ファンクションブロックを設定するための重要なパラメータの一覧です。

FFTの長さ	2048 / 2048
バッファサイズ	2048 / 2048
デシメーション係数	16 / -
計算されたスペクトルの長さ	65 / 1025

解説

サンプルコードに含まれるスコーププロジェクトでは、ファンクションブロック**FB_CMA_RealFFT**で計算された0~1 kHzのスペクトルと、ファンクションブロック**FB_CMA_ZoomFFT**で計算された48.83 Hzと361.33 Hzのスペクトルが表示されます。ここでは、周波数帯域の分解能は同じ（4.8828125 Hz）であり、スペクトル全体には長さ 2048 の FFT が、ズーム FFT を使用したセクションには長さ 128 の FFT が、デシメーションされた時間信号に対して計算されます。



要件

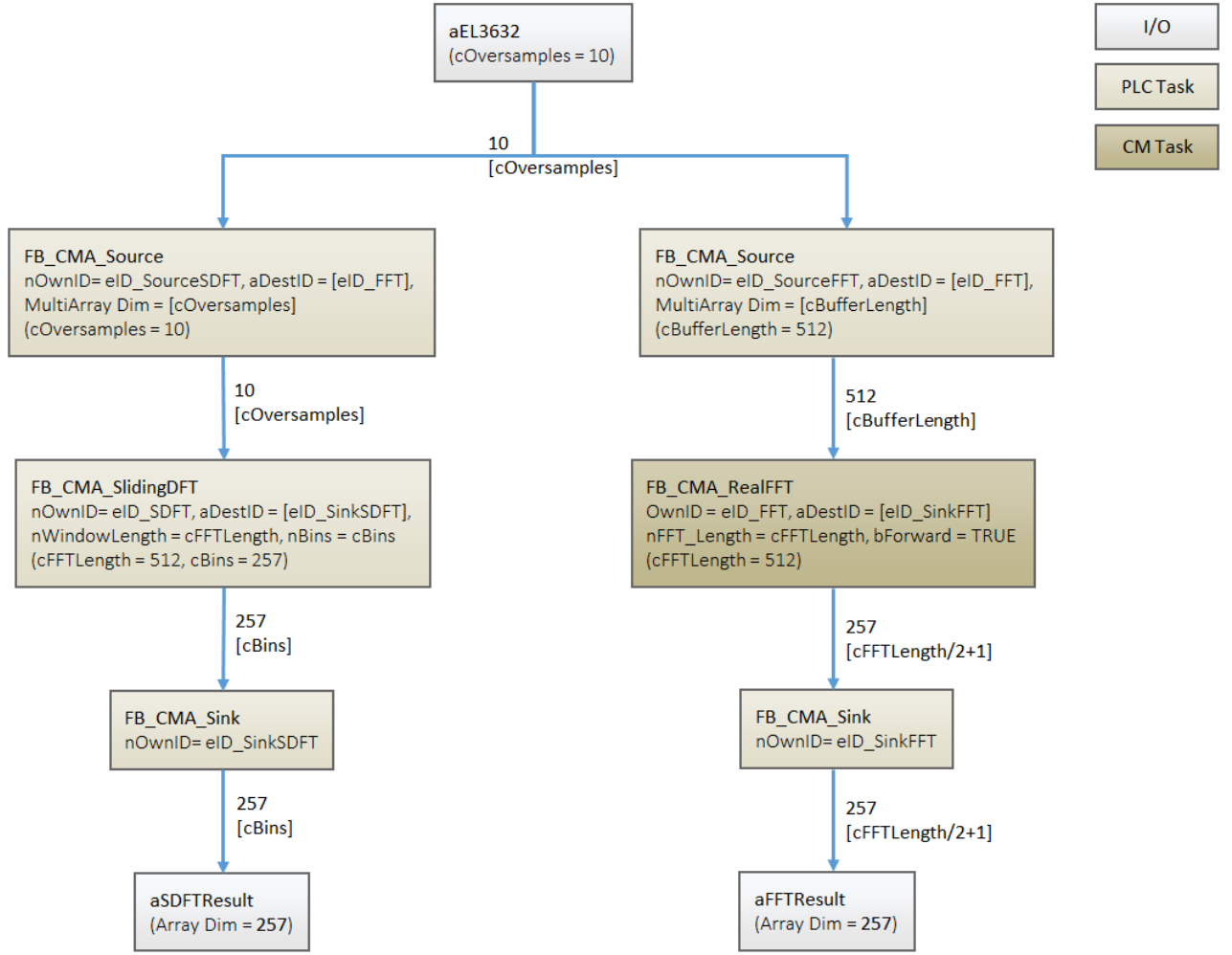
開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM, Tc3_CM_Base

6.25 スライディングDFT

このサンプルは、ファンクションブロックFB_CMA_SlidingDFT の使用例を実装したものです。全てのスペクトル値が計算され、FB_CMA_RealFFTによる計算結果と比較するようにされるように構成されています。さらに、計算されたスペクトルを改善するため、スペクトル範囲にHannウィンドウが適用されます。

このサンプルのソースコードはこちらからダウンロードできます : https://infosys.beckhoff.com/content/1033/TF3600_TC3_ConditionMonitoring/Resources/9783189259.zip

ブロック図



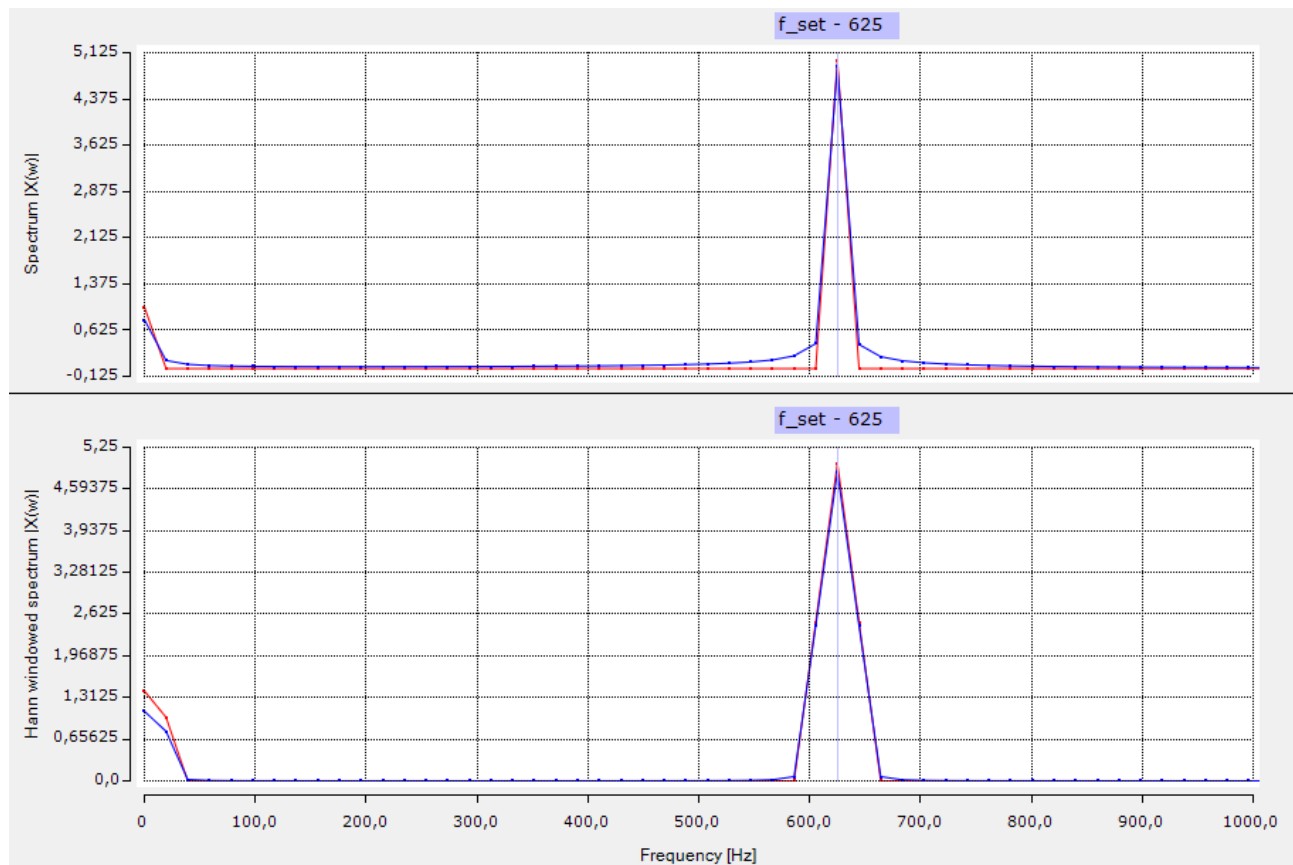
プログラムのパラメータ

以下の表は、ファンクションブロックを設定するための重要なパラメータの一覧です。

FFTの長さ	512 / 512
バッファサイズ	10 / 512
FFT結果の長さ	257 / 257

解説

サンプルに含まれるスコーププロジェクトは、計算されたスペクトルを0～1kHzの範囲で可視化します。一方で、ファンクションブロック **FB_CMA_SlidingDFT**（青線）のスペクトルでは、サイドローブの影響が顕著に現れます。この偏差は、基本的に選択された減衰パラメータ**fDampingFactor** に依存します。0.995 を超えると偏差は小さくなるが、減衰量(>0.999)が少なすぎると計算が不安定になります。スペクトル線k = 0 の再帰則の定義により、直流成分の計算はより正確です。計算値の改善は、スペクトル範囲にHannウィンドウを適用することで達成できます（下図）。



この2つのファンクションブロックの大きな違いは、それぞれのタスクにおける利用率です。これは次の図に示しています。



RealFFTの利用率のピーク（下図）とは対照的に、SDFTの利用率（上図）はより安定しています。後者の値は平均してかなり高いですが、変動は少ないです。

要件

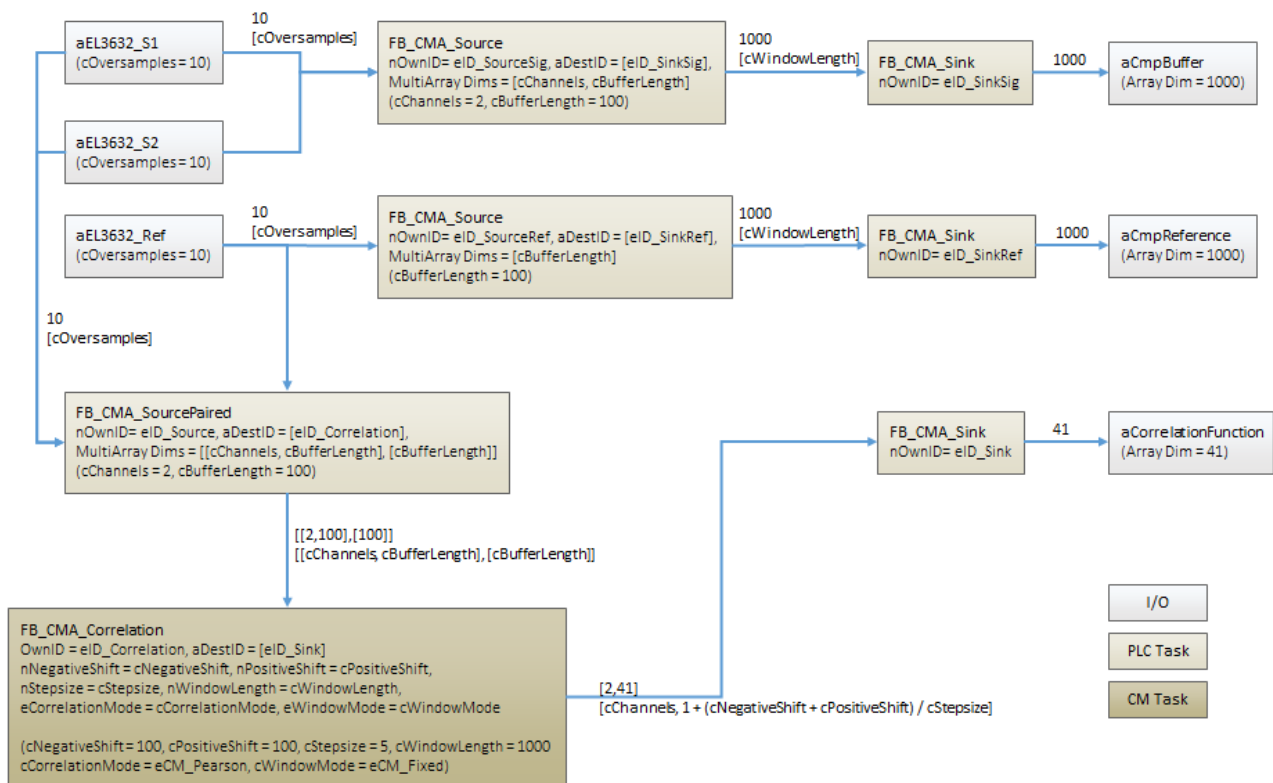
開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM, Tc3_CM_Base

6.26 関連ファンクション

このサンプルでは、ファンクションブロック FB_CMA_Correlation の使用例を示します。ここでは、基準信号に対して ± 100 サンプルの時間オフセットを考慮して、相関係数が計算されます。

このサンプルのソースコードはこちらからダウンロードできます：https://infosys.beckhoff.com/content/1033/TF3600_TC3_ConditionMonitoring/Resources/9833229195.zip

ブロック図



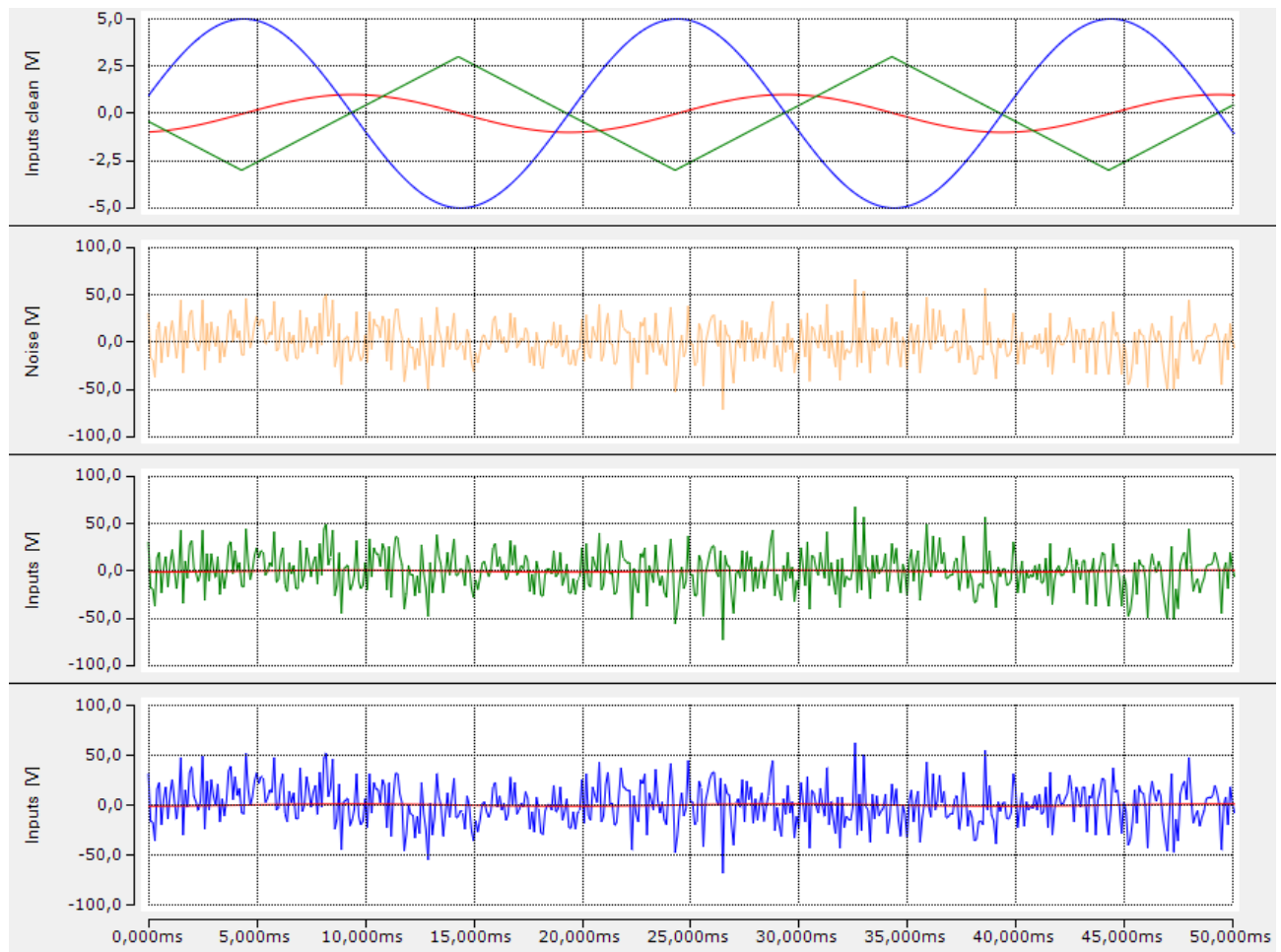
プログラムのパラメータ

以下の表は、ファンクションブロックを設定するための重要なパラメータの一覧です。

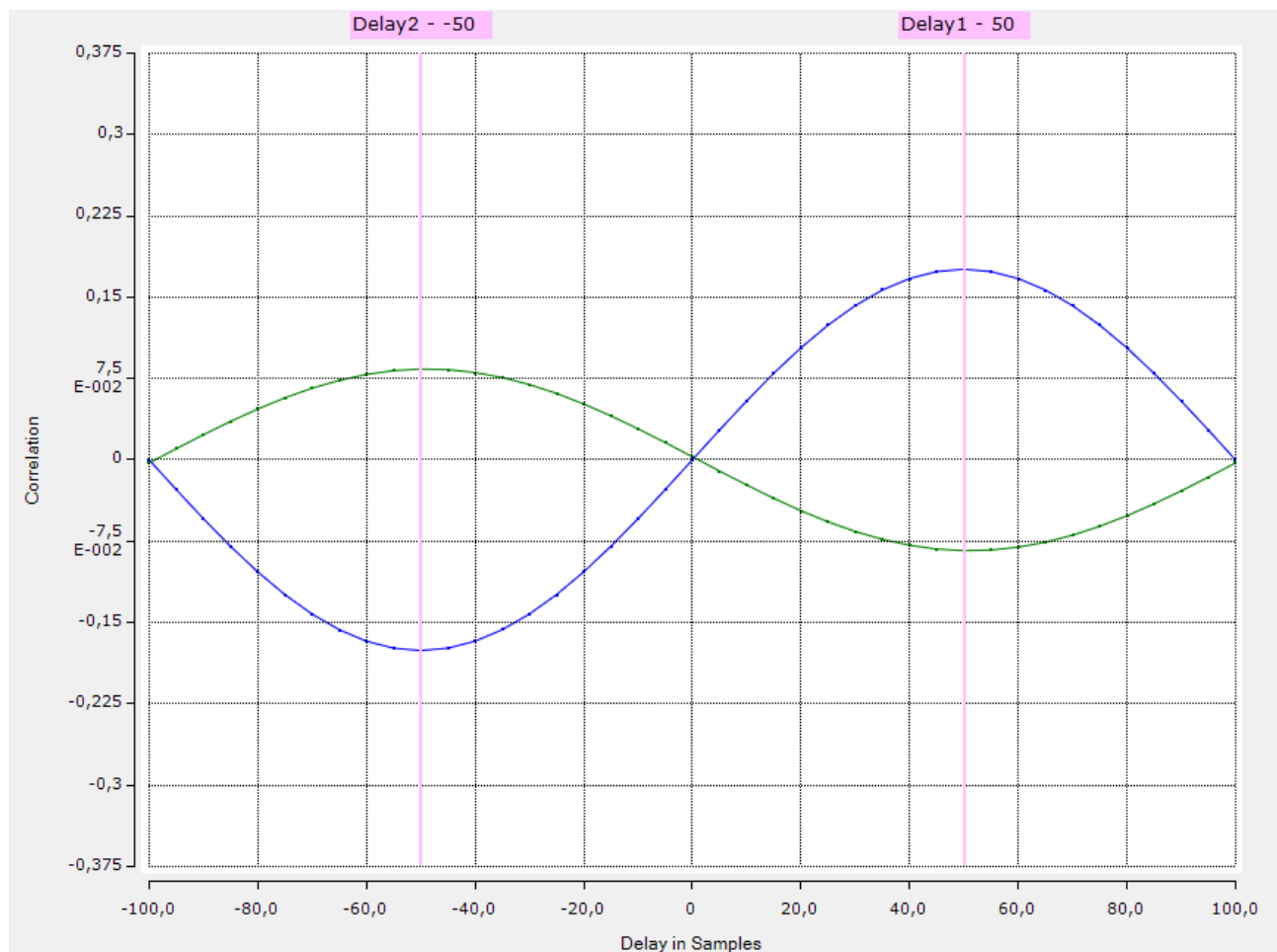
nWindowLength	1000
nNegativeShift	100
nPositiveShift	100
nStepsize	5
eCorrelationMode	eCM_Pearson
eWindowMode	eCM_Fixed

解説

このサンプルでは、振幅 1 の正弦波（赤）、+5 ms シフトした正弦波（10 kHz サンプリングレートで 50 サンプル）（青）、および同じ周波数（50 Hz）で -5 ms の遅延を持つ三角パルス（緑）の間の相関パラメータが計算されます。試験信号には通常のノイズが重畳されています。次の図は、出力信号（上）とノイズ信号（下）を示しています。



ノイズは、時間領域での出力が識別できないほど大きくなります。しかし、相関係数を計算することで、参照信号に対する分類が可能になります：



増幅された正弦波（青）の相関が最も高くなるのは、基準信号に対して 50 サンプルシフトしたとき、つまり信号が 5 ms 遅延しているときです。同様に、三角パルス（緑）は 50 サンプルシフトしており、基準信号に対して 5 ms 先行しています。

要件

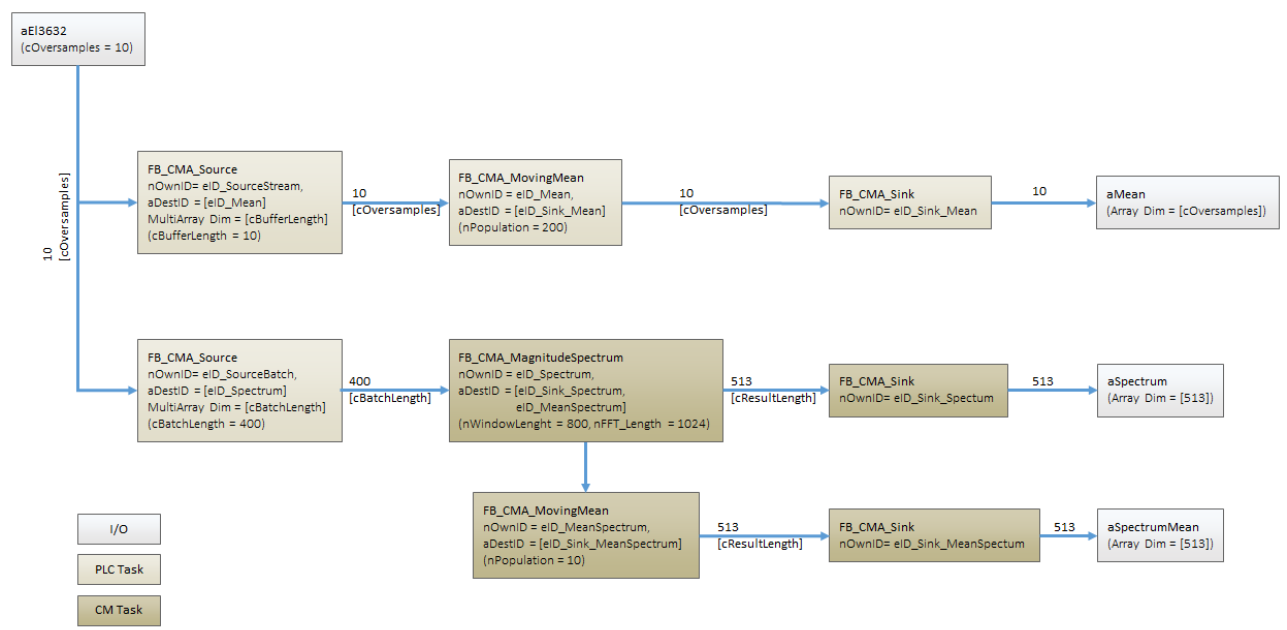
開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM, Tc3_CM_Base

6.27 移動平均

ここでは、移動平均の計算を例として、Condition Monitoringライブラリの使用方法を示します。ここでは [FB_CMA_MovingMean \[▶ 254\]](#) ブロックを使用します。同様に、移動標準偏差、歪度、過剰度も、[FB_CMA_MovingStandardDeviation \[▶ 258\]](#)、[FB_CMA_MovingSkew \[▶ 262\]](#)、[FB_CMA_MovingExcess \[▶ 266\]](#) ファンクションブロックを使用して計算できます。

サンプルのソースコードはこちらからダウンロードできます：https://infosys.beckhoff.com/content/1033/TF3600_TC3_ConditionMonitoring/Resources/16593242507.zip

ブロック図



プログラムのパラメータ

この表は、ファンクションブロックの設定に重要なパラメータの一覧です。

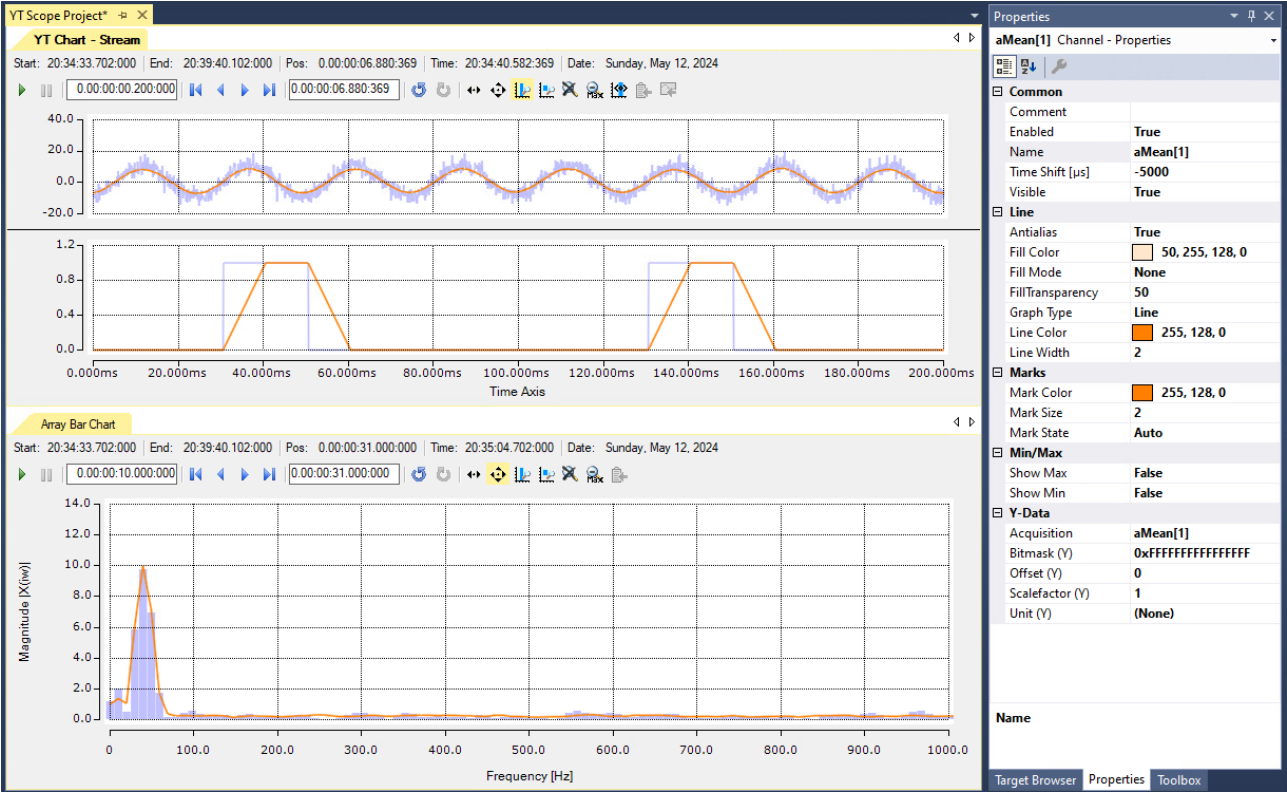
オーバーサンプル/バッファ長	10
個体数（ストリーム）	100
スタック長（スペクトル）	400
ウィンドウの長さ	800
FFTの長さ	1024
FFT結果の長さ	513

解説

このサンプルに含まれるスコーププロジェクトは、ストリーミングコンテキストにおいて、YT チャート上で計算された正弦波とパルスの移動平均を示しています。出力データが正しく処理されるようにするためには、時間が平均化にどのような影響を与えるかを考慮することが重要です。これについて以下に説明します。

- **正弦波**：計算された平均値は、時間オフセット-5000usでここに表示されます。この値は、1msのサイクルタイム（FastTaskを参照）とオーバーサンプリング係数を考慮し、内部バッファ長の半分（nPopulation）になります。
- **インパルス**：インパルス関数の平均化は、移動平均の慣性特性を示します。ここではパルス長を20msに設定します。（サイクルタイム、オーバーサンプリングの）設定にもよりますが、計算された移動平均は各パルスから10ms後に安定した値になります。

さらに、このプロジェクトには、計算された平均スペクトルを別の配列棒グラフに表示する機能が含まれています。これは、0～1kHz範囲のマグニチュードスペクトルを示しています。現在のスペクトルは棒グラフで表示され、移動平均は連続線で表示されます。



要件

開発環境	ターゲットプラットフォーム	必要なPLCライブラリ
TwinCAT v3.1.4022.25	PCまたはCX (x86、x64)	Tc3_CM, Tc3_CM_Base

7 付録

7.1 エラーコードの概要

エラーコードはHRESULT型で返されます。ゼロ以外の値のテストだけでは、HRESULT 型の値に対しては不十分です。

宣言	エラー	正常	正常（追加情報あり）	チェック関数
hrErrorCode :HRESULT ;	< 0	>= 0	> 0	SUCCEEDED(), FAILED()

以下のエラーコードが発生する可能性があります。

16#9811_0000 - 16#9811_ffff	TwinCAT (ADS) Error Codes [▶ 368]にリストされています（上位WORDを除く）。本ページの下部に注記があります。
16#9851_0000 - 16#9851_ffff	状態監視エラーコードは、E_CM_ErrorCode [▶ 275]にリストされています。
16#9852_0000 - 16#9852_0fff	状態監視解析のエラーコードは、E_CMA_ErrorCodes [▶ 278]にリストされています。
16#9871_0000 - 16#9871_ffff	マルチアレイエラーコードはE_MA_ErrorCode [▶ 279]にあります。



初期化中にエラーが発生した場合、ファンクションブロックは使用できません。

標準的なTwinCATエラーコードの詳細：

エラー値	シンボル（識別子）	エラー説明	対処法
16#9811_070A	NOMEMORY	メモリがありません。	メモリの設定に誤りがあります。 => ルータのメモリを増やしてください（メモリ管理 [▶ 68]の章を参照）。
16#9811_0719	TIMEOUT	デバイスのタイムアウトです。	バッファメモリ転送中にタイムアウトが発生した可能性があります。通常、これはCM解析チェーンでは重要ではありません。エラーへの対応は、アルゴリズムの種類とエラーが発生した正確な箇所により異なります。タイムアウト入力は、タスクのサイクルタイムと一致する場合のみ増加してください。 <u>並列処理</u> [▶ 72]の項を参照してください。

場合によっては、エラーコードによるエラー処理が最良の選択肢とは限りません。特に、標準的ではないが発生しうる入力データに対して、処理の結果が未定義の値になる場合はなおさらです。あるいは、値がプロセスから除外されている場合です。この場合、欠損値や部分的に未定義の結果は、特殊な定数NaNで記述することができます（NaN値の章 [▶ 71]を参照）。これは、プログラムのロジックではなく、特定の入力データに依存するエラーの場合に使用されます。

7.2 ADSリターンコード

エラーコードのグループ化:

グローバルエラーコード: ADSリターンコード [▶ 368]... (0x9811_0000 ...)

ルータのエラーコード: ADSリターンコード [▶ 368]... (0x9811_0500 ...)

ADS全般のエラー: ADSリターンコード [▶ 369]... (0x9811_0700 ...)

RTimeエラーコード: ADSリターンコード [▶ 370]... (0x9811_1000 ...)

グローバルエラーコード

16進数	10進数	HRESULT	名前	説明
0x0	0	0x98110000	ERR_NOERROR	エラーはありません。
0x1	1	0x98110001	ERR_INTERNAL	内部エラーです。
0x2	2	0x98110002	ERR_NORTIME	リアルタイムではありません。
0x3	3	0x98110003	ERR_ALLOCLOCKEDMEM	割り当てがロックされました - メモリエラーです。
0x4	4	0x98110004	ERR_INSERTMAILBOX	メールボックスがいっぱいです - ADSメッセージを送信できませんでした。サイクル当たりのADSメッセージ数を減らすと改善されます。
0x5	5	0x98110005	ERR_WRONGRECEIVEHMSG	HMSGに誤りがあります。
0x6	6	0x98110006	ERR_TARGETPORTNOTFOUND	ターゲットポートが見つかりません - ADSサーバが開始されていないか、ADSサーバに到達できない、あるいはADSサーバがインストールされていません。
0x7	7	0x98110007	ERR_TARGETMACHINENOTFOUND	ターゲットコンピュータが見つかりません - AMSルートが見つかりませんでした。
0x8	8	0x98110008	ERR_UNKNOWNCMDID	不明なコマンドIDです。
0x9	9	0x98110009	ERR_BADTASKID	タスクIDが無効です。
0xA	10	0x9811000A	ERR_NOIO	IOがありません。
0xB	11	0x9811000B	ERR_UNKNOWNAMSCMD	不明なAMSコマンドです。
0xC	12	0x9811000C	ERR_WIN32ERROR	Win32エラーです。
0xD	13	0x9811000D	ERR_PORTNOTCONNECTED	ポートが接続されていません。
0xE	14	0x9811000E	ERR_INVALIDAMSLENGTH	AMS長が無効です。
0xF	15	0x9811000F	ERR_INVALIDAMSNETID	AMS Net IDが無効です。
0x10	16	0x98110010	ERR_LOWINSTLEVEL	インストールレベルが低すぎます - TwinCAT 2ライセンスエラーです。
0x11	17	0x98110011	ERR_NODEBUGINTAVAILABLE	デバッグが利用できません。
0x12	18	0x98110012	ERR_PORTDISABLED	ポートが無効です - TwinCATシステムサービスが開始されていません。
0x13	19	0x98110013	ERR_PORTALREADYCONNECTED	ポートが接続済みです。
0x14	20	0x98110014	ERR_AMSSYNC_W32ERROR	AMS Sync Win32エラーです。
0x15	21	0x98110015	ERR_AMSSYNC_TIMEOUT	AMS Syncタイムアウトです。
0x16	22	0x98110016	ERR_AMSSYNC_AMSERROR	AMS同期エラーです。
0x17	23	0x98110017	ERR_AMSSYNC_NOINDEXINMAP	使用可能なAMS Sync用のインデックスマップがありません。
0x18	24	0x98110018	ERR_INVALIDAMSPORT	AMSポートが無効です。
0x19	25	0x98110019	ERR_NOMEMORY	メモリがありません。
0x1A	26	0x9811001A	ERR_TCPSEND	TCP送信エラーです。
0x1B	27	0x9811001B	ERR_HOSTUNREACHABLE	ホストに到達できません。
0x1C	28	0x9811001C	ERR_INVALIDAMSFRAGMENT	AMSフラグメントが無効です。
0x1D	29	0x9811001D	ERR_TLSEND	TLS送信エラー - セキュアなADS接続に失敗しました。
0x1E	30	0x9811001E	ERR_ACCESSDENIED	アクセス拒否 - セキュアなADSアクセスが拒否されました。

ルータのエラーコード

16進数	10進数	HRESULT	名前	説明
0x500	1280	0x98110500	ROUTERERR_NOLOCKEDMEMORY	ロックされたメモリは割り当てることができません。
0x501	1281	0x98110501	ROUTERERR_RESIZEMEMORY	ルータメモリのサイズを変更できませんでした。

16進数	10進数	HRESULT	名前	説明
0x502	1282	0x98110502	ROUTERERR_MAILBOXFULL	メールボックスが可能なメッセージの最大数に達しています。
0x503	1283	0x98110503	ROUTERERR_DEBUGBOXFULL	デバッグメールボックスが可能なメッセージの最大数に達しています。
0x504	1284	0x98110504	ROUTERERR_UNKNOWNPORTTYPE	ポートタイプが不明です。
0x505	1285	0x98110505	ROUTERERR_NOTINITIALIZED	ルータが初期化されていません。
0x506	1286	0x98110506	ROUTERERR_PORTALREADYINUSE	ポート番号が既に割り当てられています。
0x507	1287	0x98110507	ROUTERERR_NOTREGISTERED	ポートが登録されていません。
0x508	1288	0x98110508	ROUTERERR_NOMOREQUEUES	ポートの最大数に達しています。
0x509	1289	0x98110509	ROUTERERR_INVALIDPORT	ポートが無効です。
0x50A	1290	0x9811050A	ROUTERERR_NOTACTIVATED	ルータが有効ではありません。
0x50B	1291	0x9811050B	ROUTERERR_FRAGMENTBOXFULL	メールボックスが分割されたメッセージの最大数に達しています。
0x50C	1292	0x9811050C	ROUTERERR_FRAGMENTTIMEOUT	フラグメントタイムアウトが発生しました。
0x50D	1293	0x9811050D	ROUTERERR_TOBEREMOVED	ポートが削除されました。

一般的なADSエラーコード

16進数	10進数	HRESULT	名前	説明
0x700	1792	0x98110700	ADSERR_DEVICE_ERROR	一般的なデバイスエラーです。
0x701	1793	0x98110701	ADSERR_DEVICE_SRVNOTSUPP	サービスがサーバでサポートされていません。
0x702	1794	0x98110702	ADSERR_DEVICE_INVALIDGRP	インデックスグループが無効です。
0x703	1795	0x98110703	ADSERR_DEVICE_INVALIDOFFSET	インデックスオフセットが無効です。
0x704	1796	0x98110704	ADSERR_DEVICE_INVALIDACCESS	読み込み/書き込みが許可されていません。 複数の原因が考えられます。例えば、ルートを作成する際に間違ったパスワードが入力された場合などです。
0x705	1797	0x98110705	ADSERR_DEVICE_INVALIDSIZE	パラメータのサイズが正しくありません。
0x706	1798	0x98110706	ADSERR_DEVICE_INVALIDDATA	データ値が無効です。
0x707	1799	0x98110707	ADSERR_DEVICE_NOTREADY	デバイスの操作準備が完了していません。
0x708	1800	0x98110708	ADSERR_DEVICE_BUSY	デバイスがビジー状態です。
0x709	1801	0x98110709	ADSERR_DEVICE_INVALIDCONTEXT	オペレーティングシステムコンテキストが無効です。このエラーは、異なるタスクでADSブロックを使用すると発生することがあります。PLCでのマルチタスク同期によってこのエラーを解決できる場合があります。
0x70A	1802	0x9811070A	ADSERR_DEVICE_NOMEMORY	メモリが不足しています。
0x70B	1803	0x9811070B	ADSERR_DEVICE_INVALIDPARM	パラメータ値が無効です。
0x70C	1804	0x9811070C	ADSERR_DEVICE_NOTFOUND	見つかりません(ファイル...).
0x70D	1805	0x9811070D	ADSERR_DEVICE_SYNTAX	ファイルまたはコマンドの構文エラーです。
0x70E	1806	0x9811070E	ADSERR_DEVICE_INCOMPATIBLE	オブジェクトが一致しません。
0x70F	1807	0x9811070F	ADSERR_DEVICE_EXISTS	オブジェクトは既に存在しています。
0x710	1808	0x98110710	ADSERR_DEVICE_SYMBOLNOTFOUND	シンボルが見つかりません。
0x711	1809	0x98110711	ADSERR_DEVICE_SYMBOLVERSIONINVALID	シンボルのバージョンが無効です。これは、オンラインでの変更により発生する可能性があります。ハンドルを新規作成してください。
0x712	1810	0x98110712	ADSERR_DEVICE_INVALIDSTATE	デバイス(サーバ)が無効な状態です。
0x713	1811	0x98110713	ADSERR_DEVICE_TRANSMODENOTSUPP	AdsTransModeはサポートされていません。
0x714	1812	0x98110714	ADSERR_DEVICE_NOTIFYHNDINVALID	通知ハンドルが無効です。
0x715	1813	0x98110715	ADSERR_DEVICE_CLIENTUNKNOWN	通知クライアントが登録されていません。
0x716	1814	0x98110716	ADSERR_DEVICE_NOMOREHDLs	使用可能なハンドルがこれ以上ありません。
0x717	1815	0x98110717	ADSERR_DEVICE_INVALIDWATCHSIZE	通知サイズが大きすぎます。
0x718	1816	0x98110718	ADSERR_DEVICE_NOTINIT	デバイスが初期化されていません。
0x719	1817	0x98110719	ADSERR_DEVICE_TIMEOUT	デバイスのタイムアウトです。
0x71A	1818	0x9811071A	ADSERR_DEVICE_NOINTERFACE	インターフェースの確認に失敗しました。
0x71B	1819	0x9811071B	ADSERR_DEVICE_INVALIDINTERFACE	間違ったインターフェースがリクエストされました。
0x71C	1820	0x9811071C	ADSERR_DEVICE_INVALIDCLSID	クラスIDが無効です。
0x71D	1821	0x9811071D	ADSERR_DEVICE_INVALIDOBJID	オブジェクトIDが無効です。

16進数	10進数	HRESULT	名前	説明
0x71E	1822	0x9811071E	ADSERR_DEVICE_PENDING	リクエストが保留中です。
0x71F	1823	0x9811071F	ADSERR_DEVICE_ABORTED	リクエストが中止されました。
0x720	1824	0x98110720	ADSERR_DEVICE_WARNING	信号警告です。
0x721	1825	0x98110721	ADSERR_DEVICE_INVALIDARRAYIDX	配列インデックスが無効です。
0x722	1826	0x98110722	ADSERR_DEVICE_SYMBOLNOTACTIVE	シンボルが有効ではありません。
0x723	1827	0x98110723	ADSERR_DEVICE_ACCESSDENIED	アクセスが拒否されました。 複数の原因が考えられます。例えば、単方向ADSルートが逆方向に使用される場合があります。
0x724	1828	0x98110724	ADSERR_DEVICE_LICENSENOTFOUND	ライセンスが見つかりません。
0x725	1829	0x98110725	ADSERR_DEVICE_LICENSEEXPIRED	ライセンスの期限が切れています。
0x726	1830	0x98110726	ADSERR_DEVICE_LICENSEEXCEEDED	ライセンスを超えています。
0x727	1831	0x98110727	ADSERR_DEVICE_LICENSEINVALID	ライセンスが無効です。
0x728	1832	0x98110728	ADSERR_DEVICE_LICENSESYSTEMID	ライセンスの問題: システムIDが無効です。
0x729	1833	0x98110729	ADSERR_DEVICE_LICENSENOTIMELIMIT	ライセンスの期限がありません。
0x72A	1834	0x9811072A	ADSERR_DEVICE_LICENSEFUTUREISSUE	ライセンスの問題: 未来の時間です。
0x72B	1835	0x9811072B	ADSERR_DEVICE_LICENSETIMETOLONG	ライセンス期間が長すぎます。
0x72C	1836	0x9811072C	ADSERR_DEVICE_EXCEPTION	システム起動時の例外処理です。
0x72D	1837	0x9811072D	ADSERR_DEVICE_LICENSEDUPLICATED	ライセンスファイルを2回読み込みました。
0x72E	1838	0x9811072E	ADSERR_DEVICE_SIGNATUREINVALID	署名が無効です。
0x72F	1839	0x9811072F	ADSERR_DEVICE_CERTIFICATEINVALID	証明書が無効です。
0x730	1840	0x98110730	ADSERR_DEVICE_LICENSEOEMNOTFOUND	OEMからの不明な公開キーです。
0x731	1841	0x98110731	ADSERR_DEVICE_LICENSERESTRICTED	このシステムIDに対するライセンスが無効です。
0x732	1842	0x98110732	ADSERR_DEVICE_LICENSEDEMODENIED	デモライセンスは禁止されています。
0x733	1843	0x98110733	ADSERR_DEVICE_INVALIDFNCID	ファンクションIDが無効です。
0x734	1844	0x98110734	ADSERR_DEVICE_OUTOFRANGE	有効範囲外です。
0x735	1845	0x98110735	ADSERR_DEVICE_INVALIDALIGNMENT	アライメントが無効です。
0x736	1846	0x98110736	ADSERR_DEVICE_LICENSEPLATFORM	プラットフォームレベルが無効です。
0x737	1847	0x98110737	ADSERR_DEVICE_FORWARD_PL	コンテキスト - パッシブレベルに送信。
0x738	1848	0x98110738	ADSERR_DEVICE_FORWARD_DL	コンテキスト - ディスパッチレベルに送信。
0x739	1849	0x98110739	ADSERR_DEVICE_FORWARD_RT	コンテキスト - リアルタイムに送信。
0x740	1856	0x98110740	ADSERR_CLIENT_ERROR	クライアントエラーです。
0x741	1857	0x98110741	ADSERR_CLIENT_INVALIDPARAM	サービスに無効なパラメータが含まれています。
0x742	1858	0x98110742	ADSERR_CLIENT_LISTEMPTY	ポーリングリストが空です。
0x743	1859	0x98110743	ADSERR_CLIENT_VARUSED	var接続が既に使用されています。
0x744	1860	0x98110744	ADSERR_CLIENT_DUPLINVOKEID	呼び出されたIDは既に使用されています。
0x745	1861	0x98110745	ADSERR_CLIENT_SYNC TIMEOUT	タイムアウトが発生しました - 指定されたADSタイムアウトでリモートターミナルが応答しません。リモートターミナルのルート設定が間違っている可能性があります。
0x746	1862	0x98110746	ADSERR_CLIENT_W32ERROR	Win32サブシステムのエラーです。
0x747	1863	0x98110747	ADSERR_CLIENT_TIMEOUTINVALID	クライアントタイムアウト値が無効です。
0x748	1864	0x98110748	ADSERR_CLIENT_PORTNOTOPEN	ポートが開いていません。
0x749	1865	0x98110749	ADSERR_CLIENT_NOAMSADDR	AMSアドレスがありません。
0x750	1872	0x98110750	ADSERR_CLIENT_SYNCINTERNAL	Ads syncの内部エラーです。
0x751	1873	0x98110751	ADSERR_CLIENT_ADDHASH	ハッシュテーブルがオーバーフローしました。
0x752	1874	0x98110752	ADSERR_CLIENT_REMOVEHASH	テーブルにキーが見つかりません。
0x753	1875	0x98110753	ADSERR_CLIENT_NOMORESVM	キャッシュにシンボルがありません。
0x754	1876	0x98110754	ADSERR_CLIENT_SYNCRESINVALID	無効な応答を受信しました。
0x755	1877	0x98110755	ADSERR_CLIENT_SYNCPORTLOCKED	Syncポートがロックされています。
0x756	1878	0x98110756	adserr_client_requestcancelled	リクエストはキャンセルされました。

RTimeエラーコード

16進数	10進数	HRESULT	名前	説明
0x1000	4096	0x98111000	RTERR_INTERNAL	リアルタイムシステムの内部エラーです。
0x1001	4097	0x98111001	RTERR_BADTIMERPERIODS	タイマー値が無効です。

16進数	10進数	HRESULT	名前	説明
0x1002	4098	0x98111002	RTERR_INVALIDTASKPTR	タスクポインタに無効な値0 (ゼロ)が設定されています。
0x1003	4099	0x98111003	RTERR_INVALIDSTACKPTR	スタックポインタに無効な値0 (ゼロ)が設定されています。
0x1004	4100	0x98111004	RTERR_PRIOEXISTS	リクエストタスクの優先順位は既に割り当て済みです。
0x1005	4101	0x98111005	RTERR_NOMORETCB	使用可能なTCB(タスク制御ブロック)はこれ以上ありません。TCBの最大数は64です。
0x1006	4102	0x98111006	RTERR_NOMORESEMAS	使用可能な排他処理はこれ以上ありません。排他処理の最大数は64です。
0x1007	4103	0x98111007	RTERR_NOMOREQUEUES	キューに使用可能な空きスペースがありません。キューの位置の最大数は64です。
0x100D	4109	0x9811100D	RTERR_EXTIRQALREADYDEF	外部同期割り込みが既に適用されています。
0x100E	4110	0x9811100E	RTERR_EXTIRQNOTDEF	外部同期割り込みは適用されていません。
0x100F	4111	0x9811100F	RTERR_EXTIRQINSTALLFAILED	外部同期割り込みの適用に失敗しました。
0x1010	4112	0x98111010	RTERR_IRQNOTLESSOREQUAL	間違ったコンテキストでサービス機能が呼び出されました。
0x1017	4119	0x98111017	RTERR_VMXNOTSUPPORTED	Intel VT-x拡張はサポートされていません。
0x1018	4120	0x98111018	RTERR_VMXDISABLED	Intel VT-x拡張はBIOSでは有効ではありません。
0x1019	4121	0x98111019	RTERR_VMXCONTROLSMISSING	Intel VT-x拡張でファンクションが見つかりません。
0x101A	4122	0x9811101A	RTERR_VMXENABLEFAILS	Intel VT-xの有効化に失敗しました。

特定の正のHRESULTリターンコード：

HRESULT	名前	説明
0x0000_0000	S_OK	エラーはありません。
0x0000_0001	S_FALSE	エラーはありません。 例：処理は成功しましたが、結果が負または不完全です。
0x0000_0203	S_PENDING	エラーはありません。 例：処理は成功しましたが、まだ結果が出ていません。
0x0000_0256	S_WATCHDOG_TIMEOUT	例：処理は成功しましたが、タイムアウトが発生しました。

TCP Winsockのエラーコード

16進数	10進数	名前	説明
0x274C	10060	WSAETIMEDOUT	接続タイムアウトが発生しました - 接続確立中のエラーです。一定時間が経過してもリモートターミナルが応答しなかった、または接続されたホストが応答しなかったために確立された接続を維持できなかったことが原因です。
0x274D	10061	WSAECONNREFUSED	接続が拒否されました - ターゲットコンピュータが接続を拒否したため、接続を確立できませんでした。通常、このエラーは外部ホスト上で無効なサービス(サーバアプリケーションが実行されていないサービス)への接続を試行すると発生します。
0x2751	10065	WSAHOSTUNREACH	ホストへの経路がありません - ソケット操作が使用できないホストを参照しています。
その他のWinsockエラーコード：Win32エラーコード			

7.3 スペクトルスケーリングのオプション

本ページでは、スペクトル計算のスケーリングオプションの概要を説明します。以下の表は、スケーリングの記号と重要なパラメータを示しています。

シンボル (識別子)	ファンクションブロックパラメータ	意味
N	nFFT_Length	FFTの入力値数
F_s		サンプリングレート

シンボル（識別子）	ファンクションブロックパラメータ	意味
Σw_n	eWindowFunction、nWindowLength	ウィンドウ関数の値の合計
Σw_n^2	eWindowFunction、nWindowLength	ウィンドウ関数の2乗値の和
SQRT(x)		Xの平方根
MAX(X _n)		スペクトル値の最大値 X _n
RMS(x _n) = SQRT([Σ (x _n) ²] / N)		信号のRMS値
PSD(X _n)		パワースペクトルの密度
LSD(X _n)		線形スペクトルの密度
A		基準正弦信号の振幅

以下の表に、ファンクションブロックFB_CMA_PowerSpectrum [▶ 189]およびFB_CMA_MagnitudeSpectrum [▶ 161] と、これらから派生したファンクションブロックで選択できるデフォルトのスケーリングオプション（E_CM_ScalingType [▶ 273]）を示します。結果の係数は、ユーザが評価する必要はありません。必要に応じてその他のパラメータを追加できるように、2列目に記載されています。値 x_n は、ファンクションブロックの入力値を表し、値 X_k はスケーリングの結果得られる周波数チャンネル k のスペクトル値を表します。

スケーリングオプション	含有係数	説明
決定論的信号		
eCM_PeakAmplitude	$2 / \Sigma w_n$	このスケーリングは、振幅 A の入力正弦信号が最大値 A に達するように、振幅値を調整します。結果は、ウィンドウ関数の種類に依存しません。マグニチュード値の単位は、入力信号の単位と同じです。 $MAX(X_k) = A$ ただし、いわゆるスカロッピング損失が発生する可能性があるため、スペクトルの最大値だけでは振幅を安定して推定することはできません。
eCM_RootPowerSum	$2 / \text{SQRT}(N * \Sigma w_n^2)$	このスケーリングは、振幅 A の入力正弦信号に対して、パワー値の総和の2乗が A になるようにスペクトル値を調整します。したがって、マグニチュード値の2乗の総和の平方根を使用することもできます。その結果、入力信号のRMS値にSQRT(2)を掛けた値と等しくなります。 $\text{SQRT}(\Sigma X_k ^2) = A$ このスケーリングは、狭帯域信号の評価に適しています。隣接する周波数帯域を通じた加算処理は、スカロッピング損失を低減するため、eCM_PeakAmplitudeの方法よりもかなり安定しています。
eCM_RMS	$\text{SQRT}(2/N * \Sigma w_n^2)$	このスケーリングによりパワー値が得られ、その総和の平方根が入力信号の RMS 値と等しくなります。振幅Aの正弦波信号は、 $A/\text{SQRT}(2)$ の値となります： $\text{SQRT}(\Sigma X(k) ^2) = \text{RMS}(x_n) = A * \text{SQRT}(1/2)$ eCM_ROOT_POWER_SUM と同様、このスケーリングも安定しており、狭帯域信号の評価に適しています。さらに、RMS値は広帯域信号に対しても明確に定義されています。
確率信号と広帯域信号		

スケーリングオプション	含有係数	説明
eCM_PowerSpectralDensity	$\text{SQRT}(2 / \Sigma w_n^2)$	このスケーリングによって、パワースペクトル密度 (PSD) が決定されます。広帯域で確率的な信号の場合、これはFFTとウィンドウ関数のパラメータに依存しません。 $\text{PSD}(X_k) = X_k ^2 / F_s$ 物理的に正しいパワースペクトル密度を求めるには、結果を入力信号のサンプリングレート (Hz) で割る必要があります。入力信号の単位がボルト (V) の場合、振幅の単位は 1 V/Hz、電力密度の単位は 1 V²/Hz となります。線形スペクトル密度はサンプリングレートの平方根で除算する必要があり、単位は 1V/(1Hz)^{1/2}となります： $\text{LSD}(X_k) = X_k / \text{SQRT}(F_s)$
基本		
eCM_DiracScaling	$\text{sqrt}(N / \Sigma w_n^2)$	このスケーリングは、広帯域信号がスケーリングされていないFFT（上記の定義による）と等しくなるようにパワースペクトルを正規化します。そのため、ウィンドウの種類や長さによる影響は排除されます。しかし、FFT長Nの影響は、スケーリングされていないFFTと同様に存在します。
eCM_NoScaling	1	スケーリングはありません。結果は、ウィンドウ関数（慣例にしたがって常に最大値が1）の適用と、それに続くFFTで構成されます。

7.4 サポートとサービス

世界中のベッコフ支社と代理店は、包括的なサポートとサービスを提供し、ベッコフ製品とシステムソリューションに関するあらゆる質問に対して迅速かつ的確なサポートを提供しています。

ダウンロード検索

ダウンロード検索 から当社が提供する各種ファイルをダウンロードいただけます。アプリケーションレポート、技術マニュアル、図面、Configurationファイルなど、必要なファイルを検索してダウンロードできます。

様々なファイル形式でダウンロードできます。

ベッコフの支社と代理店

ベッコフ製品に関するローカルサポートおよびサービスについては、最寄りのベッコフ支社または代理店にお問い合わせください。

各国のベッコフ支社および代理店の所在はベッコフWebサイト(<http://www.beckhoff.com/ja-jp>)よりご確認ください。

また、Webサイトではベッコフ製品マニュアルも公開されています。

ベッコフのサポート

ベッコフのサポート部門はベッコフ製品に関するお問い合わせの他、各種の技術サポートを提供しています。

- サポート
- 複雑な自動化システムの設計、プログラミングおよびコミショニング
- およびベッコフのシステムコンポーネントに関する広範なトレーニングプログラム

ホットライン: +49 5246 963-157

Eメール: support@beckhoff.co.jp

ベッコフのサービス

ベッコフのサービスセンターは、各種のアフターサービスを提供することでお客様をサポートします。

- オンサイトサービス
- 修理サービス
- 部品交換サービス
- 緊急サービス

ホットライン: +49 5246 963-460

Eメール: service@beckhoff.co.jp

ベッコフ本社

Beckhoff Automation GmbH & Co. KG

Huelshorstweg 20
33415 Verl
Germany

電話: +49 5246 963-0

Eメール: info@beckhoff.com

Webサイト: www.beckhoff.com

用語集

FFT

高速フーリエ変換：離散フーリエ変換の計算方法。厳密に言えば、このような計算方法はいくつか存在し、一般的な実装では入力長として2の累乗のみを許可している（Cooley-Tukeyアルゴリズム）。共通の特徴は、 $O(n \cdot \log(n))$ のオーダーの複雑度である。つまり、2048ポイントのFFTの計算は、512ポイントの場合の4倍強の複雑さである。

NaN (非数)

IEEE 754規格にしたがって、無効な値や欠損値をマークする記号定数。NaN値の主な特徴として、以下の点が挙げられる：NaNを入力データとする全ての算術演算は、結果としてNaNを返す。少なくともオペランドの1つがNaNの場合、全ての関係演算子 $=$ 、 $<$ 、 $>$ 、 $<=$ 、 $>=$ は、必ず値Falseを返す。標準関数 `isnan` または `_isnan` は、引数が NaN の場合、Trueを返す。式 `isnan(a)` は、式 `!(a == a)` または `NOT(a = a)` に相当。さらに計算で使用する場合、NaN値が自身を再生成するという事実は、無効な値が見過ごされることができないという点で有利。

RCFAまたは根本原因故障解析

損傷の主原因を特定するための解析に使用される名称。ローラーベアリングの場合、一次損傷は結果的に複雑な損傷につながるため、これは特に重要である。原因を特定することで、被害の発生を効果的に抑制できる。

アーティファクト

例えばエイリアシングなど、処理のエラーに起因する信号の不要な変化。

ウィンドウ関数

この関数は、例えば、人工的なジャンプを加えることなく長い入力信号を分解するために、周波数解析（ウェルチ法）と組み合わせで使用される。標準では、ほとんどのケースでHannウィンドウを使用することができる。ウィンドウ関数の選択は、周波数解析の周波数分解能と時間分解能に影響する。

ウィンドウ処理

ウィンドウ関数（上記参照）による乗算の計算手順に付与された名称。

エイリアシング

信号中にサンプリングレートの半分より高い周波数が発生した場合に発生するエラー。この場合、サンプリング信号は正確に再構成で

きない（ナイキストの定理）。これらの周波数は、いわゆるイメージ周波数としてスペクトルに反映される。

オーバーラップ加算法

信号の情報を失うことなく、最初に短期スペクトルに分解し、次に周波数領域でさらに処理（フィルタ処理など）し、再び連続時間信号として再構成する方法。

クレストファクタ

信号のピーク値とRMS値の関係で、通常はデシベルで表される。

ケプストラム

周波数解析に基づく変換であり、高調波や振幅変調によって生じるスペクトル内の周期的要素を強調する。パワーケプストラムとコンプレックスケプストラムは区別される。

ケフレンシー

ケプストラムの計算結果から得られる時間軸に与えられた名称。「周波数」という言葉をシャッフルして反転させたようなこの名称は、ケプストラムの特徴である「反転」と「再ソート」の操作を示唆している。連続する2回のフーリエ変換の結果、最初は周波数領域への変換が行われ、単位は1ヘルツ（Hz）となる。しかし、2番目の変換によって得られる時間領域では、軸上にあるのは絶対時間ではなく、ケプストラムによって決定される周期的な持続時間となる。ケフレンシーの単位は秒。

サンプリング周波数

アナログ信号が最初にサンプリングされ、デジタル値に変換される際の周波数。この変換は、サンプリング周期と呼ばれる一定時間の間隔で行われる。サンプリング周期の逆数をサンプリング周波数と呼び、単位はヘルツ（Hz）。「ナイキストの定理」も参照。

スカロッピング

狭帯域信号（例えば正弦信号や校正器の信号）の正確なスペクトル値が、FFTチャンネルのどの部分にチャンネル周波数があるかによって異なるという効果である。効果の程度はウィンドウ関数に依存する。

ゼロパディング

より少ないサンプル数から一定の長さのFFTを計算する場合に適用される処理ステップ。そのため、時系列データの値の前後をゼロで埋めて、目的の値数に達するようにする。これ

には通常、時系列に誤ったジャンプが生じないように、例えばウェルチ法にしたがって信号をウィンドウ処理する必要がある。ゼロパディングはFFTの周波数分解能を向上させるが、これはサンプリングレートをFFTポイント数で割った値に等しく、元の信号の情報量は当然増加しない。

デシベルまたはdB

振動の強度または強度比を評価するための対数スケール。デシベル（記号dB）は、補助計測単位Belの10分の1として定義される。xが累乗値であれば、デシベル値yは $20 * \log_{10}(x/x_0)$ となる。x0には、値1または定義された基準値が使用される。

ナイキストの定理またはサンプリングの定理

通信技術や信号処理に関する定理で、簡単に言うと、連続信号はその信号に含まれる最も高い周波数の2倍以上の周波数でサンプリングする必要があるという定理。これにより、得られた時間離散信号から、情報の損失や曖昧さを生じることなく元の信号を再構成できる。この最大周波数はナイキスト周波数と呼ばれる。実際には、ほとんどのD/Aコンバータには、入力信号の最大周波数をサンプリングレートの半分より小さい値に制限するフィルタが組み込まれている。

ヒルベルト変換

発振信号から90度位相シフトした信号を効率よく求める変換。ヒルベルト変換は、解析信号の計算などに使用される。

ビン

マルチチャンネル信号出力の1チャンネルを示す。この呼称は、FFTやヒストグラムの形成など、信号を変換する変換で特に使用される。

フーリエ変換

時間信号を異なる周波数部分に分解できる変換であり、多くの周波数解析手法の基礎となる。無限信号の連続関数を表す連続フーリエ変換の代わりに、通常、実務では離散的かつ周期的な信号に対して定義される離散フーリエ変換（DFT）が使用される。実用上重要な離散フーリエ変換の効率的な実装として、高速フーリエ変換（FFT）がある。

ベッセル補正

一連のデータから統計的モーメント係数を推定する際に、自由度の数を考慮する補正。具体的には、例えば標準偏差は $\sqrt{n/(n-1)}$ 、歪度は $\sqrt{n*(n-1)/(n-2)}$ などの係数を乗じて補正する。一般に、nが大きな数であれば、この係数は無視できる。

モーメント係数

統計変数の平均値、標準偏差、歪度、尖度などの統計値の総称。変数の確率分布、またはヒストグラムの中心統計モーメントから計算できるため、このように呼ばれている。

円形エイリアシング

これは、周波数領域で信号が変更され、逆FFT（オーバーラップ加算法）によって時間領域に変換された際に発生する可能性のあるアーティファクト。この修正は、周波数領域での乗算として記述することができ、一般的に時間領域でのフィルタリングに相当する。これは、フィルタリングのインパルス応答によって周期的に定義されるフォールディングに相当するインパルス応答が長すぎると、時間帯の最初に属する信号部分が区間の最後に現れ、逆もまた同様である。理由は、離散フーリエ変換の周期的定義にある。そのため、周波数領域での広範囲な修正は、アーチファクトを引き起こす可能性がある。対策として、時間信号を処理する前にゼロで補い（ゼロパディング）、信号の拡張のための予備を作成することができる。

加速度スペクトル密度（ASD）

入力信号が、例えば圧電振動ピックアップで計測される加速度信号である場合、フーリエ変換の出力値で表される物理変数に与えられる名称。周波数間隔で積分した場合、加速度密度はパワー密度とほぼ同じ方法で周波数固有の加速度を生成する。通常の単位は、1ミリメートル毎秒2乗毎ヘルツ = 1 mm/s²/Hz。

噛み合い周波数

またはメッシング周波数は、ギアボックスの歯の対が互いに接触する周波数を示す。この接触がいわゆる噛み合い振動を引き起こす。

機械保護

監視パラメータが臨界閾値を超えた場合に、可能な限り速やかにプラントのスイッチを自動的に切ることを目的とした手法に与えられる名称である。これにより事故や損傷を回避できる。

高調波

基本周波数の整数倍の振動。これらは振動の発生源におけるパルス型励起や非線形効果に特徴的であり、この場合、通常はスペクトル内の互いに一定間隔で並ぶ線群によって識別できる。

時間領域

計測後に得られる、時間的にサンプリングされた値を用いた信号の表現を指す。全ての信号のフーリエスペクトルは明確に表現でき、

損失なく等価な時間信号に変換し直すことができるため、時間領域と周波数領域は（関数空間におけるいわゆる「正規直交基底」として）同じ信号の等価な表現となる。

周波数領域

または周波数空間は、FFT値に基づいて信号を表現するために付与された名称。全ての信号の複素フーリエスペクトルは明確に表現でき、損失なく等価な時間信号に変換し直すことができるため、周波数領域と時間領域は（関数空間におけるいわゆる「正規直交基底」として）同じ信号の等価な表現となる。信号解析のための多くの操作は、時間領域よりも周波数領域の方がよりシンプルかつ効率的に実行できる。

接触角

ボールベアリングのボールが走行面に接触する線と、ベアリングの軸に垂直な平面との間の角度。ラジアル荷重専用設計されたベアリングでは、接触角は常にゼロに近いが、アキシャル荷重にも対応するベアリングでは、接触角が大きくなることがあります。したがって、この値は形状およびベアリングにかかる現在の荷重の両方に依存し、ピッチ径に起因する観察可能な損傷周波数に影響を及ぼす。したがって、アキシャル荷重用のベアリングの場合、これらは一定ではない。

尖度

（カートーシスまたは曲率とも呼ばれる）：値の統計的分布の「衝動性」または「ピーク性」を示す指標で、第4の統計的中心モーメントから決定される。分布を正確に評価するため、計測された分布の尖度と正規分布の尖度との間の距離（値は3）がしばしば使用される。これを過剰尖度と呼ぶ。ガウス分布の場合、過剰尖度はゼロになるが、外れ値の多い分布はゼロよりはるかに大きな値になる。

損傷周波数

特定の機械要素が損傷したときに発生する特徴的な周波数。例えば、ローラーベアリングの転動体、内輪、外輪、および保持器の損傷には特定の周波数が割り当てられており、これらの周波数は接触角に応じて軸の回転速度に比例する。

複雑度

この場合、アルゴリズムに必要なリソース（計算時間と、必要であればメモリ空間）の指定。Condition Monitoringファンクションは、データ量が大きく異なる状態で呼び出されることがある。例えば、短期FFTは32個の値だけで計算される場合があるが、一方で16000個の値を使ってケプストラムを計算することが有用な場合もある。したがって、入

力データ数が可変で n 個の場合、アルゴリズムがデータ量の増加に伴ってどのように振る舞うかを観察する。この挙動は、コンピュータサイエンスでは通常 $O(f(n))$ という表記（「ランダウ記号」とも呼ばれる）で表される。この表記法は、 n が増加しても、複雑度が関数 $f(n)$ を大きく上回る速さで成長しないことを示している。計算時間の複雑度 $O(n)$ を持つアルゴリズムは、例えば、8倍のデータ量 n に対して8倍の計算時間を必要とするが、複雑さ $O(n^2)$ を持つアルゴリズムは、すでに64倍の計算時間を必要とする。複雑度 $O(n * \log_2 n)$ のFFTは、逆に $n=16384$ では $n=256$ の112倍の計算時間を必要とする。データ量が少ない場合、計算時間は通常、入力データ数に依存しない部分に支配される。

分位点またはパーセンタイル

統計変数から決定される値の呼称である。まず経験度数分布（密度関数）を求め、そこから累積度数分布（累積分布関数ともいう）を算出する。パーセンタイル q の値は、確率変数が全体の q パーセントのケースで到達するものの、それを超えない最大値である。この値は、累積度数分布の逆関数を作成することで決定される。分位数とパーセンタイルの唯一の違いは、分位数が対応するパーセンテージの代わりに小数値を使用することです。50パーセンタイルの値は中央値とも呼ばれる。

歪度

統計分布の非対称性を計測するもので、統計的第3中心モーメントから決定される。対称分布は歪度がゼロである。

Trademark statements

Beckhoff®, TwinCAT®, TwinCAT/BSD®, TC/BSD®, EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, Safety over EtherCAT®, TwinSAFE®, XFC®, XTS® and XPlanar® are registered trademarks of and licensed by Beckhoff Automation GmbH.

Third-party trademark statements

DSP System Toolbox, Embedded Coder, MATLAB, MATLAB Coder, MATLAB Compiler, MathWorks, Predictive Maintenance Toolbox, Simscape, Simscape™ Multibody™, Simulink, Simulink Coder, Stateflow and ThingSpeak are registered trademarks of The MathWorks, Inc.

Intel, the Intel logo, Intel Core, Xeon, Intel Atom, Celeron and Pentium are trademarks of Intel Corporation or its subsidiaries.

詳細はこちら:
www.beckhoff.com/tf3600

Beckhoff Automation GmbH & Co. KG
Hülshorstweg 20
33415 Verl
Germany
+49 5246 9630
info@beckhoff.com
www.beckhoff.com

