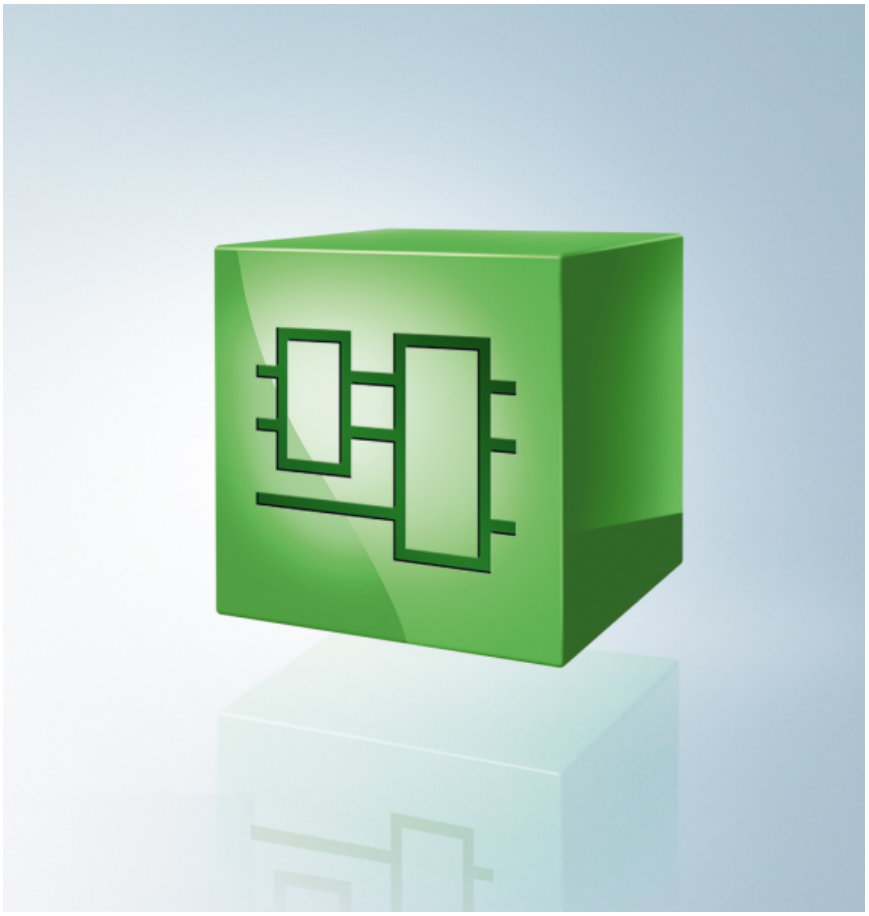


取扱説明書

PLCライブラリ: Tc2_System

TwinCAT 3

バージョン: 1.12
日付: 2020-04-30

BECKHOFF

目次

1 序文

1.1 取扱説明書に関する注記

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、オートメーションエンジニアリングの専門技術者のみの使用を対象としています。
コンポーネントのインストールとコミッショニングの際には、取扱説明書および以下の注意事項と説明に従うことが重要です。
技術者には各設置およびコミッショニングのそれぞれの時点で、発行された取扱説明書を使用する義務があります。

本製品を使用するうえでの責任者は、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、すで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。

商標

Beckhoff®、TwinCAT®、EtherCAT®、EtherCAT G®、EtherCAT G10®、EtherCAT P®、Safety over EtherCAT®、TwinSAFE®、XFC®、XTS®および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。

特許出願

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®は、Beckhoff Automation GmbH（ドイツ)によりライセンスを受けた登録商標および特許技術です。

著作権

1.2 安全に関する指示事項

安全に関する注意事項

この取扱説明書に記載された安全に関する指示や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。

納入仕様

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

この説明書は関連する国内法規を熟知した、制御およびオートメーションエンジニアリングの専門家の使用を目的としています。

安全記号の説明

この取扱説明書では、安全に関する指示や注意事項とともに以下の安全記号を使用します。安全に関する指示事項はよくお読みになり、必ず指示に従ってください。

⚠ 危険

重大な人的傷害の危険

この記号が付いた安全に関する注意事項に従わないと、人命および健康に直ちに危害を及ぼします。

⚠ 警告

人的傷害の危険

この記号が付いた安全に関する注意事項に従わないと、人命および健康に危険を及ぼします。

⚠ 注意

人的傷害の恐れ

この記号が付いた安全に関する注意事項に従わないと、人命および健康に危険を及ぼす恐れがあります。

注記

物的損害と環境汚染

この記号が付いた安全に関する注意事項に従わないと、物的損害と環境汚染をもたらす恐れがあります。

ヒントまたはアドバイス

この記号が示す情報により、さらに理解が深まります。

2 概要

PLCアプリケーションで必要なすべてのファンクションブロックおよびファンクションが、IEC61131-3で標準化されているわけではありません。Tc2_Systemライブラリには、IEC61131-3の標準化範囲に属さない、メーカ固有のTwinCATシステム用ファンクションおよびファンクションブロックが含まれています。

汎用ファンクションブロック

名前	説明
DRAND [▶ 13]	乱数ジェネレータです。
FB_IecCriticalSection [▶ 13]	重要なセクションは排他処理をする必要があります。
FB_SetLedColor_BAPI [▶ 15]	ユーザLEDからBIOS APIをサポートするPCおよび組込み型PCに切り替えます。
GETCURTASKINDEX [▶ 16]	現在のタスクのインデックスを判定します。

ADSファンクションブロック

名前	説明
ADSREAD [▶ 16]	データをADS経由で読み込みします。
ADSREADEX [▶ 18]	データをADS経由で読み込み、読み込んだデータのバイト数も返します。
ADSWRITE [▶ 19]	データをADS経由で書き込みします。
ADSRDWRT [▶ 20]	データをADS経由で読み込み、書き込みをします。
ADSRDWRTEX [▶ 22]	データをADS経由で書き込み、読み込んだデータのバイト数も返します。
ADSRDSTATE [▶ 23]	デバイスの状態をADS経由で読み込みします。
ADSWRTCTL [▶ 25]	コントロールワードをADS経由で書き込みします。
ADSRDDEVINFO [▶ 26]	デバイス情報をADS経由で読み込みします。

拡張ADSファンクションブロック

名前	説明
ADSREADIND [▶ 28]	ADSREAD指示
ADSWRITEIND [▶ 29]	ADSWRITE指示
ADSRDWRTIND [▶ 30]	ADSRDWRT指示
ADSREADRES [▶ 31]	ADSREADレスポンス
ADSWRITERES [▶ 32]	ADSWRITEレスポンス
ADSRDWRTRES [▶ 33]	ADSRDWRTレスポンス

データアクセス用ファンクションブロック

これらのファンクションブロックを使用して、PLCからのファイルをPCでローカルに処理することができます。TwinCATターゲットシステムは、AMSネットワークアドレスで識別します。このメカニズムにより、ネットワークの他のTwinCATシステム上でファイルを格納したり、編集したりすることが可能になります。ファイルへのアクセスは、3つの連続したフェーズで構成されます。

1. ファイルを開く。
2. 開いたファイルへの読み込み、書き込み処理
3. ファイルを閉じる。

ファイルを開くことで、名前が判別されいてる外部ファイルと実行中のプログラム間との一時的な接続を確立します。ファイルを閉じることで、処理の終了を指示し、定義された出力状態とします。これにより、他のプログラムでの処理が可能になります。

名前	説明
FB_EOF [▶ 34]	ファイル終端をチェック
FB_FileOpen [▶ 35]	ファイルを開く
FB_FileClose [▶ 37]	ファイルを閉じる
FB_FileGets [▶ 40]	ファイルから文字列を取得

名前	説明
FB_FilePuts [▶ 41]	ファイルに文字列を挿入
FB_FileRead [▶ 42]	ファイルから読み込み
FB_FileWrite [▶ 43]	ファイルへの書き込み
FB_FileSeek [▶ 45]	ファイルポインタを移動
FB_FileTell [▶ 46]	ファイルポインタの位置を取得
FB_FileDelete [▶ 47]	ファイルを削除
FB_FileRename [▶ 48]	ファイル名を変更
FB_CreateDir [▶ 49]	ディレクトリを新規作成
FB_RemoveDir [▶ 51]	ディレクトリを削除

イベントログ用ファンクションブロック

TwinCATイベントロガーには、TwinCATシステムに表示されるすべてのメッセージ(イベント)を管理するタスクがあり、発生したメッセージ(イベント)をTwinCATログファイルへ書き込みします。

名前	説明
ADSLOGEVENT [▶ 52]	TwinCATイベントログメッセージの送信およびその確認をします。
ADSCLEAREVENTS [▶ 55]	TwinCATイベントログメッセージの送信およびその確認をします。
FB_SimpleAdsLogEvent [▶ 55]	TwinCATイベントログメッセージの送信およびその確認をします。

IECステップ/SFCフラグ用ファンクションブロック

SFCプログラム/プロジェクトでIECステップまたはSFCフラグが使用されている場合は、これらのファンクション/ファンクションブロックが必要となります。

名前	説明
AnalyzeExpression [▶ 57]	SFCフラグ使用時に必要
AnalyzeEspressionTable [▶ 59]	SFCフラグ使用時に必要
AnalyzeExpressionCombined [▶ 62]	SFCフラグ使用時に必要
AppendErrorString [▶ 62]	SFCフラグ使用時、エラー説明での文字列フォーマットに必要
SFCActionControl [▶ 62]	IECステップの使用を有効化

ウォッチドッグ用ファンクションブロック

名前	説明
FB_PcWatchdog [▶ 63]	PCウォッチドッグの有効化/無効化 次のメインボードを実装したIPCでのみ使用可能: IP-4GVI63、CB1050、CB2050、CB3050、CB1051、CB2051、CB3051
FB_PcWatchdog_BAPI [▶ 64]	PCウォッチドッグの有効化/無効化 次のメインボードを実装したIPCでのみ使用可能: BIOSバージョンが0.44以降のCBxx63

タイムファンクションブロック

名前	説明
GETCPUCOUNTER [▶ 66]	CPUサイクルカウンタの読み込み
GETCPUACCOUNT [▶ 66]	PLCタスクサイクルカウンタの読み込み

汎用ファンクション

名前	説明
F_CheckMemoryArea [▶ 68]	指定されたサイズでリクエストされた変数が位置するメモリ領域に関する情報を返す

名前	説明
F_CmpLibVersion [▶ 69]	既存のライブラリとリクエストされたバージョンを比較
F_CreateIPv4Addr [▶ 69]	バイト型のIPv4アドレスを文字列に変換
F_ScanIPv4AddrIds [▶ 70]	文字列型のIPv4アドレスをバイトに変換
F_GetMappingPartner [▶ 71]	マッピング先のオブジェクトIDを返す
F_GetMappingStatus [▶ 71]	PLC変数の現在のマッピングステータスを返す
F_GetStructMemberAlignment [▶ 71]	使用されているメモリアライメントに関する情報の読み込み
F_SplitPathName [▶ 74]	パス名を4つの部分に分割
SETBIT32 [▶ 75]	DWORD内ビット指定でのセット
CSETBIT32 [▶ 76]	DWORD内ビット指定でのビット反転
GETBIT32 [▶ 77]	DWORD内ビット指定での読み込み
CLEARBIT32 [▶ 77]	DWORD内ビット指定でのリセット
GETCURTASTINDEXEX [▶ 78]	タスクインデックスを特定
LPTSIGNAL [▶ 78]	パラレルポートピンの1つに信号を出力
TestAndSet [▶ 79]	割り込みオプションなしのフラグをセットおよびチェック

ADS用ファンクション

以下に記載するファンクションは、ADSインターフェイスを使用し、PLCコールからのWindows-NTオペレーティングシステムのいくつかの機能(メッセージボックスの出力など)へのアクセスを可能にします。

名前	説明
ADSLOGDINT [▶ 80]	DINT変数をNT Eventlog、Messagebox、またはその両方にロギング
ADSLOGLREAL [▶ 81]	(L)REAL変数をNT Eventlog、Messagebox、またはその両方にロギング
ADSLOGSTR [▶ 83]	文字列変数をNT Eventlog、Messagebox、またはその両方にロギング
F_CreateAmsNetId [▶ 84]	AmsNetId文字列を作成
F_ScanAmsNetIds [▶ 84]	AmsNetId文字列をアドレスバイトの配列に変換

文字ファンクション

名前	説明
F_ToASC [▶ 86]	文字をASCIIコードに変換
F_ToCHR [▶ 85]	ASCIIコードを文字に変換

I/Oポートアクセス

名前	説明
F_IOPortRead [▶ 86]	I/Oポートから読み込み
F_IOPortWrite [▶ 87]	I/Oポートに書き込み

メモリファンクション

PLCランタイムシステムのメモリ領域への直接アクセスを可能にするファンクションです。

注記

システムクラッシュまたは禁止されたメモリ領域へのアクセス

これらのファンクションは物理メモリへの直接アクセスを可能にするため、適用する際には特に注意が必要です。パラメータ値を間違えると、システムクラッシュや禁止されたメモリ領域へのアクセスが発生する可能性があります。

名前	説明
MEMCMP [▶ 89]	2つのメモリ領域の変数値を比較
MEMCPY [▶ 90]	メモリ領域の変数値を他のメモリ領域にコピー

名前	説明
MEMMOVE [▶ 91]	重複するメモリ領域の値をコピー
MEMSET [▶ 92]	メモリ領域の変数を特定の値にセット

タイムファンクション

名前	説明
F_GetSystemTime [▶ 92]	オペレーティングシステムのタイムスタンプの読み込み
F_GetTaskTime [▶ 93]	タスクのターゲット開始時刻の読み込み

3 ファンクションブロック

3.1 汎用ファンクションブロック

3.1.1 DRAND

このファンクションブロックを使用すると、LREAL型の(疑似)乱数を生成できます。

VAR_INPUT

```
VAR_INPUT
    Seed : INT;
END_VAR
```

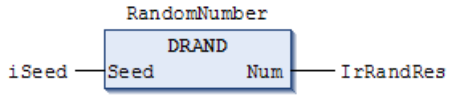
Seed: 乱数列を指定するための初期値です。

VAR_OUTPUT

```
VAR_OUTPUT
    Num : LREAL;
END_VAR
```

Num: この出力は、倍精度の0.0～1.0の範囲の疑似乱数を返します。この際、ジェネレータは期間あたり1075の確率値で数列を作成します。

FBDでのファンクションブロックの例:



この例では、LREAL値0.643412が生成され、返されます。入力パラメータSeedは、数列の初期値に影響を与えます。たとえば、異なるセッションで決定論的に再現可能な乱数列を生成する場合は、同一のSeed値を使用する必要があります。

要件

開発環境	ターゲットシステム	PLCライブラリ(カテゴリグループ)
TwinCAT v3.1.0	PCまたはCX (x86、x64、ARM)	Tc2_System (システム)

3.1.2 FB_IecCriticalSection

このファンクションブロックを使用して、クリティカルセクションを相互排他的に作成します。クリティカルセクションでは、1つまたは通常複数の変数に影響を与える変更が行われます。変更中は、状態に矛盾が発生します。このため、このような変更は一度に1つのタスクによってのみ実行する必要があります。これを可能にするために、このファンクションブロックにはメソッドEnter()およびLeave()が用意されています。Enter()を正常に呼び出すと、クリティカルセクションにアクセスできるようになり、このセクションが占有されているとみなされるようになります。変更が完了した際には、クリティカルセクションからLeave()で退出する必要があります。

⚠ 注意

タスクの停止によるサイクルタイムアウト

占有されているクリティカルセクションに対して、他のタスクがEnter()のコールによってアクセスを試行すると、TwinCATスケジューラがこれをブロックします。このタスクは、このセクションが再度有効になるまでブロックされます。有効になると、プログラムコードの処理が続行され、クリティカルセクションに進入します。

- 待機中のタスクでサイクルがオーバーランすることを回避するため、クリティカルセクションは短くしてください。複数のタスクがクリティカルセクションへの進入を待機している場合、優先度に基づいてアクセスが許可されます。

占有されているクリティカルエリアへの侵入を試行したためにタスクがTwinCATスケジューラによってブロックされている場合、これによって「busy waiting」は発生しません。このため、この間に優先度の低いタスクがCPUを使用します。

Windows CE

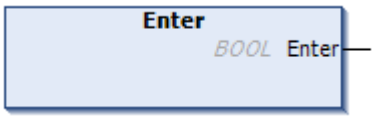
この機能は、Windows CEオペレーティングシステムのTwinCAT v3.1.4022.29以降でサポートされています。(古いバージョンのTwinCATでは、メソッドがFALSEを返します)。

代替方法

クリティカルセクションは、ファンクションTestAndSet() [▶ 79]によって実現することもできます。このファンクションを使用して、クリティカルセクションの内容を選択およびチェックできます。ただし、このファンクションにはブロック効果がなく、サイクルでセクションを処理できない可能性があります。

原則として、クリティカルセクションの数および長さは、可能な限り小さくする必要があります。

Enter() method



このメソッドは、クリティカルセクションの開始をマークします。

考えられる戻り値:

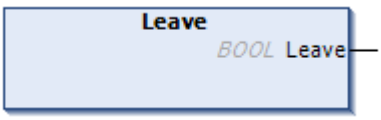
TRUE:

- クリティカルセクションに進入できます。

FALSE:

- クリティカルセクションに進入できません。
- このファンクションブロックは、ランタイムによってサポートされていません。
- クリティカルセクションが他のPLCタスクによって占有されています。このタスクはブレークポイント上で停止しています。戻り値FALSEの時は、タスクが常にブロックされることを回避し、I/Oの更新を維持してください。

Leave() method



このメソッドは、クリティカルセクションの終了をマークします。このメソッドは、クリティカルセクションの完了時に必ず呼び出す必要があります。

考えられる戻り値:

TRUE:

- セクションから正常に退出しました。

FALSE:

- このファンクションブロックは、ランタイムによってサポートされていません。
- クリティカルセクションがEnterで進入できませんでした。

ファンクションブロックの適用例:

ファンクションブロックFB_IecCriticalSectionにより、共有ファイルへのアクセスをセキュアにできます。ファンクションブロックのインスタンスおよびセキュアにするデータは、グローバルに作成されます。

```
VAR_GLOBAL
    fbCrititcalSection : FB_IecCriticalSection;
END_VAR

IF fbCrititcalSection.Enter() THEN
    (* start of critical section *)

    (* end of critical section *)
    fbCrititcalSection.Leave();
END_IF
```

要件

開発環境	ターゲットシステム	PLCライブラリ(カテゴリグループ)
TwinCAT v3.1.4020	PCまたはCX (x86、x64) WES、WES7、Win7、Win10	Tc2_System (システム)
TwinCAT v3.1.4022.29	PCまたはCX (x86、ARM) WinCE	Tc2_System (システム)

3.1.3 FB_SetLedColor_BAPI

ファンクションブロックFB_SetLedColor_BAPIを使用して、BIOS APIをサポートしているPCおよび組込み型PCのユーザLEDを切り替えることができます。eNewColorパラメータに従いLEDの色がbExecuteの立ち上がりに切り替わります。eNewColorパラメータはLEDをオフ(eNewColor = eUsrLED_Off)、または赤(eNewColor = eUsrLED_Red)、青(eNewColor = eUsrLED_Blue)、緑(eNewColor = eUsrLED_Green)に設定することが可能です。

VAR_INPUT

```
VAR_INPUT
    sNetID          : T_AmsNetID;
    eNewColor       : E_UsrLED_Color;
    bExecute        : BOOL;
    tTimeout        : TIME;
END_VAR
```

sNetID: デバイスのAMSネットワーク識別子(空文字列またはローカルネットワーク識別子) (T_AmsNetId [▶ 99]型)

eNewColor: 新しいLEDの色(E_UsrLED_Color [▶ 98]型)

bExecute: コマンドが立ち上がりで実行されます。

tTimeout: ADS通信が中止されるまでの時間です。

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy     : BOOL;
    bError    : BOOL;
    nErrID    : UDINT;
END_VAR
```

bBusy: ファンクションブロックがアクティブな間はTRUEです。

bError: コマンド実行中にエラーが発生するとTRUEとなります。

nErrID: 最後に実行されたコマンドのADSエラーコードまたはコマンド固有のエラーコードが表示されます。次のコマンド実行時に、0にリセットされます。

要件

開発環境	ターゲットシステム	PLCライブラリ(カテゴリグループ)
TwinCAT v3.1.0	PCまたはCX (x86、x64、ARM)	Tc2_System (システム) v3.4.14

3.1.4 GETCURTASKINDEX

ファンクションブロックGETCURTASKINDEXは、現在呼び出されているタスクのタスクインデックスを判定します。

現在のコールがリアルタイムコンテキストで発生しているのか、または周期PLCタスクからのコールなのかを判別する方法については、GETCURTASKINDEXEX [▶ 78]ファンクションの説明を参照してください。たとえば、初期化中のFB_initの自動コールは、周期PLCタスクからのコールではありません。

VAR_INPUT

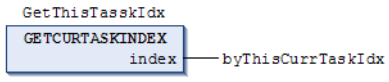
```
VAR_INPUT
(*none*)
END_VAR
```

VAR_OUTPUT

```
VAR_OUTPUT
    index : BYTE;
END_VAR
```

index: コールしているタスクの現在のタスクインデックス(1～4)を返します。

FBDでのブロックコールの例:

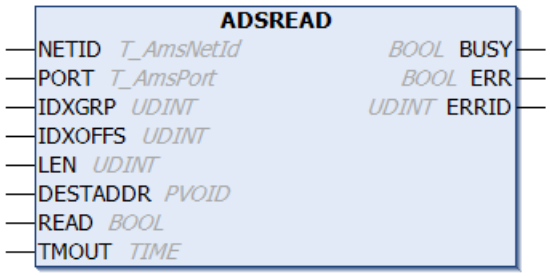


要件

開発環境	ターゲットシステム	PLCライブラリ(カテゴリグループ)
TwinCAT v3.1.0	PCまたはCX (x86、x64、ARM)	Tc2_System (システム)

3.2 ADSファンクションブロック

3.2.1 ADSREAD



ADSデバイスからデータを読み込むために、ファンクションブロックがADS-READコマンドを実行します。

VAR_INPUT

```
VAR_INPUT
    NETID      : T_AmsNetId;
    PORT       : T_AmsPort;
    IDXGRP     : UDINT;
    IDXOFFS    : UDINT;
    LEN        : UDINT;
    DESTADDR   : PVOID;
    READ       : BOOL;
    TMOUT      : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

NETIDADSコマンドの送信先であるターゲットデバイスのAMSネットワーク識別子の文字列です(型: T_AmsNetId [▶ 99]型)。

PORT: ADSデバイスのポート番号です(型: T_AmsPort [▶ 100]型)。

IDXGRP: リクエストするADSサービスのインデックスグループ番号(32ビット、符号なし)。この値は、アドレス指定されたデバイスのADSテーブルで確認できます。

IDXOFFS: リクエストするADSサービスのインデックスオフセット番号(32ビット、符号なし)。この値は、アドレス指定されたデバイスのADSテーブルで確認できます。

LEN: 読み込みするデータ長(バイト)。

DESTADDR: 読み込みしたデータを受信するバッファのアドレス。LENのバイト長を格納できるバッファサイズは、プログラマが決定する必要があります。単一の変数、配列、構造体をバッファにすることができ、そのアドレスはADR演算子で指定します。

READ: ADSコマンドは、この入力の立ち上がりで実行されます。

TMOUT: ファンクションがキャンセルされるまでの時間です。

VAR_OUTPUT

```
VAR_OUTPUT
    BUSY     : BOOL;
    ERR      : BOOL;
    ERRID    : UDINT;
END_VAR
```

BUSY: この出力は、ファンクションブロックがコマンドを実行するまではTRUEのままですが、Timeout入力の期間を過ぎるとFALSEになります。BUSY = TRUEの間は、入力で新しいコマンドが許可されません。これはサービスの実行ではなく、コマンド実行完了までの時間がモニタされることに注意してください。

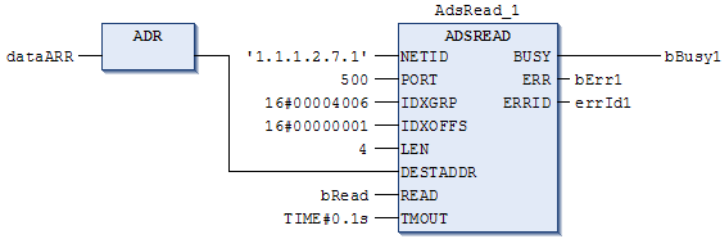
ERR: この出力は、コマンド実行中にエラーが発生すると直ちにTRUEに切り替わります。ERRIDには、コマンド固有のエラーコードが含まれています。ファンクションブロックにタイムアウトエラーが発生すると、ERRがTRUE、ERRIDが1861 (16進数: 0x745)となります。次のコマンド実行により、FALSEにリセットされます。

ERRID: 最後に実行されたコマンドのADSエラーコード [▶ 119]またはコマンド固有のエラーコードです。次のコマンド実行により、0にリセットされます。

STでのブロックコールの例:

- AdsReadファンクションブロックでの例 [▶ 111]

FBDでのブロックコールの例:

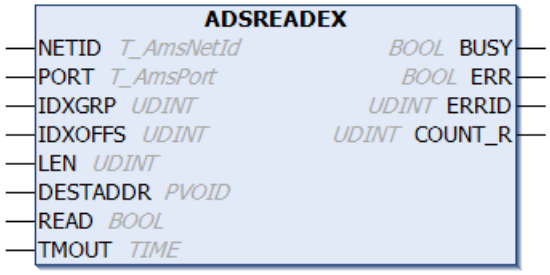


ここでは、軸番号6のエラーステータス(サイズが4バイトの要素)を問い合わさし、その値をdataArrバッファに書き込みます。IDXGRP 00004006 (hex)およびIDXOFFS 00000001 (hex)については、NC-ADSの説明を参照してください。

要件

開発環境	ターゲットシステム	PLCライブラリ(カテゴリグループ)
TwinCAT v3.1.0	PCまたはCX (x86、x64、ARM)	Tc2_System (システム)

3.2.2 ADSREADEX



ADSデバイスからデータを読み込みするために、ファンクションブロックがADS-READコマンドを実行します。このファンクションブロックには、ADSREADファンクションブロックと同一の機能の他に、実際に読み込みしたデータバイト数をパラメータとして返す機能があります。

VAR_INPUT

```
VAR_INPUT
    NETID    : T_AmsNetId;
    PORT     : T_AmsPort;
    IDXGRP   : UDINT;
    IDXOFFS  : UDINT;
    LEN      : UDINT;
    DESTADDR : PVOID;
    READ     : BOOL;
    TMOUT    : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

NETID: ADSコマンドの送信先であるターゲットデバイスのAMSネットワーク識別子の文字列です(型: T_AmsNetId [▶ 99]型)。

PORT: ADSデバイスのポート番号です(型: T_AmsPort [▶ 100]型)。

IDXGRP: リクエストするADSサービスのインデックスグループ番号(32ビット、符号なし)。この値は、アドレス指定されたデバイスのADSテーブルで確認できます。

IDXOFFS: リクエストするADSサービスのインデックスオフセット番号(32ビット、符号なし)。この値は、アドレス指定されたデバイスのADSテーブルで確認できます。

LEN: 読み込みするデータ長(バイト)。

DESTADDR: 読み込みしたデータを受信するバッファのアドレス。LENのバイト長を格納できるバッファサイズを、プログラム内で指定する必要があります。単一の変数、配列、構造体をバッファにすることができ、そのアドレスはADR演算子で指定します。

READ: ADSコマンドは、この入力の立ち上がりで実行されます。

TMOUT: ファンクションがキャンセルされるまでの時間です。

VAR_OUTPUT

```
VAR_OUTPUT
    BUSY    : BOOL;
    ERR     : BOOL;
    ERRID   : UDINT;
    COUNT_R : UDINT;
END_VAR
```

BUSY: この出力は、ファンクションブロックがコマンドを実行するまではTRUEのままですが、Timeout入力の期間を過ぎるとFALSEになります。BUSY = TRUEの間は、入力で新しいコマンドが許可されません。これはサービスの実行ではなく、コマンド実行完了までの時間がモニタされることに注意してください。

ERR: この出力は、コマンド実行中にエラーが発生すると直ちにTRUEに切り替わります。ERRIDには、コマンド固有のエラーコードが含まれています。ファンクションブロックにタイムアウトエラーが発生すると、ERRがTRUE、ERRIDが1861 (16進数: 0x745)となります。次のコマンド実行により、FALSEにリセットされます。

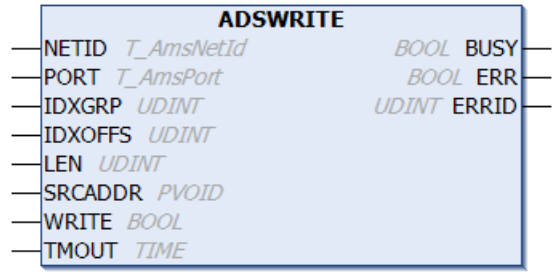
ERRID: 最後に実行されたコマンドのADSエラーコード [▶ 119]またはコマンド固有のエラーコードです。次のコマンド実行により、0にリセットされます。

COUNT_R: 正常に読み込みしたデータバイト数です。

要件

開発環境	ターゲットシステム	PLCライブラリ(カテゴリグループ)
TwinCAT v3.1.0	PCまたはCX (x86、x64、ARM)	Tc2_System (システム)

3.2.3 ADSWRITE



データをADSデバイスに送信するためのADS-WRITEコマンドを実行するブロックです。

VAR_INPUT

```
VAR_INPUT
    NETID   : T_AmsNetId;
    PORT    : T_AmsPort;
    IDXGRP  : UDINT;
    IDXOFFS : UDINT;
    LEN     : UDINT;
    SRCADDR : PVOID;
    WRITE   : BOOL;
    TMOUT   : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

NETID: ADSコマンドの送信先であるターゲットデバイスのAMSネットワーク識別子の文字列です(型: T_AmsNetId [▶ 99]型)。

PORT: ADSデバイスのポート番号です(型: T_AmsPort [▶ 100])。

IDXGRP: リクエストするADSサービスのインデックスグループ番号(32ビット、符号なし)。この値は、アドレス指定されたデバイスのADSテーブルで確認できます。

IDXOFFS: リクエストするADSサービスのインデックスオフセット番号(32ビット、符号なし)。この値は、アドレス指定されたデバイスのADSテーブルで確認できます。

LEN: 読み込みするデータ長(バイト)。

SRCADDR: 書込み用のデータが格納されているバッファのアドレス。LENのバイト長が取得できるバッファサイズを、プログラム内で指定する必要があります。単一の変数、配列、構造体をバッファにすることができ、そのアドレスはADR演算子で指定します。

WRITE: ADSコマンドは、この入力の立ち上がりで実行されます。

TMOUT: ファンクションがキャンセルされるまでの時間です。

VAR_OUTPUT

```
VAR_OUTPUT
    BUSY    : BOOL;
    ERR     : BOOL;
    ERRID   : UDINT;
END_VAR
```

BUSY: この出力は、ファンクションブロックがコマンドを実行するまではTRUEのままですが、Timeout入力の期間を過ぎるとFALSEになります。BUSY = TRUEの間は、入力で新しいコマンドが許可されません。これはサービスの実行ではなく、コマンド実行完了までの時間がモニタされることに注意してください。

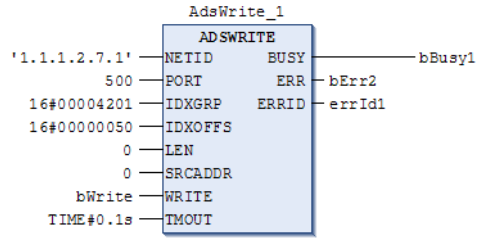
ERR: この出力は、コマンド実行中にエラーが発生すると直ちにTRUEに切り替わります。ERRIDには、コマンド固有のエラーコードが含まれています。ファンクションブロックにタイムアウトエラーが発生すると、ERRがTRUE、ERRIDが1861 (16進数: 0x745)となります。次のコマンド実行により、FALSEにリセットされます。

ERRID: 最後に実行されたコマンドのADSエラーコード [▶ 119]またはコマンド固有のエラーコードです。次のコマンド実行により、0にリセットされます。

STでのブロックコールの例:

- AdsWriteファンクションブロックでの例 [▶ 112]

FBDでのブロックコールの例

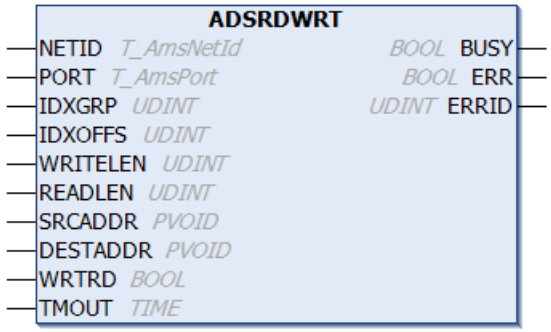


ここでは、NC軸1番がIDXGRP 00004201 (hex)およびIDXOFFS 00000050 (hex)の書き込み命令で無効化されています。この軸を有効化するには、IDXOFFS 00000051 (hex)の他の書き込み命令を実行する必要があります。この書き込み命令はパラメータを必要としないため、入力LENおよびSRCADDRが影響を与えることはありませんが、0に設定する必要があります。

要件

開発環境	ターゲットシステム	PLCライブラリ(カテゴリグループ)
TwinCAT v3.1.0	PCまたはCX (x86、x64、ARM)	Tc2_System (システム)

3.2.4 ADSRDWRT



このブロックにより、ADS-WRITE/READ命令を組み合わせて実行できます。データは一回のコールで、ADSデバイス(書き込み)およびそのレスポンスデータに送信されます。

VAR_INPUT

```
VAR_INPUT
    NETID     : T_AmsNetId;
    PORT      : T_AmsPort;
```

```
    IDXGRP    : UDINT;
    IDXOFFS   : UDINT;
    WRITELEN  : UDINT;
    READLEN   : UDINT;
    SRCADDR   : PVOID;
    DESTADDR  : PVOID;
    WRTRD     : BOOL;
    TMOUT     : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

NETID: ADSコマンドの送信先であるターゲットデバイスのAMSネットワーク識別子の文字列です(型: T_AmsNetId [▶ 99]型)。

PORT: ADSデバイスのポート番号です(型: T_AmsPort [▶ 100]型)。

IDXGRP: リクエストするADSサービスのインデックスグループ番号(32ビット、符号なし)。この値は、アドレス指定されたデバイスのADSテーブルで確認できます。

IDXOFFS: リクエストするADSサービスのインデックスオフセット番号(32ビット、符号なし)。この値は、アドレス指定されたデバイスのADSテーブルで確認できます。

WRITELEN: 書き込みするデータのバイト数。

READLEN: 読み込みするデータのバイト数。

SRCADDR: 書込み用のデータが格納されているバッファのアドレス。WRITELENのバイト長を格納できるバッファサイズは、プログラマが決定する必要があります。単一の変数、配列、構造体をバッファにすることができ、そのアドレスはADR演算子で指定します。

DESTADDR: 読み込みしたデータを受信するバッファのアドレス。READLENのバイト長を格納できるバッファサイズは、プログラマが決定する必要があります。単一の変数、配列、構造体をバッファにすることができ、そのアドレスはADR演算子で指定します。

WRTRD: ADSコマンドは、この入力の立ち上がりで実行されます。

TMOUT: ファンクションがキャンセルされるまでの時間です。

VAR_OUTPUT

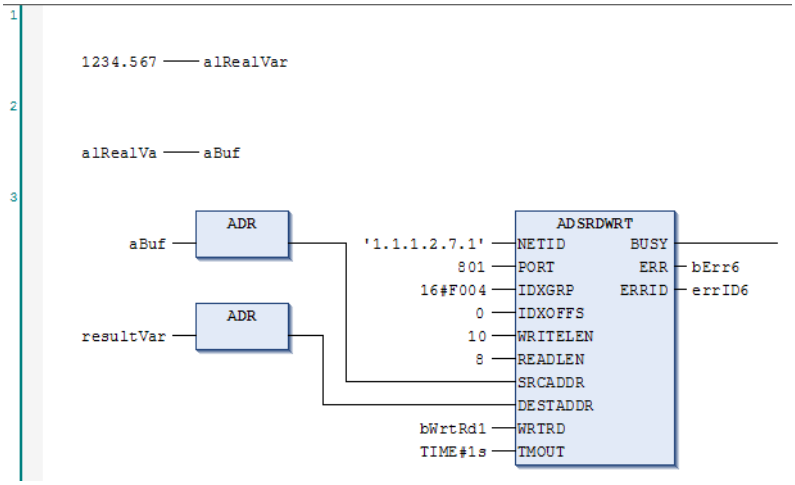
```
VAR_OUTPUT
    BUSY    : BOOL;
    ERR     : BOOL;
    ERRID   : UDINT;
END_VAR
```

BUSY: この出力は、ファンクションブロックがコマンドを実行するまではTRUEのままですが、Timeout入力の期間を過ぎるとFALSEになります。BUSY = TRUEの間は、入力で新しいコマンドが許可されません。これはサービスの実行ではなく、コマンド実行完了までの時間がモニタされることに注意してください。

ERR: この出力は、コマンド実行中にエラーが発生すると直ちにTRUEに切り替わります。ERRIDには、コマンド固有のエラーコードが含まれています。ファンクションブロックにタイムアウトエラーが発生すると、ERRがTRUE、ERRIDが1861 (16進数: 0x745)となります。次のコマンド実行により、FALSEにリセットされます。

ERRID: 最後に実行されたコマンドのADSエラーコード [▶ 119]またはコマンド固有のエラーコードです。次のコマンド実行により、0にリセットされます。

FBDでのブロックコールの例:

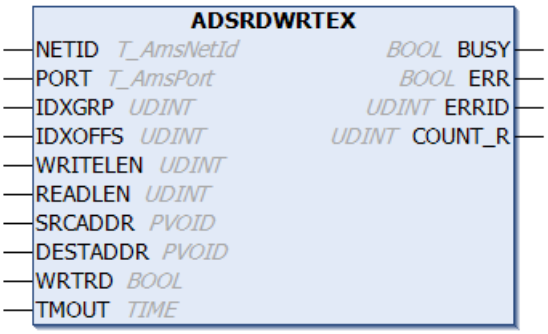


ここでは、「aLRealVar」という名前の変数の値が、Net-Id 1.1.1.2.7.1のコンピュータ上で動作しているPLCから読み込みします。このために、このコンピュータのアドレス、PLCの1番目のランタイムシステムのポート番号、インデックスグループ、および名前による変数読み込みのためのインデックスオフセット(F004 hex、0)が与えられます。変数名はバッファに格納され、PLCサーバに提供されます。この変数はグローバルであるため、先頭にドットが付加されます。これにより、書き込みするデータの長さは10文字(1つのドットと9つの文字)となります。読み込みする変数の型はLREALであるため、8バイト分読み込みされます。書き込みするデータのアドレスとして名前バッファのアドレスが与えられ、受信データのアドレスとしてLREAL変数(resultVar)のアドレスが与えられます。上図は、WriteRead命令実行後のフロー制御のブロックの状態を示しています。aLRealVarに含まれていた値1234.567が、resultVarにも含まれます。

要件

開発環境	ターゲットシステム	PLCライブラリ(カテゴリグループ)
TwinCAT v3.1.0	PCまたはCX (x86、x64、ARM)	Tc2_System (システム)

3.2.5 ADSRDWRTEX



このブロックにより、ADS-WRITE/READ命令を組み合わせて実行できます。データは一回のコールで、ADSデバイス(書き込み)およびそのレスポンスデータに送信されます。ADSRDWRTファンクションブロックとは異なり、ADSRDWRTEXは実際に読み込みするデータバイト数をパラメータとして提供します。

VAR_INPUT

```
VAR_INPUT
    NETID    : T_AmsNetId;
    PORT     : T_AmsPort;
    IDXGRP   : UDINT;
    IDXOFFS  : UDINT;
    WRITELEN : UDINT;
    READLEN  : UDINT;
    SRCADDR  : PVOID;
    DESTADDR : PVOID;
    WRTRD    : BOOL;
    TMOUT    : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

NETID: ADSコマンドの送信先であるターゲットデバイスのAMSネットワーク識別子の文字列です(型: T_AmsNetId [▶ 99]型)。

PORT: ADSデバイスのポート番号です(型: T_AmsPort [▶ 100]型)。

IDXGRP: リクエストするADSサービスのインデックスグループ番号(32ビット、符号なし)。この値は、アドレス指定されたデバイスのADSテーブルで確認できます。

IDXOFFS: リクエストするADSサービスのインデックスオフセット番号(32ビット、符号なし)。この値は、アドレス指定されたデバイスのADSテーブルで確認できます。

WRITELEN: 書き込みするデータのバイト数。

READLEN: 読み込みするデータのバイト数。

SRCADDR: 書込み用のデータが格納されているバッファのアドレス。WRITELENのバイト長を格納できるバッファサイズは、プログラマが決定する必要があります。WRITELENのバイト長を格納できるバッファサイズは、プログラマが決定する必要があります。

DESTADDR: 読み込みしたデータを受信するバッファのアドレス。READLENのバイト長を格納できるバッファサイズは、プログラマが決定する必要があります。WRITELENのバイト長を格納できるバッファサイズは、プログラマが決定する必要があります。

WRTRD: ADSコマンドは、この入力の立ち上がりで実行されます。

TMOUT: ファンクションがキャンセルされるまでの時間です。

VAR_OUTPUT

```
VAR_OUTPUT
    BUSY    : BOOL;
    ERR     : BOOL;
    ERRID   : UDINT;
    COUNT_R : UDINT;
END_VAR
```

BUSY: この出力は、ファンクションブロックがコマンドを実行するまではTRUEのままですが、Timeout入力の期間を過ぎるとFALSEになります。BUSY = TRUEの間は、入力で新しいコマンドが許可されません。これはサービスの実行ではなく、コマンド実行完了までの時間がモニタされることに注意してください。

ERR: この出力は、コマンド実行中にエラーが発生すると直ちにTRUEに切り替わります。ERRIDには、コマンド固有のエラーコードが含まれています。ファンクションブロックにタイムアウトエラーが発生すると、ERRがTRUE、ERRIDが1861 (16進数: 0x745)となります。次のコマンド実行により、FALSEにリセットされます。

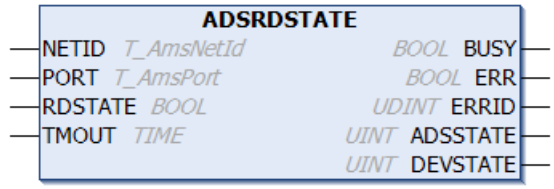
ERRID: 最後に実行されたコマンドのADSエラーコード [▶ 119]またはコマンド固有のエラーコードです。次のコマンド実行により、0にリセットされます。

COUNT_R: 正常に読み込みしたデータバイト数です。

要件

開発環境	ターゲットシステム	PLCライブラリ(カテゴリグループ)
TwinCAT v3.1.0	PCまたはCX (x86、x64、ARM)	Tc2_System (システム)

3.2.6 ADSRDSTATE



このブロックにより、ADSデバイスの状態をリクエストできます。

VAR_INPUT

```
VAR_INPUT
    NETID   : T_AmsNetId;
    PORT    : T_AmsPort;
    RDSTATE : BOOL;
    TMOUT   : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

NETID: ADSコマンドの送信先であるターゲットデバイスのAMSネットワーク識別子の文字列です(型: T_AmsNetId [▶ 99]型)。

PORT: ADSデバイスのポート番号です(型: T_AmsPort [▶ 100]型)。

RDSTATE: ADSコマンドは、この入力の立ち上がりで実行されます。

TMOUT: ファンクションがキャンセルされるまでの時間です。

VAR_OUTPUT

```
VAR_OUTPUT
    BUSY      : BOOL;
    ERR       : BOOL;
    ERRID     : UDINT;
    ADSSTATE  : UINT;
    DEVSTATE  : UINT;
END_VAR
```

BUSY: この出力は、ファンクションブロックがコマンドを実行するまではTRUEのままですが、Timeout入力の期間を過ぎるとFALSEになります。BUSY = TRUEの間は、入力で新しいコマンドが許可されません。これはサービスの実行ではなく、コマンド実行完了までの時間がモニタされることに注意してください。

ERR: この出力は、コマンド実行中にエラーが発生すると直ちにTRUEに切り替わります。ERRIDには、コマンド固有のエラーコードが含まれています。ファンクションブロックにタイムアウトエラーが発生すると、ERRがTRUE、ERRIDが1861 (16進数: 0x745)となります。次のコマンド実行により、FALSEにリセットされます。

ERRID: 最後に実行されたコマンドのADSエラーコード [▶ 119]またはコマンド固有のエラーコードです。次のコマンド実行により、0にリセットされます。

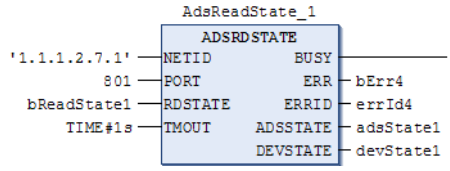
ADSSTATE:

ADSターゲットデバイスの特定の状態識別コードを含みます。ここで返されるコードは、すべてのADSサーバに対して指定されます:

- ADSSTATE_INVALID = 0;
- ADSSTATE_IDLE = 1;
- ADSSTATE_RESET = 2;
- ADSSTATE_INIT = 3;
- ADSSTATE_START = 4;
- ADSSTATE_RUN= 5;
- ADSSTATE_STOP = 6;
- ADSSTATE_SAVECFG = 7;
- ADSSTATE_LOADCFG = 8;
- ADSSTATE_POWERFAILURE = 9;
- ADSSTATE_POWERGOOD = 10;
- ADSSTATE_ERROR = 11;
- ADSSTATE_SHUTDOWN = 12;
- ADSSTATE_SUSPEND = 13;
- ADSSTATE_RESUME = 14;
- ADSSTATE_CONFIG = 15;
- ADSSTATE_RECONFIG = 16;
- ADSSTATE_STOPPING = 17;
- ADSSTATE_INCOMPATIBLE = 18;
- ADSSTATE_EXCEPTION = 19;

DEVSTATE: ADSターゲットデバイスの特定の状態識別コードです。ここで返されるコードは、そのADSデバイス固有の追加情報です。

FBDでのファンクションブロックコールの例:

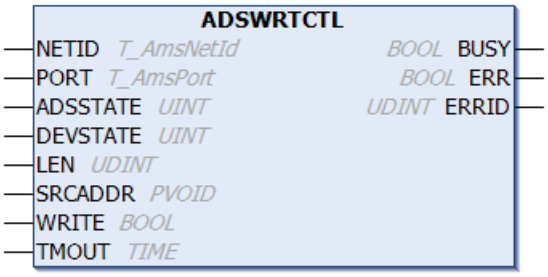


この例では、ネットワークアドレスが1.1.1.2.7.1のコンピュータ上のPLCランタイムシステム1（ポート番号801）の状態が問い合わされています。レスポンスはadsState = 1（IDLE）、追加コードなし(devState=0)です。

要件

開発環境	ターゲットシステム	PLCライブラリ(カテゴリグループ)
TwinCAT v3.1.0	PCまたはCX（x86、x64、ARM）	Tc2_System（システム）

3.2.7 ADSWRTCTL



このブロックにより、デバイスの開始、停止、リセットなど、ADSデバイスの状態に影響を与えるADS制御コマンドの実行が可能です。

VAR_INPUT

```
VAR_INPUT
    NETID     : T_AmsNetId;
    PORT      : T_AmsPort;
    ADSSTATE  : UINT;
    DEVSTATE  : UINT;
    LEN       : UDINT;
    SRCADDR   : PVOID;
    WRITE     : BOOL;
    TMOUT     : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

NETID: ADSコマンドの送信先であるターゲットデバイスのAMSネットワーク識別子の文字列です(型: T_AmsNetId [▶ 99]型)。

PORT: ADSデバイスのポート番号です(型: T_AmsPort [▶ 100]型)。

ADSSTATE: ADSターゲットデバイスの状態識別コードです。ここで示されるコードは、すべてのADSサーバに対して指定されます:

- ADSSTATE_IDLE = 1;
- ADSSTATE_RESET = 2;
- ADSSTATE_INIT = 3;
- ADSSTATE_START = 4;
- ADSSTATE_RUN= 5;
- ADSSTATE_STOP = 6;
- ADSSTATE_SAVECFG = 7;
- ADSSTATE_LOADCFG = 8;

DEVSTATE: ADSターゲットデバイスの特定の状態識別コードです。ここで返されるコードは、そのADSデバイス固有の追加情報です。

LEN: 書き込みするデータのバイト数を含みます。

SRCADDR: 書き込み用のデータが格納されているバッファのアドレスを含みます。LENのバイト長が取得できるバッファサイズを、プログラム内で指定する必要があります。

WRITE: ADSコマンドは、この入力の立ち上がりで実行されます。

TMOUT: ファンクションがキャンセルされるまでの時間です。

VAR_OUTPUT

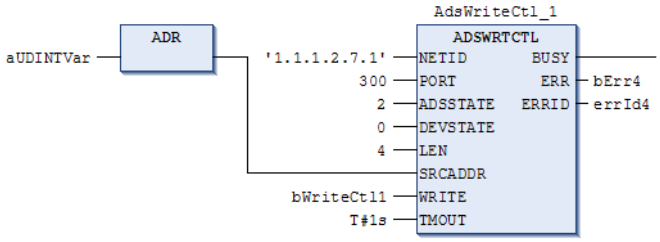
```
VAR_OUTPUT
    BUSY    : BOOL;
    ERR     : BOOL;
    ERRID   : UDINT;
END_VAR
```

BUSY: この出力は、ファンクションブロックがコマンドを実行するまではTRUEのままですが、Timeout入力の期間を過ぎるとFALSEになります。BUSY = TRUEの間は、入力で新しいコマンドが許可されません。これはサービスの実行ではなく、コマンド実行完了までの時間がモニタされることに注意してください。

ERR: この出力は、コマンド実行中にエラーが発生すると直ちにTRUEに切り替わります。ERRIDには、コマンド固有のエラーコードが含まれています。ファンクションブロックにタイムアウトエラーが発生すると、ERRがTRUE、ERRIDが1861 (16進数: 0x745)となります。次のコマンド実行により、FALSEにリセットされます。

ERRID: 最後に実行されたコマンドのADSエラーコード [▶ 119]またはコマンド固有のエラーコードです。次のコマンド実行により、0にリセットされます。

FBDでのブロックコールの例:



この例では、リセットコマンド(ADSSTATE=2)が追加データhex.AFFEとともにI/Oサーバ(ポート300)に送信されます。この結果、I/Oサーバがバスリセットを実行します。

要件

開発環境	ターゲットシステム	PLCライブラリ(カテゴリグループ)
TwinCAT v3.1.0	PCまたはCX (x86、x64、ARM)	Tc2_System (システム)

3.2.8 ADSRDDEVINFO

このブロックを使用して、全般的なデバイス情報を読み込みできます。

VAR_INPUT

```
VAR_INPUT
    NETID   : T_AmsNetId;
    PORT    : T_AmsPort;
```

```
    RDINFO : BOOL;
    TMOUT  : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

NETID: ADSコマンドの送信先であるターゲットデバイスのAMSネットワーク識別子の文字列です(型: T_AmsNetId [▶ 99]型)。

PORT: ADSデバイスのポート番号です(型: T_AmsPort [▶ 100]型)。

RDINFO: ADSコマンドは、この入力の立ち上がりで実行されます。

TMOUT: ファンクションがキャンセルされるまでの時間です。

VAR_OUTPUT

```
VAR_OUTPUT
    BUSY    : BOOL;
    ERR     : BOOL;
    ERRID   : UDINT;
    DEVNAME : STRING(19);
    DEVVER  : UDINT;
END_VAR
```

BUSY: この出力は、ファンクションブロックがコマンドを実行するまではTRUEのままですが、Timeout入力の期間を過ぎるとFALSEになります。BUSY = TRUEの間は、入力で新しいコマンドが許可されません。これはサービスの実行ではなく、コマンド実行完了までの時間がモニタされることに注意してください。

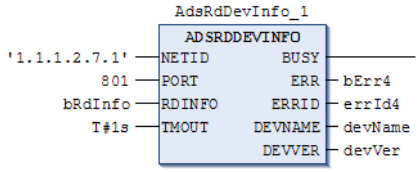
ERR: この出力は、コマンド実行中にエラーが発生すると直ちにTRUEに切り替わります。ERRIDには、コマンド固有のエラーコードが含まれています。ファンクションブロックにタイムアウトエラーが発生すると、ERRがTRUE、ERRIDが1861 (16進数: 0x745)となります。次のコマンド実行により、FALSEにリセットされます。

ERRID: 最後に実行されたコマンドのADSエラーコード [▶ 119]またはコマンド固有のエラーコードです。次のコマンド実行により、0にリセットされます。

DEVNAME: ADSデバイス名です。

DEVVER: ADSデバイスのバージョン番号です。

FBDでのブロックコールの例:



この例では、コンピュータ1.1.1.2.7.1上の1番目のPLCランタイムシステム (ポート801)のデバイス情報が読み込みされます。この結果、名前「PLC Server」およびバージョン番号02.00.7を受信します。

要件

開発環境	ターゲットシステム	PLCライブラリ(カテゴリグループ)
TwinCAT v3.1.0	PCまたはCX (x86、x64、ARM)	Tc2_System (システム)

3.3 拡張ADSファンクションブロック

3.3.1 概要

拡張ADSファンクションブロックにより、ADSデバイスとPLCタスク間のクライアント/サーバ通信を作成できます。ADSデバイスでは、Windowsアプリケーション(AdsDLL/AdsOcxを使用)であることも、その他のPLCランタイムシステムであることもあります。ADSデバイスとPLCタスク間の通信は、以下のサービスプリミティブを使用して処理されます。

- リクエスト
- 指示

- レスポンス
- 確認

ADSデバイスとPLCタスク間の通信は、次のように進行します。最初に、ADSデバイスがターゲットデバイス(PLCタスク)にリクエストを送信します。このリクエストは、指示によってターゲットデバイスに登録されます。これを受けて、ターゲットデバイス(PLCタスク)は対応するサービスを実行します。実行されるサービスは、インデックスグループ/オフセットパラメータによって符号化されます。次に、PLCがADSデバイスにレスポンスを送信します。レスポンスは、ADSソースデバイスによる確認として登録されます。

1つのPLCタスクにつき、１つの指示およびレスポンスファンクションブロックのインスタンスを使用することをお勧めします。(ADSREADIND、ADSREADRES、ADSWRITEIND、ADSWRITERES、ADSRDWRTIND、およびADSRDWRTRESのうち1つのインスタンス)。使用可能なADSサービス(READ、WRITE、およびREAD & WRITE)ごとに、適切な指示またはレスポンスファンクションブロックが用意されています。

拡張ADSファンクションブロック

サービス	名前	説明
READ	ADSREADIND [▶ 28]	ADSREAD指示
	ADSREADRES [▶ 31]	ADSREADレスポンス
WRITE	ADSWRITEIND [▶ 29]	ADSWRITE指示
	ADSWRITERES [▶ 32]	ADSWRITEレスポンス
READ & WRITE	ADSRDWRTIND [▶ 30]	ADS-READ & WRITE指示
	ADSRDWRTRES [▶ 33]	ADS-READ & WRITEレスポンス

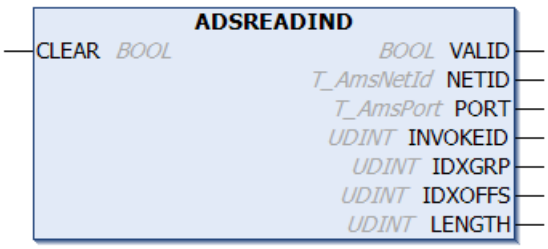
ADSデバイスは、ポートアドレスおよびネットワークアドレス(NETID)によってアドレス指定されます。

PLCタスクのポートアドレス = _TaskInfo.AdsPort

コメント:

- リクエストがPLCタスクにルーティングされるように、リクエスト作成時に最高値ビット(IG:=0x80000001など)をインデックスグループパラメータに入力する必要があります。
- 各PLCタスクには3つのFifo (ADSREADIND Fifo、ADSWRITEIND Fifo、およびADSRDWRTIND Fifo)があり、受信したリクエスト(指示)がFifoに一時的に格納されます。
  各Fifoには、指示が処理される(レスポンスが送信される)まで、最大で10の指示を格納できます。たとえば、12のADSREADリクエストがPLCタスクに同時に送信される場合、10のリクエストがFifoに指示として格納され、2つが確認レスポンス(破棄)されてADSエラーメッセージ1814 (0x716)が出力されます。この場合、エラーコードを分析し、必要に応じて2つの失敗したADSREADリクエストを繰り返します。指示はADSxxxxxxINDインスタンスを呼び出すことで、関連付けられたFifoから個別に取得されます。これにより新しい指示をFifoに格納可能になります。

3.3.2 ADSREADIND



このファンクションブロックはADSREADリクエストをPLCタスクに指示として登録し、これらの指示の処理を可能にします。VALID出力に指示のキューとして、CLEAR入力の立ち上がりで出力されます。またCLEAR入力の立ち上がりで処理されている指示が表示されます。CLEAR入力の立ち下がりにファンクションブロックが解除され、その他の指示を処理できます。指示の処理後、ADSREADRES [▶ 31]ファンクションブロックによってレスポンスをソースデバイスに送信する必要があります。出力のPORTおよびNETIDパラメータを使用してソースデバイスをアドレス指定します。ソースデバイスへのリクエストに対するレスポンスをソートするINVOKEIDパラメータは、パラメータとしてソースデバイスとして戻されます。

VAR_INPUT

```
VAR_INPUT
    CLEAR   : BOOL;
END_VAR
```

CLEAR: この入力の立ち上がりに、指示が処理済みとしてレポートされ、ADSREADINDファンクションブロックの出力がリセットされます。立ち下がりによってファンクションブロックが解除され、その他の指示を処理できます。

VAR_OUTPUT

```
VAR_OUTPUT
    VALID    : BOOL;
    NETID    : T_AmsNetId;
    PORT     : T_AmsPort;
    INVOKEID : UDINT;
    IDXGRP   : UDINT;
    IDXOFFS  : UDINT;
    LENGTH   : UDINT;
END_VAR
```

VALID: 指示がファンクションブロックにより登録されると出力がセットされます、CLEAR入力の立ち上がりからの指示が処理されたことがレポートされるまでセットされたままの状態となります。

NETID: ADSコマンドの送信先であるソースデバイスのAMSネットワーク識別子の文字列です(型: T_AmsNetId [▶ 99]型)。

PORT: ADSデバイスのポート番号です(型: T_AmsPort [▶ 100]型)。

INVOKEID: 送信されたコマンドのハンドルです。InvokeIdがソースデバイスから指定され、コマンドの識別に使用されます。

IDXGRP: リクエストするADSサービスのインデックスグループ番号(32ビット、符号なし)。

IDXOFFS: リクエストするADSサービスのインデックスオフセット番号(32ビット、符号なし)。

LENGTH: 読み込みするデータのバイト数。

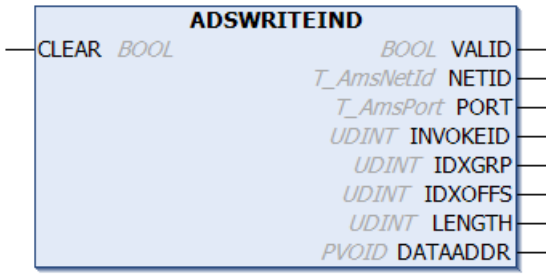
STでのブロックコールの例:

- AdsReadInd/AdsReadResファンクションブロックでの例 [▶ 108]

要件

開発環境	ターゲットシステム	PLCライブラリ(カテゴリグループ)
TwinCAT v3.1.0	PCまたはCX (x86、x64、ARM)	Tc2_System (システム)

3.3.3 ADSWRITEIND



このファンクションブロックはADSWRITEリクエストをPLCタスクに指示として登録し、これらの指示の処理を可能にします。VALID出力に指示のキューとして、CLEAR入力の立ち上がりで出力されます。またCLEAR入力の立ち上がりで処理されている指示が表示されます。CLEAR入力の立ち下がりにファンクションブロックが解除され、その他の指示を処理できます。指示の処理後、ADSWRITERES [▶ 32]ファンクションブロックによってレスポンスをソースデバイスに送信する必要があります。出力のPORTおよびNETIDパラメータを使用してソースデバイスをアドレス指定します。ソースデバイスへのリクエストに対するレスポンスを分類するINVOKEIDパラメータは、パラメータとしてソースデバイスとして戻されます。

VAR_INPUT

```
VAR_INPUT
    CLEAR : BOOL;
END_VAR
```

CLEAR: この入力の立ち上がりに、指示が処理済みとしてレポートされ、ADSWRITEINDファンクションブロックの出力がリセットされます(DATAADDR = 0、LENGTH = 0 !)。立ち下がりによってファンクションブロックが解除され、その他の指示を処理できます。

VAR_OUTPUT

```
VAR_OUTPUT
    VALID     : BOOL;
    NETID     : T_AmsNetId;
    PORT      : T_AmsPort;
    INVOKEID  : UDINT;
    IDXGRP    : UDINT;
    IDXOFFS   : UDINT;
    LENGTH    : UDINT;
    DATAADDR  : PVOID;
END_VAR
```

VALID: 指示がファンクションブロックにより登録されると出力がセットされます、CLEAR入力の立ち上がりからの指示が処理されたことがレポートされるまでセットされたままの状態となります。

NETID: ADSコマンドの送信先であるソースデバイスのAMSネットワーク識別子の文字列です(型: T_AmsNetId [▶ 99]型)。

PORT: ADSデバイスのポート番号です(型: T_AmsPort [▶ 100]型)。

INVOKEID: 送信されたコマンドのハンドルです。Invokeidがソースデバイスから指定され、コマンドの識別に使用されます。

IDXGRP: リクエストするADSサービスのインデックスグループ番号(32ビット、符号なし)。

IDXOFFS: リクエストするADSサービスのインデックスオフセット番号(32ビット、符号なし)。

LENGTH: 書き込みするデータのバイト数。

DATAADDR: 書き込み先データのデータバッファのアドレスです。

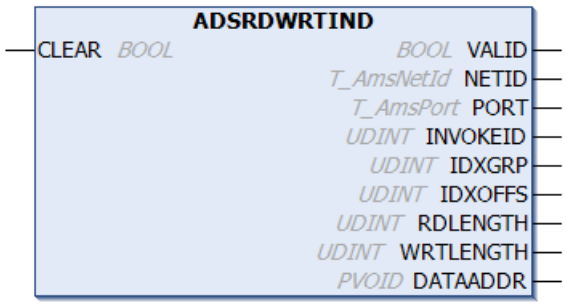
STでのブロックコールの例:

- AdsWriteInd/AdsWriteResファンクションブロックでの例 [▶ 110]

要件

開発環境	ターゲットシステム	PLCライブラリ(カテゴリグループ)
TwinCAT v3.1.0	PCまたはCX (x86、x64、ARM)	Tc2_System (システム)

3.3.4 ADSRDWRTIND



このファンクションブロックはADSRDWRTリクエストをPLCタスクに指示として登録し、これらの指示の処理を可能にします。VALID出力に指示のキューとして、CLEAR入力の立ち上がりで出力されます。またCLEAR入力の立ち上がりで処理されている指示が表示されます。CLEAR入力の立ち下がりにファンクションブロックが解除され、その他の指示を処理できます。指示の処理後、ADSRDWRTRES [▶ 33]ファンクションブロックによってレスポンスをソースデバイスに送信する必要があります。出力のPORTおよびNETIDパラメータを使用してソースデバイスをアドレス指定します。ソースデバイスへのリクエストに対するレスポンスを分類するINVOKEIDパラメータは、パラメータとしてソースデバイスとして戻されます。

VAR_INPUT

```
VAR_INPUT
    CLEAR : BOOL;
END_VAR
```

CLEAR: この入力の立ち上がりに、指示が処理済みとしてレポートされ、ADSRDWRTINDファンクションブロックの出力がリセットされます。立ち下がりにファンクションブロックが解除され、その他の指示を処理できます。

VAR_OUTPUT

```
VAR_OUTPUT
    VALID     : BOOL;
    NETID     : T_AmsNetId;
    PORT      : T_AmsPort;
    INVOKEID  : UDINT;
    IDXGRP    : UDINT;
    IDXOFFS   : UDINT;
    RDLENGTH  : UDINT;
    WRTLENGTH : UDINT;
    DATAADDR  : PVOID;
END_VAR
```

VALID: 指示がファンクションブロックにより登録されると出力がセットされます、CLEAR入力の立ち上がりからの指示が処理されたことがレポートされるまでセットされたままの状態となります。

NETID: ADSコマンドの送信先であるターゲットデバイスのAMSネットワーク識別子の文字列です(型: T_AmsNetId [▶ 99]型)。

PORT: ADSデバイスのポート番号です(型: T_AmsPort [▶ 100]型)。

INVOKEID: 送信されたコマンドのハンドルです。InvokeIdがソースデバイスから指定され、コマンドの識別に使用されます。

IDXGRP: リクエストするADSサービスのインデックスグループ番号(32ビット、符号なし)。

IDXOFFS: リクエストするADSサービスのインデックスオフセット番号(32ビット、符号なし)。

RDLENGTH: 読み込みするデータのバイト数。

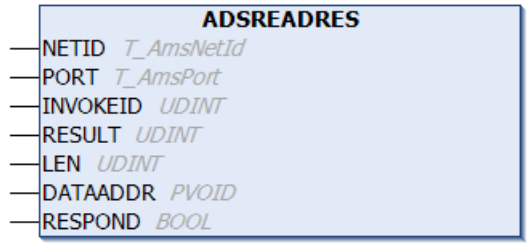
WRTLENGTH: 書き込みするデータのバイト数。

DATAADDR: 書き込み先データのデータバッファのアドレスです。

要件

開発環境	ターゲットシステム	PLCライブラリ(カテゴリグループ)
TwinCAT v3.1.0	PCまたはCX (x86、x64、ARM)	Tc2_System (システム)

3.3.5 ADSREADRES



ADSREADRESファンクションブロックを使用して、PLCタスクの指示に対して受信確認をします。RESPOND入力の立ち上がりに、レスポンスがADSソースデバイスに送信されます。ソースデバイスは、PORTおよびNETIDパラメータによってアドレス指定されます。ソースデバイスへのリクエストに対するレスポンスの分類のINVOKEIDパラメータは、ADSREADIND [▶ 28]ファンクションブロックの出力を採用します。エラーコードは、RESULTパラメータによってADSソースデバイスに返すことができます。

VAR_INPUT

```
VAR_INPUT
    NETID     : T_AmsNetId;
    PORT      : T_AmsPort;
    INVOKEID  : UDINT;
    RESULT    : UDINT;
    LEN       : UDINT;
    DATAADDR  : PVOID;
    RESPOND   : BOOL;
END_VAR
```

NETID: ADSコマンドの送信先であるターゲットデバイスのAMSネットワーク識別子の文字列です(型: T_AmsNetId [▶ 99]型)。

PORT: ADSデバイスのポート番号です(型: T_AmsPort [▶ 100]型)。

INVOKEID: 送信されたコマンドのハンドルです。InvokeIdがソースデバイスから指定され、コマンドの識別に使用されます。

RESULT: ソースデバイスに送信されるADSエラーコード [▶ 119]またはコマンド固有のエラーコードを含む文字列です。

LEN: 読み込みするデータ長(バイト)。

DATAADDR: 読み込みしたデータが位置するデータバッファのアドレスです。

RESPOND: ファンクションブロックは、この入力の立ち上がりで有効化されます。

VAR_OUTPUT

```
VAR_OUTPUT
(*none*)
END_VAR
```

STでのブロックコールの例:

- AdsReadInd/AdsReadResファンクションブロックでの例 [▶ 108]

要件

開発環境	ターゲットシステム	PLCライブラリ(カテゴリグループ)
TwinCAT v3.1.0	PCまたはCX (x86、x64、ARM)	Tc2_System (システム)

3.3.6 ADSWRITERES

ADSWRITERESファンクションブロックを使用して、PLCタスクの指示に対して受信確認をします。RESPOND入力の立ち上がりに、レスポンスがADSソースデバイスに送信されます。ソースデバイスは、PORTおよびNETIDパラメータによってアドレス指定されます。ソースデバイスへのリクエストに対するレスポンスの分類のINVOKEIDパラメータは、ADSWRITEIND [▶ 28]ファンクションブロックの出力を採用します。エラーコードは、RESULTパラメータによってADSソースデバイスに返すことができます。

VAR_INPUT

```
VAR_INPUT
    NETID     : T_AmsNetId;
    PORT      : T_AmsPort;
    INVOKEID  : UDINT;
    RESULT    : UDINT;
    RESPOND   : BOOL;
END_VAR
```

NETID: ADSコマンドの送信先であるターゲットデバイスのAMSネットワーク識別子の文字列です(型: T_AmsNetId [▶ 99]型)。

PORT: ADSデバイスのポート番号です(型: T_AmsPort [▶ 100]型)。

INVOKEID: 送信されたコマンドのハンドルです。InvokeIdがソースデバイスから指定され、コマンドの識別に使用されます。

RESULT: ソースデバイスに送信されるADSエラーコード [▶ 119]またはコマンド固有のエラーコードです。

RESPOND: ファンクションブロックは、この入力の立ち上がりで有効化されます。

VAR_OUTPUT

```
VAR_OUTPUT
(*none*)
END_VAR
```

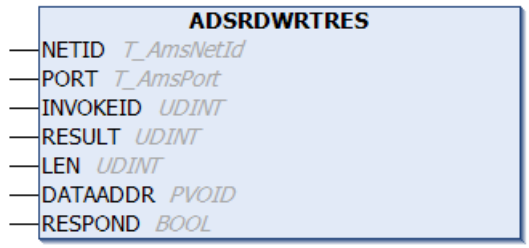
STでのブロックコールの例:

- AdsWriteInd/AdsWriteResファンクションブロックでの例 [▶ 110]

要件

開発環境	ターゲットシステム	PLCライブラリ(カテゴリグループ)
TwinCAT v3.1.0	PCまたはCX (x86、x64、ARM)	Tc2_System (システム)

3.3.7 ADSRDWRTRES



ADSRDWRTRESファンクションブロックを使用して、PLCタスクの指示に対して受信確認をします。RESPOND入力の立ち上がりに、レスポンスがADSソースデバイスに送信されます。ソースデバイスは、PORTおよびNETIDパラメータによってアドレス指定されます。ソースデバイスへのリクエストに対するレスポンスの分類のINVOKEIDパラメータは、ADSRDWRTIND [▶ 30]ファンクションブロックの出力を採用します。エラーコードは、RESULTパラメータによってADSソースデバイスに返すことができます。

VAR_INPUT

```
VAR_INPUT
    NETID    : T_AmsNetId;
    PORT     : T_AmsPort;
    INVOKEID : UDINT;
    RESULT   : UDINT;
    LEN      : UDINT;
    DATAADDR : PVOID;
    RESPOND  : BOOL;
END_VAR
```

NETID: ADSコマンドの送信先であるターゲットデバイスのAMSネットワーク識別子の文字列です(型: T_AmsNetId [▶ 99]型)。

PORT: ADSデバイスのポート番号です(型: T_AmsPort [▶ 100]型)。

INVOKEID: 送信されたコマンドのハンドルです。InvokeIdがソースデバイスから指定され、コマンドの識別に使用されます。

RESULT: ソースデバイスに送信されるADSエラーコード [▶ 119]またはコマンド固有のエラーコードを含む文字列です。

LEN: 読み込みするデータのバイト数です。

DATAADDR: 読み込みしたデータが位置するデータバッファのアドレスです。

RESPOND: ファンクションブロックは、この入力の立ち上がりで有効化されます。

VAR_OUTPUT

```
VAR_OUTPUT
(*none*)
END_VAR
```

要件

開発環境	ターゲットシステム	PLCライブラリ(カテゴリグループ)
TwinCAT v3.1.0	PCまたはCX (x86、x64、ARM)	Tc2_System (システム)

3.4 ファイルファンクションブロック

3.4.1 FB_EOF

このファンクションブロックは、ファイルの終端に達したかどうかをチェックします。

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy  : BOOL;
    bError : BOOL;
    nErrId : UDINT;
    bEOF   : BOOL;
END_VAR
```

bBusy: ファンクションブロックが有効な場合、この出力がTRUEにセットされ、フィードバックを受信するまでセットされたままの状態となります。bBusyがTRUEである限り、新しいコマンドを実行できません。

bError: コマンド実行中にエラーが発生して、bBusy出力がリセットされている場合、この出力がセットされます。

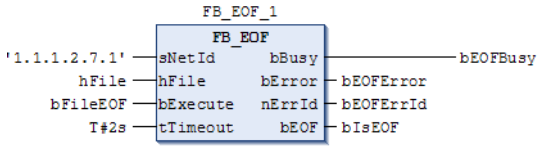
nErrId: bError出力のセット時に、ADSエラーコード [▶ 119]またはコマンド固有のエラーコードを返します。

bEOF: ファイルの終端に達すると、この出力がセットされます。

コマンド固有のエラーコード	考えられる原因
0x703	無効または不明なファイルの取り扱いです。
0x70E	ファイルが間違ったメソッド(古いFILEOPENファンクションブロックなど)で開かれました。

FBDでのファンクションブロックコールの例:

```
PROGRAM Test
VAR
    fbEOF       : FB_EOF;
    hFile       : UINT;
    bFileEOF    : BOOL;
    bEOFBusy    : BOOL;
    bEOFError   : BOOL;
    nEOFErrorId : UDINT;
    bIsEOF      : BOOL;
END_VAR
```



要件

開発環境	ターゲットシステム	PLCライブラリ(カテゴリグループ)
TwinCAT v3.1.0	PCまたはCX (x86、x64、ARM)	Tc2_System (システム)

3.4.2 FB_FileOpen

このファンクションブロックは、編集するファイルを新規作成する、または既存のファイルを開きます。

VAR_INPUT

```
VAR_INPUT
    sNetId      : T_AmsNetId;
    sPathName   : T_MaxString;
    nMode       : DWORD;
    ePath       : E_OpenPath := PATH_GENERIC;
    bExecute    : BOOL;
    tTimeout    : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

sNetId: ADSコマンドの送信先であるターゲットデバイスのAMSネットワーク識別子の文字列です(型: T_AmsNetId [▶ 99]型)。

sPathName: 開くファイルのパスおよびファイル名です。パスは、ローカルコンピュータのファイルシステムしか指定できません。ここでは、ネットワークパスは指定できません(型: T_MaxString [▶ 102]型)。

nMode: ファイルを開く際のモードです。下記のコードは、ライブラリで定数として事前定義済みの、ファイルを開く際の各種モードです。これらはシンボル名でファンクションブロックに入力します。ファイルを開く際のモードは、OR演算子と組み合わせられます。ファイルを開く際のモードの組み合わせは、CやC++のfopenファンクションでのモードと類似しています。

モード	説明
FOPEN_MODEREAD	r: ファイルを読み込み専用で開きます。ファイルが見つからない、または存在していない場合はエラーが返されます。
FOPEN_MODEWRITE	w: 新規 (空) のファイルを書き込み可で開きます。ファイルが既に存在している場合、このファイルは上書きされます。
FOPEN_MODEAPPEND	a: ファイルの終端から、書き込み可でファイルを開きます(追記)。ファイルが存在していない場合、ファイルが新規作成されます。
FOPEN_MODEREAD OR FOPEN_MODEPLUS	r+: 読み込み/書き込み用にファイルを開きます。このファイルが存在している必要があります。
FOPEN_MODEWRITE OR FOPEN_MODEPLUS	w+: 読み込み/書き込み用に新規 (空) のファイルを開きます。ファイルが既に存在している場合、このファイルは上書きされます。
FOPEN_MODEAPPEND OR FOPEN_MODEPLUS	a+: ファイルの終端で、読み込み/書き込み用にファイルを開きます(追記)。ファイルが存在していない場合、ファイルが新規作成されます。ファイルパスが既知である必要があります。ファイルパスが不明な場合、エラー1804が表示されます。ファイルがモード「a」または「a+」で開かれている場合、すべての書き込み操作は必ずファイルの終端で実行されます。ファイルポ

モード	説明
	インタはFB_FileSeekで移動できますが、書き込みの場合はファイルポインタがデフォルトでファイルの終端に移動されるため、既存のデータを上書きできません。
FOPEN_MODEBINARY	b: ファイルをバイナリモードで開きます。
FOPEN_MODETEXT	t: ファイルをテキストモードで開きます。

ePath: この入力を使用して、開くファイルのターゲットデバイス上のTwinCATシステムパスを選択します（型: E_OpenPath [▶ 96]型）。

bExecute: ファンクションブロックは、この入力の立ち上がりで有効化されます。

tTimeout: ADSコマンドの実行によって超過してはならないタイムアウトの長さを表します。

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy  : BOOL;
    bError : BOOL;
    nErrId : UDINT;
    hFile  : UINT;(* file handle *)
END_VAR
```

bBusy: ファンクションブロックが有効な場合、この出力がTRUEにセットされ、フィードバックを受信するまでセットされたままの状態となります。bBusyがTRUEである限り、新しいコマンドを実行できません。

bError: コマンド実行中にエラーが発生して、bBusy出力がリセットされている場合、この出力がセットされます。

nErrId: bError出力のセット時に、ADSエラーコード [▶ 119]またはコマンド固有のエラーコードを返します。

hFile: ファイルが正常に開かれた際に作成されるファイルハンドラーを含みます。このファイルハンドラーは、ファイルを編集するためのIDとして他のファイルファンクションブロックに送信されます。

コマンド固有のエラーコード	考えられる原因
0x703	nModeまたはePathパラメータが不明または無効です。
0x70C	ファイルが見つかりません。ファイル名またはパスが無効です。
0x716	空きのファイルハンドラーがありません。

ヒント:

ORは3つのパラメータまでにしか使用できません。

モード = [パラメータ1] OR [パラメータ2] OR [パラメータ3]

パラメータ1には、以下の値を使用できます。

- FOPEN_MODEREAD
- FOPEN_MODEWRITE
- FOPEN_MODEAPPEND

パラメータ2には、以下の値を使用できます。

- FOPEN_MODEPLUS

パラメータ3には、以下の値を使用できます。

- FOPEN_MODEBINARY
- FOPEN_MODETEXT

バイナリモードまたはテキストモードが指定されていない場合、ファイルはオペレーティングシステム変数によって定義されているモードで開かれます。通常、ファイルはテキストモードで開かれます。テキストまたはバイナリモードを宣言することを推奨します。このシステム変数は、PLCでは変更できません。
以下の組み合わせが可能です。

テキストファイルオープンモード	バイナリファイルオープンモード
FOPEN_MODEREAD OR FOPEN_MODETEXT	FOPEN_MODEREAD OR FOPEN_MODEBINARY

テキストファイルオープンモード	バイナリファイルオープンモード
FOPEN_MODEWRITE OR FOPEN_MODETEXT	FOPEN_MODEWRITE OR FOPEN_MODEBINARY
FOPEN_MODEAPPEND OR FOPEN_MODETEXT	FOPEN_MODEAPPEND OR FOPEN_MODEBINARY
FOPEN_MODEREAD OR FOPEN_MODEPLUS OR FOPEN_MODETEXT	FOPEN_MODEREAD OR FOPEN_MODEPLUS OR FOPEN_MODEBINARY
FOPEN_MODEWRITE OR FOPEN_MODEPLUS OR FOPEN_MODETEXT	FOPEN_MODEWRITE OR FOPEN_MODEPLUS OR FOPEN_MODEBINARY
FOPEN_MODEAPPEND OR FOPEN_MODEPLUS OR FOPEN_MODETEXT	FOPEN_MODEAPPEND OR FOPEN_MODEPLUS OR FOPEN_MODEBINARY

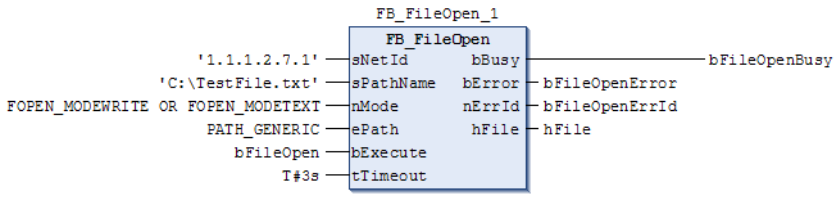
その他の組み合わせはすべて正しくありません。禁止されているオープンモードの例:
FOPEN_MODEBINARY OR FOPEN_MODETEXT
FOPEN_MODEWRITE OR FOPEN_MODEAPPEND

STでのブロックコールの例:

- PLCからのファイルアクセス [▶ 115]

FBDでのブロックコールの例:

```
PROGRAM Test
VAR
    fbFileOpen      : FB_FileOpen;
    bFileOpen       : BOOL;
    bFileOpenBusy   : BOOL;
    bFileOpenError  : BOOL;
    nFileOpenErrId  : UDINT;
    hFile           : UINT;
END_VAR
```



これにより、ファイル*TestFile2.txt*がドライブC:のルートディレクトリにテキストモードで作成(または上書き)されます。

要件

開発環境	ターゲットシステム	PLCライブラリ(カテゴリグループ)
TwinCAT v3.1.0	PCまたはCX (x86、x64、ARM)	Tc2_System (システム)

3.4.3 FB_FileClose

このファンクションブロックはファイルを閉じ、他のプログラムで処理を行うための定義された状態にします。

VAR_INPUT

```
VAR_INPUT
    sNetId    : T_AmsNetId;
    hFile     : UINT;
    bExecute  : BOOL;
    tTimeout  : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

sNetId: ADSコマンドの送信先であるターゲットデバイスのAMSネットワーク識別子の文字列です(型: T_AmsNetId [▶ 99]型)。

hFile: 閉じるファイルのハンドルです。

bExecute: ファンクションブロックは、この入力の立ち上がりで有効化されます。

tTimeout: ADSコマンドの実行によって超過してはならないタイムアウトの長さを表します。

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy    : BOOL;
    bError   : BOOL;
    nErrId   : UDINT;
END_VAR
```

bBusy: ファンクションブロックが有効な場合、この出力がTRUEにセットされ、フィードバックを受信するまでセットされたままの状態となります。bBusyがTRUEである限り、新しいコマンドを実行できません。

bError: コマンド実行中にエラーが発生して、bBusy出力がリセットされている場合、この出力がセットされます。

nErrId: bError出力のセット時に、ADSエラーコード [▶ 119]またはコマンド固有のエラーコードを返します。

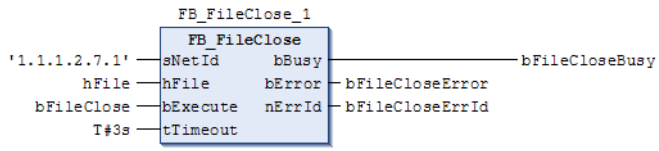
コマンド固有のエラーコード	考えられる原因
0x703	無効または不明なファイル操作です。
0x70E	ファイルが間違ったメソッド(古いFILEOPENファンクションブロックなど)で開かれました。

STでのブロックコールの例:

- PLCからのファイルアクセス [▶ 115]

FBDでのブロックコールの例:

```
PROGRAM Test
VAR
    fbFileClose       : FB_FileClose;
    hFile             : UINT;
    bFileClose        : BOOL;
    bFileCloseBusy    : BOOL;
    bFileCloseError   : BOOL;
    nFileCloseErrorId : UDINT;
END_VAR
```



ここでは、FB_FileOpenによって生成されたファイルハンドラーに関連付けられているファイルが再度閉じられます。

要件

開発環境	ターゲットシステム	PLCライブラリ(カテゴリグループ)
TwinCAT v3.1.0	PCまたはCX (x86、x64、ARM)	Tc2_System (システム)

3.4.4 FB_FileLoad

このファンクションブロックを使用して、ファイルの内容を読み込みできます。ファイルはバイナリモードで明示的に開かれ、内容が読み込みしてからファイルが再度閉じられます。

VAR_INPUT

```
VAR_INPUT
    sNetId     : T_AmsNetId;
    sPathName  : T_MaxString;
    pReadBuff  : PVOID;
    cbReadLen  : UDINT;
    bExecute   : BOOL;
    tTimeout   : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

sNetId: ADSコマンドのアドレス指定先であるターゲットデバイスのAMSネットワーク識別子の文字列です(型: T_AmsNetId [▶ 99]型)。

sPathName: 開くファイルの格納パスおよびファイル名です。パスは、コンピュータのローカルファイルシステムしか示せません。ここでは、ネットワークパスは指定できません(型: T_MaxString [▶ 102]型)。

pReadBuff: データが読み込みされるバッファのアドレスです。単一の変数、配列、構造体をバッファにすることができ、そのアドレスはADR演算子で指定します。

cbReadLen: 読み込みするバイト数です。

bExecute: ファンクションブロックは、この入力の立ち上がりで有効化されます。

tTimeout: 内部ADSコマンドの実行によって超過してはならないタイムアウトの長さを表します。

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy  : BOOL;
    bError : BOOL;
    nErrId : UDINT;
    cbRead : UDINT;
END_VAR
```

bBusy: ファンクションブロックが有効な場合、この出力がTRUEにセットされ、フィードバックを受信するまでセットされたままの状態となります。bBusyがTRUEである限り、新しいコマンドを実行できません。

bError: コマンド実行中にエラーが発生した場合、bBusy出力がリセットされていると、この出力がセットされます。

nErrId: bError出力のセット時に、ADSエラーコード [▶ 119]またはコマンド固有のエラーコードを返します。

cbRead: 読み込みされたバイト数です。

要件

開発環境	ターゲットシステム	PLCライブラリ(カテゴリグループ)
TwinCAT v3.1.4022.0	PCまたはCX (x86、x64、ARM)	Tc2_System (システム) >= v3.4.22.0

3.4.5 FB_FileGets

このファンクションブロックは、ファイルから文字列を読み込みします。文字列は改行文字まで(改行文字を含む)、ファイルの終端まで、またはsLineの最大許容長まで読み込みされます。ヌル終端文字列が自動的に追加されます。ファイルはテキストモードで開かれている必要があります。

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy    : BOOL;
    bError   : BOOL;
    nErrId   : UDINT;
    sLine    : T_MaxString;
    bEOF     : BOOL;
END_VAR
```

bBusy: ファンクションブロックが有効な場合、この出力がTRUEにセットされ、フィードバックを受信するまでセットされたままの状態となります。bBusyがTRUEである限り、新しいコマンドを実行できません。

bError: コマンド実行中にエラーが発生して、bBusy出力がリセットされている場合、この出力がセットされます。

nErrId: bError出力のセット時に、ADSエラーコード [▶ 119]またはコマンド固有のエラーコードを返します。

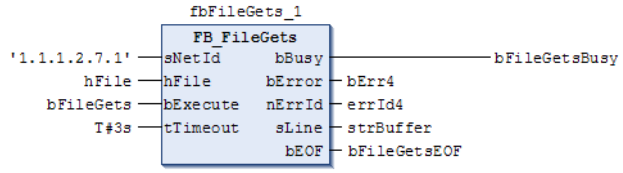
sLine: 読み込みした文字列です(型: T_MaxString [▶ 102]型)。

bEOF: ファイルの終端に達し、データバイトをそれ以上読み込みできなかった場合(cbRead=0)に、この出力がセットされます。データバイトを読み込みできた場合(cbRead>0)は、この出力はセットされません。

コマンド固有のエラーコード	考えられる原因
0x703	無効または不明なファイル操作です。
0x70A	読み込みバッファのメモリがありません。
0x70E	ファイルが間違ったメソッド(古いFILEOPENファンクションブロックなど)で開かれました。

FBDでのファンクションブロックコールの例:

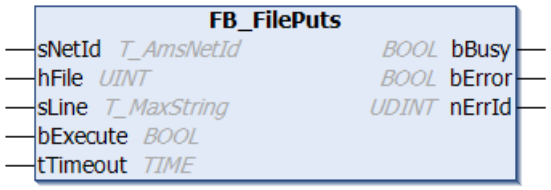
```
PROGRAM Test
VAR
    fbFileGets         : FB_FileGets;
    hFile              : UINT;
    bFileGets          : BOOL;
    bFileGetsBusy      : BOOL;
    bFileGetsError     : BOOL;
    nFileGetsErrorId   : UDINT;
    strBuffer          : STRING;
    bFileGetsEOF       : BOOL;
END_VAR
```



要件

開発環境	ターゲットシステム	PLCライブラリ(カテゴリグループ)
TwinCAT v3.1.0	PCまたはCX (x86、x64、ARM)	Tc2_System (システム)

3.4.6 FB_FilePuts



このファンクションブロックは、ファイルに文字列を書き込みします。文字列はヌル終端文字列までファイルに書き込みしますが、ヌル文字は含まれません。ファイルはテキストモードで開かれている必要があります。

VAR_INPUT

```
VAR_INPUT
    sNetId    : T_AmsNetId;
    hFile     : UINT;
    sLine     : T_MaxString;(* string to write *)
    bExecute  : BOOL;
    tTimeout  : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

sNetId: ADSコマンドの送信先であるターゲットデバイスのAMSネットワーク識別子の文字列です(型: T_AmsNetId [▶ 99]型)。

hFile: ファンクションブロックFB_FileOpen作成時に生成されたファイルハンドラーです。

sLine: ファイルに書き込みされる文字列です(型: T_MaxString [▶ 102]型)。

bExecute: ファンクションブロックは、この入力の立ち上がりで有効化されます。

tTimeout: ADSコマンドの実行によって超過してはならないタイムアウトの長さを表します。

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy  : BOOL;
    bError : BOOL;
    nErrId : UDINT;
END_VAR
```

bBusy: ファンクションブロックが有効な場合、この出力がTRUEにセットされ、フィードバックを受信するまでセットされたままの状態となります。bBusyがTRUEである限り、新しいコマンドを実行できません。

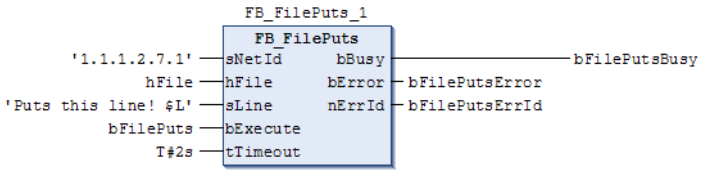
bError: コマンド実行中にエラーが発生して、bBusy出力がリセットされている場合、この出力がセットされます。

nErrId: bError出力のセット時に、ADSエラーコード [▶ 119]またはコマンド固有のエラーコードを返します。

コマンド固有のエラーコード	考えられる原因
0x703	無効または不明なファイル操作です。
0x70E	ファイルが間違ったメソッド(古いFILEOPENファンクションブロックなど)で開かれました。

FBDでのファンクションブロックコールの例:

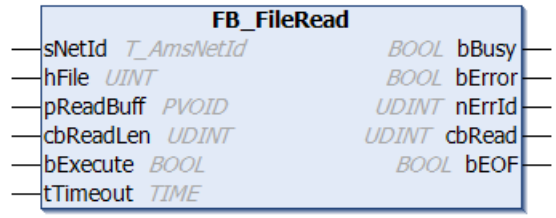
```
PROGRAM Test
VAR
    fbFilePuts       : FB_FilePuts;
    hFile            : UINT;
    bFilePuts        : BOOL;
    bFilePutsBusy    : BOOL;
    bFilePutsError   : BOOL;
    nFilePutsErrorId : UDINT;
END_VAR
```



要件

開発環境	ターゲットシステム	PLCライブラリ(カテゴリグループ)
TwinCAT v3.1.0	PCまたはCX (x86、x64、ARM)	Tc2_System (システム)

3.4.7 FB_FileRead



このファンクションブロックを使用して、既に開かれているファイルの内容を読み込みできます。読み込みアクセスの前に、ファイルが対応するモードで開かれている必要があります。

VAR_INPUT

```
VAR_INPUT
    sNetId      : T_AmsNetId;
    hFile       : UINT;
    pReadBuff   : PVOID;
    cbReadLen   : UDINT;
    bExecute    : BOOL;
    tTimeout    : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

sNetId: ADSコマンドの送信先であるターゲットデバイスのAMSネットワーク識別子の文字列です(型: T_AmsNetId [▶ 99]型)。

hFile: ファンクションブロックFB_FileOpen作成時に生成されたファイルハンドラーです。

pReadBuff: データが読み込みされるバッファのアドレスです。単一の変数、配列、構造体をバッファにすることができ、そのアドレスはADR演算子で指定します。

cbReadLen: 読み込みするバイト数です。

bExecute: ファンクションブロックは、この入力の立ち上がりで有効化されます。

tTimeout: ADSコマンドの実行によって超過してはならないタイムアウトの長さを表します。

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy  : BOOL;
    bError : BOOL;
    nErrId : UDINT;
    cbRead : UDINT;
    bEOF   : BOOL;
END_VAR
```

bBusy: ファンクションブロックが有効な場合、この出力がTRUEにセットされ、フィードバックを受信するまでセットされたままの状態となります。bBusyがTRUEである限り、新しいコマンドを実行できません。

bError: コマンド実行中にエラーが発生して、bBusy出力がリセットされている場合、この出力がセットされます。

nErrId: bError出力のセット時に、ADSエラーコード [▶ 119]またはコマンド固有のエラーコードを返します。

cbRead: 読み込みされたバイト数を含みます。

bEOF: ファイルの終端に達し、データバイトをそれ以上読み込みできなかった場合(cbRead=0)に、この出力がセットされます。データバイトを読み込みできた場合(cbRead>0)は、この出力はセットされません。

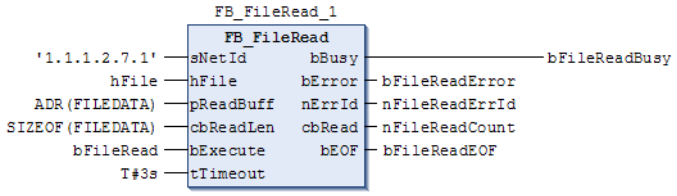
コマンド固有のエラーコード	考えられる原因
0x703	無効または不明なファイル操作です。
0x70A	読み込みバッファのメモリがありません。
0x70E	ファイルが間違ったメソッド(古いFILEOPENファンクションブロックなど)で開かれました。

STでのブロックコールの例:

- PLCからのファイルアクセス [▶ 115]

FBDでのファンクションブロックコールの例:

```
PROGRAM Test
VAR
    fbFileRead        : FB_FileRead;
    hFile             : UINT;
    bFileRead         : BOOL;
    bFileReadBusy     : BOOL;
    bFileReadError    : BOOL;
    nFileReadErrorId  : UDINT;
    nFileReadCount    : UDINT;
    bFileReadEOF      : BOOL;
    fileData          : ARRAY[0..9] OF BYTE;
END_VAR
```

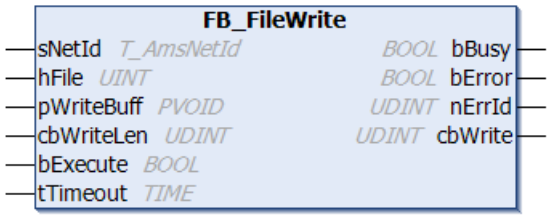


bExecuteでの立ち上がりおよび読み込み命令の正常実行後、ファイルから読み込みされたバイト数はFILEDATAで確認できます。パラメータcbReadを使用して、最後の読み込み操作中に実際に読み込みしたバイト数を判定できます。

要件

開発環境	ターゲットシステム	PLCライブラリ(カテゴリグループ)
TwinCAT v3.1.0	PCまたはCX (x86、x64、ARM)	Tc2_System (システム)

3.4.8 FB_FileWrite



このファンクションブロックは、ファイルにデータを書き込みします。書き込みアクセスの場合、ファイルが対応するモードで開かれ、外部プログラムによって処理が行えるように再度閉じる必要があります。

VAR_INPUT

```
VAR_INPUT
    sNetId     : T_AmsNetId;
    hFile      : UINT;
    pWriteBuff : PVOID;
    cbWriteLen : UDINT;
```

```
    bExecute   : BOOL;
    tTimeout   : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

sNetId: ADSコマンドの送信先であるターゲットデバイスのAMSネットワーク識別子の文字列です(型: T_AmsNetId [▶ 99]型)。

hFile: ファンクションブロックFB_FileOpen作成時に生成されたファイルハンドラーです。

pWriteBuff: データが書き込みされるバッファのアドレスです。単一の変数、配列、構造体をバッファにすることができ、そのアドレスはADR演算子で指定します。

cbWriteLen: 書き込みするバイト数です。

bExecute: ファンクションブロックは、この入力の立ち上がりで有効化されます。

tTimeout: ADSコマンドの実行によって超過してはならないタイムアウトの長さを表します。

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy    : BOOL;
    bError   : BOOL;
    nErrId   : UDINT;
    cbWrite  : UDINT;
END_VAR
```

bBusy: ファンクションブロックが有効な場合、この出力がTRUEにセットされ、フィードバックを受信するまでセットされたままの状態となります。bBusyがTRUEである限り、新しいコマンドを実行できません。

bError: コマンド実行中にエラーが発生して、bBusy出力がリセットされている場合、この出力がセットされます。

nErrId: *bError*出力のセット時に、ADSエラーコード [▶ 119]またはコマンド固有のエラーコードを返します。

cbWrite: 最後に正常に書き込みしたデータバイト数を含みます。正常に書き込みしたデータバイト数が、リクエストされた長さ(cbWriteLen)未満であるか、ゼロであると書き込みエラーとなります。たとえば、データストレージデバイスの空き容量がない場合、書き込みエラーが発生する可能性があります。書き込みエラーが発生すると、bErrorおよびnErrID出力はセットされません。PLCアプリケーションは書き込みするデータバイト数を認識しているため、実際に書き込みした長さと長さを比較して、書き込みエラーを検出できます。書き込みエラーが発生すると、内部的なファイルポインタの位置が未定義となります。

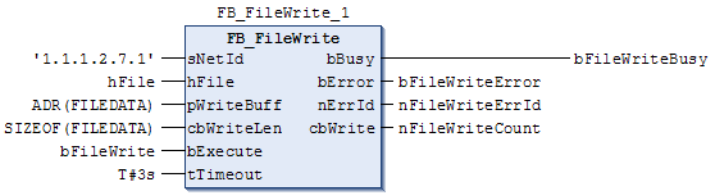
コマンド固有のエラーコード	考えられる原因
0x703	無効または不明なファイル操作です。
0x70E	ファイルが間違ったメソッド(古いFILEOPENファンクションブロックなど)で開かれました。

STでのブロックコールの例:

- PLCからのファイルアクセス [▶ 115]

FBDでのブロックコールの例:

```
PROGRAM Test
VAR
    fbFileWrite      : FB_FileWrite;
    hFile            : UINT;
    bFileWrite       : BOOL;
    bFileWriteBusy   : BOOL;
    bFileWriteError  : BOOL;
    nFileWriteErrorId: UDINT;
    nFileWriteCount  : UDINT;
    fileData         : ARRAY[0..9] OF BYTE;
END_VAR
```



この例では、bFileWriteでの立ち上がり後、配列FILEDATAの10バイトがファイルに書き込みします。

要件

開発環境	ターゲットシステム	PLCライブラリ(カテゴリグループ)
TwinCAT v3.1.0	PCまたはCX (x86、x64、ARM)	Tc2_System (システム)

3.4.9 FB_FileSeek

このファンクションブロックは、開いているファイルのファイルポインタを定義可能な位置にセットします。

VAR_INPUT

```
VAR_INPUT
    sNetId    : T_AmsNetId;
    hFile     : UINT;
    nSeekPos  : DINT;(* new seek pointer position *)
    eOrigin   : E_SeekOrigin:= SEEK_SET;
    bExecute  : BOOL;
    tTimeout  : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

sNetId: ADSコマンドの送信先であるターゲットデバイスのAMSネットワーク識別子の文字列です(型: T_AmsNetId [▶ 99]型)。

hFile: ファンクションブロックFB_FileOpen作成時に生成されたファイルハンドラーです。

nSeekPos: ファイルポインタの新しい位置です。

eOrigin: ファイルポインタが移動される相対位置です(型: E_SeekOrigin [▶ 96]型)。

bExecute: ファンクションブロックは、この入力の立ち上がりで有効化されます。

tTimeout: ADSコマンドの実行によって超過してはならないタイムアウトの長さを表します。

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy  : BOOL;
    bError : BOOL;
    nErrId : UDINT;
END_VAR
```

bBusy: ファンクションブロックが有効な場合、この出力がTRUEにセットされ、フィードバックを受信するまでセットされたままの状態となります。bBusyがTRUEである限り、新しいコマンドを実行できません。

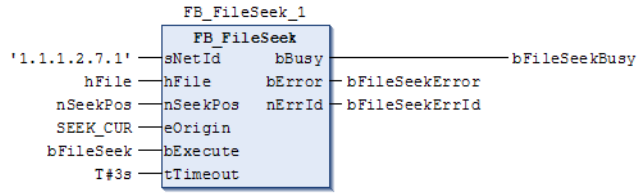
bError: コマンド実行中にエラーが発生して、bBusy出力がリセットされている場合、この出力がセットされます。

nErrId: bError出力のセット時に、ADSエラーコード [▶ 119]またはコマンド固有のエラーコードを返します。

コマンド固有のエラーコード	考えられる原因
0x703	無効または不明なファイル操作です。
0x70E	ファイルが間違ったメソッド(古いFILEOPENファンクションブロックなど)で開かれました。

FBDでのファンクションブロックコールの例:

```
PROGRAM Test
VAR
    fbFileSeek       : FB_FileSeek;
    hFile            : UINT;
    nSeekPos         : DINT;
    bFileSeek        : BOOL;
    bFileSeekBusy    : BOOL;
    bFileSeekError   : BOOL;
    nFileSeekErrorId : UDINT;
END_VAR
```



要件

開発環境	ターゲットシステム	PLCライブラリ(カテゴリグループ)
TwinCAT v3.1.0	PCまたはCX (x86、x64、ARM)	Tc2_System (システム)

3.4.10 FB_FileTell

このファンクションブロックは、ファイルポインタの現在位置を判定します。位置は、ファイルの開始からの相対オフセットを示します。

「Append at end of file」モードで開かれたファイルでは、現在位置は次の書き込みアクセスの位置ではなく、最後のI/O操作によって判定されることに注意してください。たとえば、読み込み操作後、ファイルポインタは次の書き込みアクセスが行われる位置ではなく、次の読み込みアクセスが行われる位置に配置されます(Appendモードでは、ファイルポインタは書き込み操作の前に必ず終端に移動されます)。

以前にI/O操作が行われておらず、ファイルがAppendモードで開かれている場合、ファイルポインタはファイルの先頭に配置されます。

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy    : BOOL;
    bError   : BOOL;
    nErrId   : UDINT;
    nSeekPos : DINT;(* On error, nSEEKPOS returns -1 *)
END_VAR
```

bBusy: ファンクションブロックが有効な場合、この出力がTRUEにセットされ、フィードバックを受信するまでセットされたままの状態となります。bBusyがTRUEである限り、新しいコマンドを実行できません。

bError: コマンド実行中にエラーが発生して、bBusy出力がリセットされている場合、この出力がセットされます。

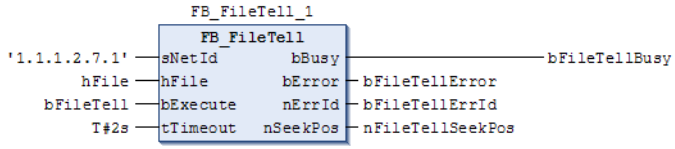
nErrId: bError出力のセット時に、ADSエラーコード [▶ 119]またはコマンド固有のエラーコードを返します。

nSeekPos: ファイルポインタの現在位置を返します。

コマンド固有のエラーコード	考えられる原因
0x703	無効または不明なファイル操作です。
0x70E	ファイルが間違ったメソッド(古いFILEOPENファンクションブロックなど)で開かれました。

FBDでのファンクションブロックコールの例:

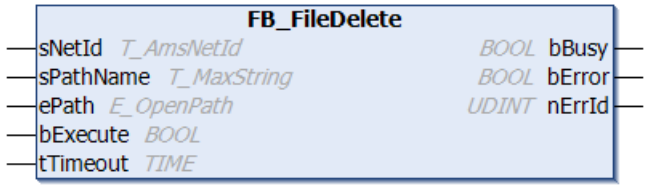
```
PROGRAM Test
VAR
    fbFileTell         : FB_FileTell;
    hFile              : UINT;
    bFileTell          : BOOL;
    bFileTellBusy      : BOOL;
    bFileTellError     : BOOL;
    nFileTellErrorId   : UDINT;
    nFileTellSeekPos   : DINT;
END_VAR
```



要件

開発環境	ターゲットシステム	PLCライブラリ(カテゴリグループ)
TwinCAT v3.1.0	PCまたはCX (x86、x64、ARM)	Tc2_System (システム)

3.4.11 FB_FileDelete



このファンクションブロックは、データストレージデバイスからファイルを削除します。

VAR_INPUT

```
VAR_INPUT
    sNetId    : T_AmsNetId;
    sPathName : T_MaxString;(* file path and name *)
    ePath     : E_OpenPath := PATH_GENERIC;
    bExecute  : BOOL;
    tTimeout  : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

sNetId: ADSコマンの送信先であるターゲットデバイスのAMSネットワーク識別子の文字列です(型: T_AmsNetId [▶ 99]型)。

sPathName: 開くファイルのパスおよびファイル名です。(型: T_MaxString [▶ 102]型)。

ePath: この入力を使用して、開くファイルのターゲットデバイス上のTwinCATシステムパスを選択できます(型: E_OpenPath [▶ 96]型)。

bExecute: ファンクションブロックは、この入力の立ち上がりで有効化されます。

tTimeout: ADSコマンドの実行によって超過してはならないタイムアウトの長さを表します。

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy  : BOOL;
    bError : BOOL;
    nErrId : UDINT;
END_VAR
```

bBusy: ファンクションブロックが有効な場合、この出力がTRUEにセットされ、フィードバックを受信するまでセットされたままの状態となります。bBusyがTRUEである限り、新しいコマンドを実行できません。

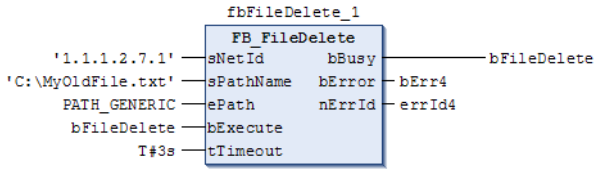
bError: コマンド実行中にエラーが発生して、bBusy出力がリセットされている場合、この出力がセットされます。

nErrId: bError出力のセット時に、ADSエラーコード [▶ 119]またはコマンド固有のエラーコードを返します。

コマンド固有のエラーコード	考えられる原因
0x70C	ファイルが見つかりません。sPathNameまたはePathパラメータが無効です。

FBDでのファンクションブロックコールの例:

```
PROGRAM Test
VAR
    fbFileDelete     : FB_FileDelete;
    bFileDelete      : BOOL;
    bFileDeleteBusy  : BOOL;
    bFileDeleteError : BOOL;
    nFileDeleteErrId : UDINT;
END_VAR
```



要件

開発環境	ターゲットシステム	PLCライブラリ(カテゴリグループ)
TwinCAT v3.1.0	PCまたはCX (x86、x64、ARM)	Tc2_System (システム)

3.4.12 FB_FileRename

このファンクションブロックを使用して、ファイル名を変更できます。

VAR_INPUT

```
VAR_INPUT
    sNetId   : T_AmsNetId;
    sOldName : T_MaxString;
    sNewName : T_MaxString;
    ePath    : E_OpenPath := PATH_GENERIC;     (* Default: generic file path*)
    bExecute : BOOL;
    tTimeout : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

sNetId: ADSコマンドの送信先であるターゲットデバイスのAMSネットワーク識別子の文字列です(型: T_AmsNetId [▶ 99]型)。

sOldName: 以前のファイル名(型: T_MaxString [▶ 102]型)。

sNewName: 新しいファイル名(型: T_MaxString [▶ 102]型)。

ePath: この入力を使用して、開くファイルのターゲットデバイス上のTwinCATシステムパスを選択できます(型: E_OpenPath [▶ 96]型)。

bExecute: ファンクションブロックは、この入力の立ち上がりで有効化されます。

tTimeout: ADSコマンドの実行によって超過してはならないタイムアウトの長さを表します。

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy  : BOOL;
    bError : BOOL;
    nErrId : UDINT;
END_VAR
```

bBusy: ファンクションブロックが有効な場合、この出力がTRUEにセットされ、フィードバックを受信するまでセットされたままの状態となります。bBusyがTRUEである限り、新しいコマンドを実行できません。

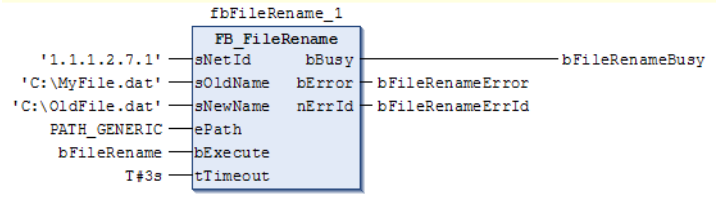
bError: コマンド実行中にエラーが発生して、bBusy出力がリセットされている場合、この出力がセットされます。

nErrId: bError出力のセット時に、ADSエラーコード [▶ 119]またはコマンド固有のエラーコードを返します。

コマンド固有のエラーコード	考えられる原因
0x70C	ファイルが見つかりません。sOldName、sNewName、またはePathパラメータが無効です。

FBでのファンクションブロックコールの例:

```
PROGRAM Test
VAR
    fbFileRename      : FB_FileRename;
    bFileRename       : BOOL;
    bFileRenameBusy   : BOOL;
    bFileRenameError  : BOOL;
    nFileRenameErrId  : UDINT;
END_VAR
```



要件

開発環境	ターゲットシステム	PLCライブラリ(カテゴリグループ)
TwinCAT v3.1.0	PCまたはCX (x86、x64、ARM)	Tc2_System (システム)

3.4.13 FB_CreateDir

このファンクションブロックを使用して、データストレージデバイス上にディレクトリを新規作成できます。

VAR_INPUT

```
VAR_INPUT
    sNetId    : T_AmsNetId;
    sPathName : T_MaxString;
    ePath     : E_OpenPath := PATH_GENERIC; (* Default: generic file path*)
    bExecute  : BOOL;
    tTimeout  : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

sNetId: ADSコマンドの送信先であるターゲットデバイスのAMSネットワーク識別子の文字列です(型: T_AmsNetId [▶ 99]型)。

sPathName: 新しいディレクトリ名です。このファンクションブロックのコール時にできる操作は、ディレクトリの新規作成のみです(型: T_MaxString [▶ 102]型)。

ePath: この入力を使用して、ターゲットデバイス上の新規ディレクトリのTwinCATシステムパスを選択できます(型: E_OpenPath [▶ 96]型)。

bExecute: ファンクションブロックは、この入力の立ち上がりで有効化されます。

tTimeout: ADSコマンドの実行によって超過してはならないタイムアウトの長さを表します。

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy   : BOOL;
    bError  : BOOL;
    nErrId  : UDINT;
END_VAR
```

bBusy: ファンクションブロックが有効な場合、この出力がTRUEにセットされ、フィードバックを受信するまでセットされたままの状態となります。bBusyがTRUEである限り、新しいコマンドを実行できません。

bError: コマンド実行中にエラーが発生して、bBusy出力がリセットされている場合、この出力がセットされます。

nErrId: bError出力のセット時に、ADSエラーコード [▶ 119]またはコマンド固有のエラーコードを返します。

コマンド固有のエラーコード	考えられる原因
0x723	フォルダが既に存在しているか、またはsPathNameまたはePathパラメータが無効です。

STの例:

bCreateでの立ち上がりに、メインディレクトリC:¥に「PRJDATA」という名前のディレクトリが新規作成されます。bRemoveの立ち上がりに、この名前のディレクトリを削除できます。

bBootFolder = TRUEの場合、*..¥TwinCAT¥Boot*ディレクトリにディレクトリを作成する、またはこのディレクトリのディレクトリを削除することが可能です。

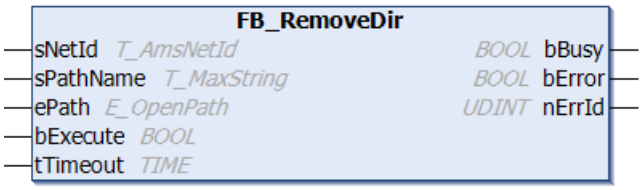
```
PROGRAM MAIN
VAR
    sFolderName   : STRING := 'PRJDATA'; (* folder name *)
    bBootFolder   : BOOL;

    ePath         : E_OpenPath; (* folders root path *)
    sPathName     : STRING;

    fbCreateDir   : FB_CreateDir;
    bCreate       : BOOL;
    bCreate_Busy  : BOOL;
    bCreate_Error : BOOL;
    nCreate_ErrID : UDINT;

    fbRemoveDir   : FB_RemoveDir;
    bRemove       : BOOL;
```

```
    bRemove_Busy  : BOOL;
    bRemove_Error : BOOL;
    nRemove_ErrID : UDINT;
END_VAR


ePath := SEL( bBootFolder, PATH_GENERIC, PATH_BOOTPATH );
sPathName := SEL( bBootFolder, CONCAT('C:\', sFolderName), sFolderName );

IF bCreate THEN
    bCreate := FALSE;
    fbCreateDir( bExecute := FALSE );
    fbCreateDir(sNetId:= '',
        sPathName:= sPathName,
        ePath:= ePath,
        bExecute:= TRUE,
        tTimeout:= DEFAULT_ADS_TIMEOUT,
        bBusy=>bCreate_Busy, bError=>bCreate_Error, nErrId=>nCreate_ErrID );
ELSE
    fbCreateDir( bExecute := FALSE, bBusy=>bCreate_Busy, bError=>bCreate_Error, nErrId=>nCreate_ErrI
D );
END_IF

IF bRemove THEN
    bRemove := FALSE;
    fbRemoveDir( bExecute := FALSE );
    fbRemoveDir(sNetId:= '',
        sPathName:= sPathName,
        ePath:= ePath,
        bExecute:= TRUE,
        tTimeout:= DEFAULT_ADS_TIMEOUT,
        bBusy=>bRemove_Busy, bError=>bRemove_Error, nErrId=>nRemove_ErrID );
ELSE
    fbRemoveDir( bExecute := FALSE, bBusy=>bRemove_Busy, bError=>bRemove_Error, nErrId=>nRemove_ErrI
D );
END_IF
```

要件

開発環境	ターゲットシステム	PLCライブラリ(カテゴリグループ)
TwinCAT v3.1.0	PCまたはCX (x86、x64、ARM)	Tc2_System (システム)

3.4.14 FB_RemoveDir



このファンクションブロックを使用して、データストレージデバイスからディレクトリを削除できます。ファイルを含んでいるディレクトリは削除できません。

VAR_INPUT

```
VAR_INPUT
    sNetId    : T_AmsNetId;
    sPathName : T_MaxString;
    ePath     : E_OpenPath := PATH_GENERIC; (* Default: generic file path*)
    bExecute  : BOOL;
    tTimeout  : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

sNetId: ADSコマンドの送信先であるターゲットデバイスのAMSネットワーク識別子の文字列です(型: T_AmsNetId [▶ 99]型)。

sPathName: 削除するディレクトリ名です。このファンクションブロックのコール時には、ディレクトリを1つだけ削除できます。sPathNameの最後のコンポーネントには、削除するディレクトリ名が含まれている必要があります(型: T_MaxString [▶ 102]型)。

ePath: この入力を使用して、ターゲットデバイスから削除するディレクトリのTwinCATシステムパスを選択できます(型: E_OpenPath [▶ 96]型)。

bExecute: ファンクションブロックは、この入力の立ち上がりで有効化されます。

tTimeout: ADSコマンドの実行によって超過してはならないタイムアウトの長さを表します。

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy  : BOOL;
    bError : BOOL;
    nErrId : UDINT;
END_VAR
```

bBusy: ファンクションブロックが有効な場合、この出力がTRUEにセットされ、フィードバックを受信するまでセットされたままの状態となります。bBusyがTRUEである限り、新しいコマンドを作成できません。

bError: コマンド実行中にエラーが発生して、*bBusy*出力がリセットされている場合、この出力がセットされます。

nErrId: *bError*出力のセット時に、ADSエラーコード [▶ 119]またはコマンド固有のエラーコードを返します。

コマンド固有のエラーコード	考えられる原因
0x70C	フォルダが見つからないか、またはsPathNameまたはePathパラメータが無効です。

STの例:

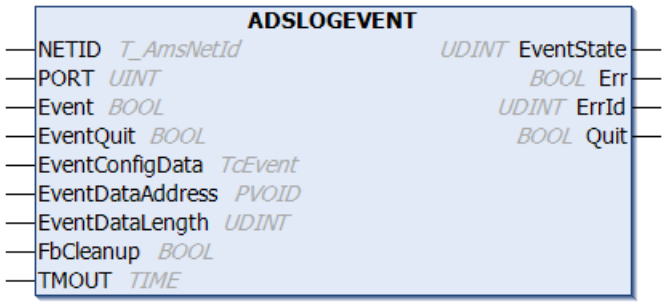
FB_CreateDir [▶ 49]の説明を参照してください。

要件

開発環境	ターゲットシステム	PLCライブラリ(カテゴリグループ)
TwinCAT v3.1.0	PCまたはCX (x86、x64、ARM)	Tc2_System (システム)

3.5 イベントロガーファンクションブロック

3.5.1 ADSLOGEVENT



このファンクションブロックにより、TwinCATイベントロガーへのメッセージの送信および確認レスポンスが可能です。

VAR_INPUT

```
VAR_INPUT
    NETID             : T_AmsNetId;
    PORT              : T_AmsPort;
    Event             : BOOL;
    EventQuit         : BOOL;
    EventConfigData   : TcEvent;
    EventDataAddress  : PVOID;
    EventDataLength   : UDINT;
    FbCleanup         : BOOL;
    TMOUT             : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

NETID: ADSコマンドの送信先であるターゲットデバイスのAMSネットワーク識別子の文字列です(型: T_AmsNetId [▶ 99]型)。

PORT: ADSデバイスのポート番号です。TwinCATイベントロガーのポート番号は110です(型: T_AmsPort [▶ 100]型)。

Event: 立ち上がりではイベントの「発生」が、立ち下りではイベントの「終了」が信号送信されます。

EventQuit: 立ち上がりで、イベントが確認レスポンスされます。

EventConfigData: イベントパラメータのデータ構造体です(型: TcEvent [▶ 102]型)。

EventDataAddress: イベントと共に送信されるデータのアドレスです。

EventDataLength: イベントと共に送信されるデータの長さです。

FbCleanup: TRUEの場合、ファンクションブロックが完全に初期化されます。

TMOUT: ファンクションがキャンセルされるまでの時間を示します。

VAR_OUTPUT

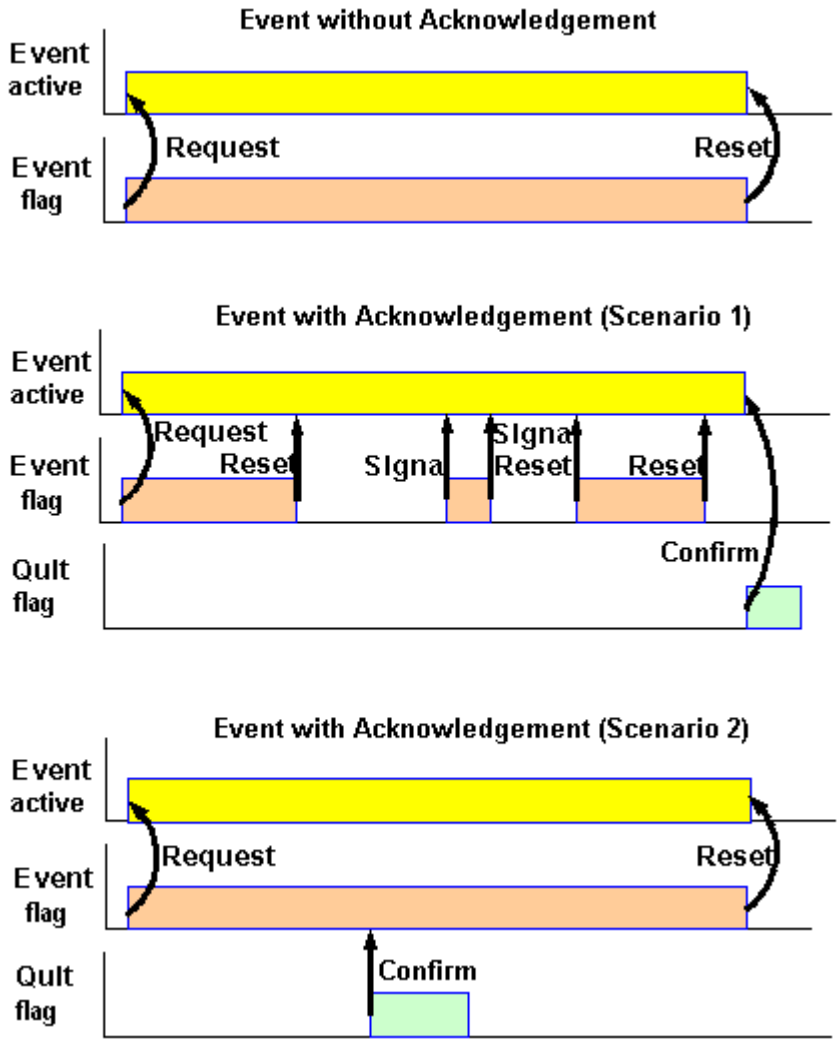
```
VAR_OUTPUT
    EventState    : UDINT;
    Err           : BOOL;
    ErrId         : UDINT;
    Quit          : BOOL;
END_VAR
```

EventState: イベントの状態です。

Err: この出力は、コマンド実行中にエラーが発生するとTRUEに切り替わります。ErrIdには、コマンド固有のエラーコードが含まれています。ブロックにタイムアウトエラーが発生すると、ERRがTRUE、ERRIDが1861(16進数: 0x745)となります。次のコマンド実行により、FALSEにリセットされます。

ErrId: 最後に実行されたコマンドのADSエラーコード [▶ 119]またはコマンド固有のエラーコードです。次のコマンド実行により、0にリセットされます。

Quit: イベントが確認レスポンスされます。



上図は、基本的な構造を示しています。

必須ではない確認レスポンスメッセージでは、ファンクションブロックのEvent入力の立ち上がりでイベントがレポートされ、イベントがイベントロガーでも有効になります。Event入力の立ち下りエッジで、リセットがトリガされます。この信号により、イベントがキャンセルされ、イベントロガーにもこのキャンセルが報告されます。

確認必須のレスポンスメッセージでは、イベントがEvent入力の立ち上がりで再度有効になります。このイベントは以下の方法で無効化できます。

- Event入力の立ち下がりによって(その前に確認レスポンス信号がPLC、またはビジュアリゼーションからQuit入力を着信した場合)。
- Quit入力の立ち上がりによって(リセットがEvent入力の立ち下がりで開始されている場合)。

リセットがイベントの有効化と確認レスポンス発生の間に行われる場合、次のEvent入力の発生は「信号」と呼ばれます。この際、既に有効化されているイベントでリクエストがレポートされます。

STでのADSLOGEVENTファンクションブロックの使用例:

- PLCからのイベントログ信号の送信/確認レスポンス [▶ 113]

要件

開発環境	ターゲットシステム	PLCライブラリ(カテゴリグループ)
TwinCAT v3.1.0	PCまたはCX (x86、x64、ARM)	Tc2_System (システム)

3.5.2 ADSCLEAREVENTS

このファンクションブロックにより、TwinCATイベントロガーへのメッセージの送信および確認レスポンスが可能です。

VAR_INPUT

```
VAR_INPUT
    NETID    : T_AmsNetId;
    bClear   : BOOL;
    iMode    : UDINT;
    tTimeout : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
```

NETID: ADSコマンドの送信先であるターゲットデバイスのAMSネットワーク識別子の文字列です(型: T_AmsNetId [▶ 99]型)。

bClear: イベントが立ち上がりで削除されます。

iMode: イベントを削除する際のモードです。Enum型のE_TcEventClearModes [▶ 97]で定義されます。

tTimeout: ファンクションがキャンセルされるまでの時間を示します。

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy  : BOOL;
    bErr   : BOOL;
    iErrId : UDINT;
END_VAR
```

bBusy: ファンクションブロックが有効な場合、この出力がTRUEにセットされ、フィードバックを受信するまでセットされたままの状態となります。bBusyがTRUEである限り、新しいコマンドを実行できません。

bErr: この出力は、コマンド実行中にエラーが発生すると直ちにTRUEに切り替わります。iErrIdには、コマンド固有のエラーコードが含まれています。ファンクションブロックにタイムアウトエラーが発生すると、bErr がTRUE、iErrId が1861 (16進数: 0x745)となります。次のコマンド実行により、FALSEにリセットされます。

iErrId: 最後に実行されたコマンドのADSエラーコード [▶ 119]またはコマンド固有のエラーコードです。次のコマンド実行により、0にリセットされます。

要件

開発環境	ターゲットシステム	PLCライブラリ(カテゴリグループ)
TwinCAT v3.1.0	PCまたはCX (x86、x64、ARM)	Tc2_System (システム)

3.5.3 FB_SimpleAdsLogEvent

このファンクションブロックにより、TwinCATイベントロガーへのメッセージの送信および確認レスポンスが可能です。ADSLOGEVENTブロックとは異なり、FB_SimpleAdsLogEventブロックを使用してPLCからイベントをパラメータセットすることはできませんが、イベントを簡単な方法でセット、リセット、および確認レスポンスできます。

VAR_INPUT

```
VAR_INPUT
    SourceId    : INT;
    EventId     : INT;
    bSetEvent   : BOOL;
    bQuit       : BOOL;
END_VAR
```

SourceId: ソースのIDです。イベントロガーでソースを明確に識別するために使用します。

EventId: イベントのIDです。イベントロガーでイベントを明確に識別するために使用します。

bSetEvent: 立ち上がりではイベントの「発生」が、立ち下りではイベントの「終了」が信号送信されます。

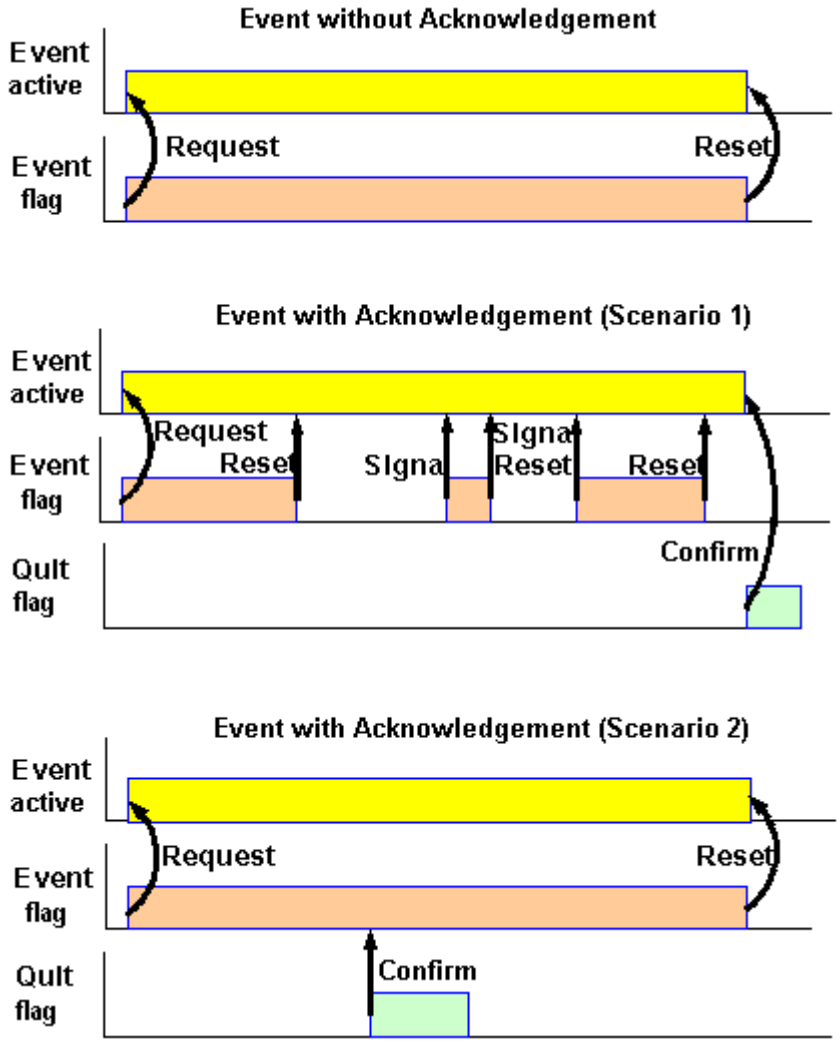
bQuit: 立ち上がりで、イベントが確認レスポンスされます。

VAR_OUTPUT

```
VAR_OUTPUT
    ErrId      : UDINT;
    Error      : BOOL;
END_VAR
```

ErrId: 最後に実行されたコマンドのADSエラーコード [▶ 119]またはコマンド固有のエラーコードです。次のコマンド実行により、0にリセットされます。

Error: この出力は、コマンド実行中にエラーが発生すると直ちにTRUEに切り替わります。ErrIdには、コマンド固有のエラーコードが含まれています。ブロックにタイムアウトエラーが発生すると、Error = TRUE、ErrId = 1861 (16進数: 0x745)となります。次のコマンド実行により、FALSEにリセットされます。



上図は、一般的なシーケンスを示しています。

確認必須ではないレスポンスメッセージでは、ファンクションブロックのEvent入力の立ち上がりでイベントがレポートされ、イベントがイベントロガーでも有効になります。Event入力の立ち下りで、リセットがトリガされます。この信号により、イベントがキャンセルされ、イベントロガーにもこのキャンセルが報告されます。

確認必須のレスポンスメッセージでは、イベントがEvent入力の立ち上がりで再度有効になります。このイベントは以下の方法で無効化できます。

- Event入力の立ち下がりによって(その前に確認レスポンス信号がPLC、またはビジュアリゼーションからQuit入力を着信した場合)。
- Quit入力の立ち上がりによって(その前にEvent入力の立ち下がりでリセットが実行されている場合)。

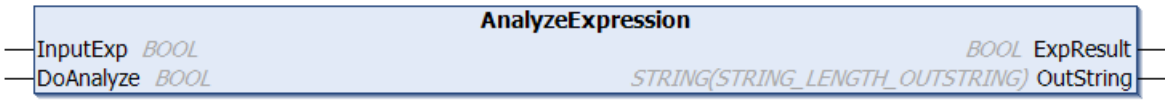
イベントのリセットが、イベントの有効化と確認レスポンスの受信の間に行われる場合、次のEvent入力の受信は「信号」と呼ばれます。このため、リクエストは既に有効なイベントに通知されます。

要件

開発環境	ターゲットシステム	PLCライブラリ(カテゴリグループ)
TwinCAT v3.1.0	PCまたはCX (x86、x64、ARM)	Tc2_System (システム)

3.6 IECステップ/SFCフラグファンクションブロック

3.6.1 AnalyzeExpression



このファンクションブロックは、SFCフラグを使用するPLCプロジェクトで使用できます。インスタンスは生成されません。対応するPLCライブラリが、プロジェクトに追加されている必要があります。その他の構成要件は、以下の説明に記載されています。

AnalyzeExpressionおよびAnalyzeExpressionTableファンクションブロックを使用して、遷移および歩進条件を分析できます。設定された時間が経過しても遷移がトリガされない場合、これらのファンクションブロックを使用して遷移を分析できます。

これらのファンクションブロックは、STプログラミング言語で実装されている式または遷移の分析にしか使用できません。

- AnalyzeExpression:
  - このファンクションブロックは分析結果、つまり切り替えが発生しなかった理由(どの部分的条件が満たされなかったか)を文字列に出力します。部分的条件を構成する変数は、演算子で互いに結合しています(例: bVar1 AND (bVar2 OR bVar3))。
  - 出力にはSFCフラグ「SFCErrorAnalyzation」が使用されます。
- AnalyzeExpressionTable:
  - このファンクションブロックは、切り替えが発生しなかったすべての変数を個別に出力します。個々の変数は、配列要素として記録または出力されます。各配列要素に対する情報には、変数の名前、アドレス、コメント、および現在値が含まれます。
  - 出力にはSFCフラグ「SFCErrorAnalyzationTable」が使用されます。

設定要件

SFCでAnalyzeExpressionまたはAnalyzeExpressionTableを有効にするには、以下の設定が必要です。

- PLCライブラリTc2_Systemをインクルードする。
- SFC-POUで以下の変数を宣言する:
  SFCEnableLimit: BOOL := TRUE;
- [Properties]ウィンドウで、後続の遷移/切り替え条件を分析するSFCダイアグラムのステップに対して最大有効時間を設定します(SFCエレメントプロパティも参照)。
- PLCプロパティまたはPOUプロパティで、SFCの設定を行います(SFCフラグおよびコマンドプロパティ(PLCプロジェクト) > カテゴリSFCも参照):

- [Flags]タブ:
  以下のSFCフラグの[Active]および[Declare]チェックボックスをチェックします:
  SFCError、SFCEnableLimit、SFCErrorAnalyzation、SFCErrorAnalyzationTable
- [Build]タブ:
  オプション[Calculate active transitions only]を有効にします。

サンプル

前述の設定は、以下のサンプルに実装されています。「Step1」では、最大有効時間が1秒に設定されています。関連付けられた発信側の遷移「Trans_ST」が1秒経過してもトリガされない場合、この遷移はファンクションブロックAnalyzeExpressionおよびAnalyzeExpressionTableによって分析されます。変数SFCErrorがTRUEにセットされ、変数SFCErrorStepに値「Step 1」が格納されます。

分析結果「SFCErrorAnalyzation」または「SFCErrorAnalyzationTable」が、式が(部分的に)まだトリガされていないことを示しているため、「Step1」から退出できます。

遷移「Trans_ST」は以下の表現で構成されています。

```
b1 AND (b2 OR b3);
```

- ケース1: 3つの変数b1、b2、b3のいずれもTRUEではない。
  - 「SFCErrorAnalyzation」は、分析結果「b1 AND (b2 OR b3)」を表示します。
  - 「SFCErrorAnalyzationTable」は、3つの変数b1、b2、b3すべてを詳細な変数情報と共に表示します。
  - 図1も参照してください。
- ケース2: 変数b1がTRUEにセットされている。分析は以下のように変化します。
  - 「SFCErrorAnalyzation」は、分析結果「(b2 OR b3)」を表示します。
  - 「SFCErrorAnalyzationTable」は、2つの変数b2およびb3を対応する変数情報と共に表示します。
  - 図2も参照してください。

図1:

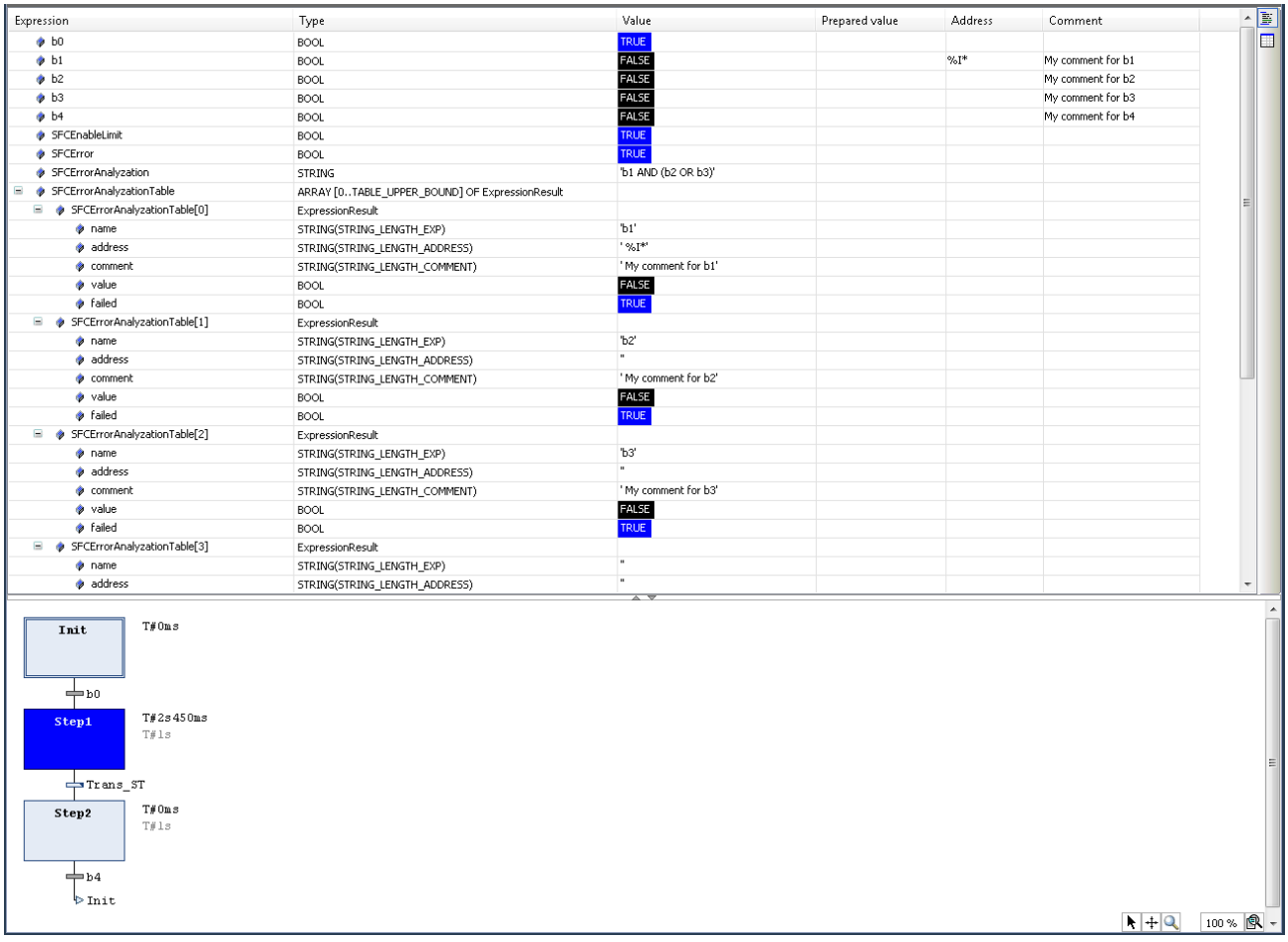
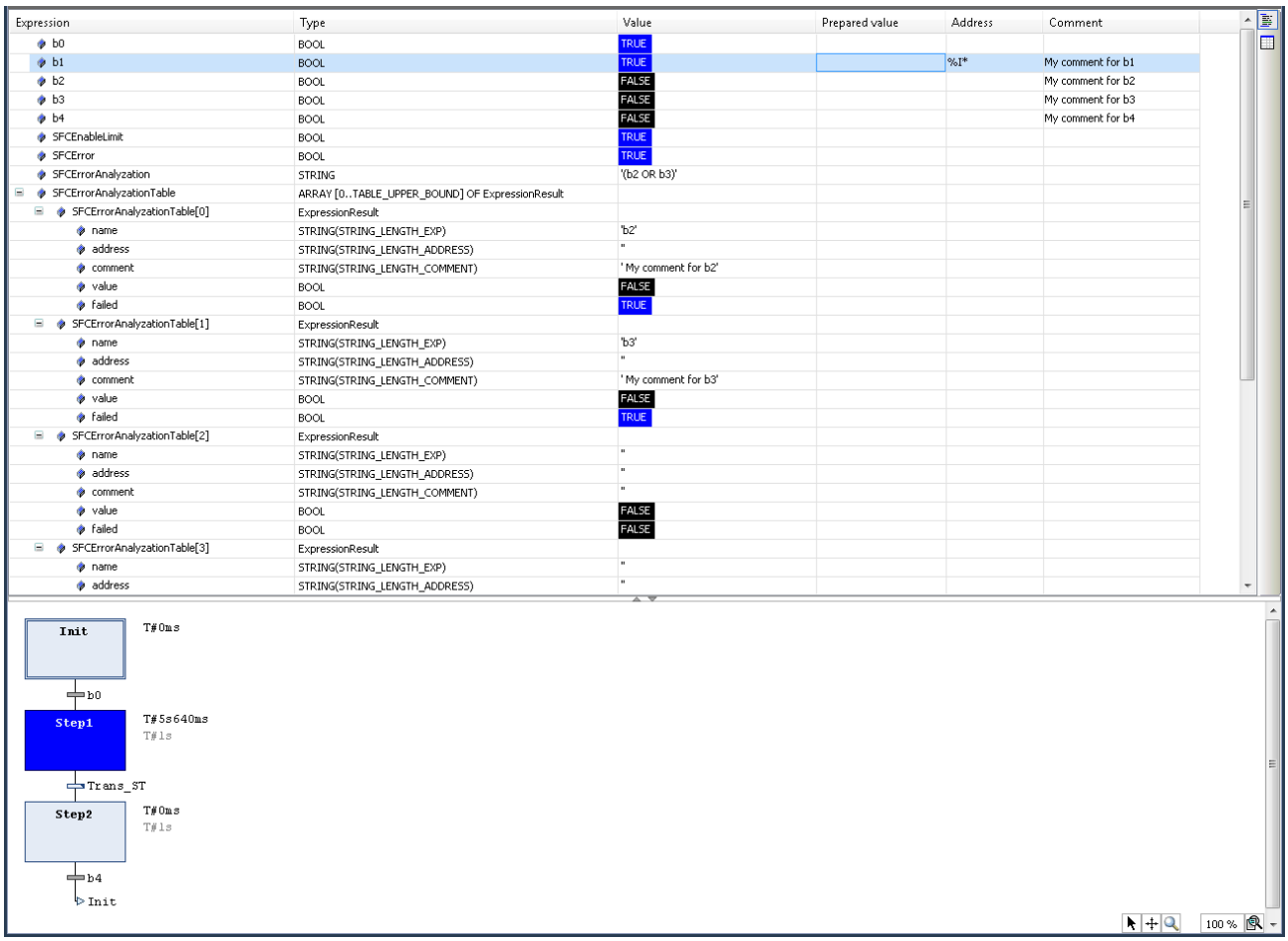


図2:



要件

開発環境	ターゲットシステム	PLCライブラリ(カテゴリグループ)
TwinCAT v3.1.0	PCまたはCX (x86、x64、ARM)	Tc2_System (システム)

3.6.2 AnalyzeExpressionTable

このファンクションブロックは、SFCフラグを使用するPLCプロジェクトで使用できます。インスタンスは生成されません。対応するPLCライブラリが、プロジェクトに追加されている必要があります。その他の構成要件は、以下の説明に記載されています。

AnalyzeExpressionおよびAnalyzeExpressionTableファンクションブロックを使用して、遷移および歩進条件を分析できます。設定された時間が経過しても遷移がトリガされない場合、これらのファンクションブロックを使用して遷移を分析できます。

これらのファンクションブロックは、STプログラミング言語で実装されている式または遷移の分析にしか使用できません。

- AnalyzeExpression:
  - このファンクションブロックは分析結果、つまり切り替えが発生しなかった理由(どの部分的条件が満たされなかったか)を文字列に出力します。部分的条件を構成する変数は、演算子で互いに結合しています(例: bVar1 AND (bVar2 OR bVar3))。
  - 出力にはSFCフラグ「SFCErrorAnalyzation」が使用されます。
- AnalyzeExpressionTable:

- このファンクションブロックは、切り替えが発生しなかったすべての変数を個別に出力します。個々の変数は、配列要素として記録または出力されます。各配列要素に対する情報には、変数の名前、アドレス、コメント、および現在値が含まれます。
- 出力にはSFCフラグ「SFCErrorAnalyzationTable」が使用されます。

設定要件

SFCでAnalyzeExpressionまたはAnalyzeExpressionTableを有効にするには、以下の設定が必要です。

- PLCライブラリTc2_Systemをインクルードする。
- SFC-POUで以下の変数を宣言する:
  SFCEnableLimit: BOOL := TRUE;
- [Properties]ウィンドウで、後続の遷移/切り替え条件を分析するSFCダイアグラムのステップに対して最大有効時間を設定します(SFCエレメントプロパティも参照)。
- PLCプロパティまたはPOUプロパティで、SFCの設定を行います(SFCフラグおよびコマンドプロパティ(PLCプロジェクト) > カテゴリSFCも参照):
  - [**Flags**]タブ:
    以下のSFCフラグの[**Active**]および[**Declare**]チェックボックスをチェックします:
    SFCError、SFCEnableLimit、SFCErrorAnalyzation、SFCErrorAnalyzationTable
  - [**Build**]タブ:
    オプション[**Calculate active transitions only**]を有効にします。

サンプル

前述の設定は、以下のサンプルに実装されています。「Step1」では、最大有効時間が1秒に設定されています。関連付けられた発信側の遷移「Trans_ST」が1秒経過してもトリガされない場合、この遷移はファンクションブロックAnalyzeExpressionおよびAnalyzeExpressionTableによって分析されます。変数SFCErrorがTRUEにセットされ、変数SFCErrorStepに値「Step 1」が格納されます。

分析結果「SFCErrorAnalyzation」または「SFCErrorAnalyzationTable」が、式が(部分的に)まだトリガされていないことを示しているため、「Step1」から退出できます。

遷移「Trans_ST」は以下の表現で構成されています。

```
b1 AND (b2 OR b3);
```

- ケース1: 3つの変数b1、b2、b3のいずれもTRUEではない。
  - 「SFCErrorAnalyzation」は、分析結果「b1 AND (b2 OR b3)」を表示します。
  - 「SFCErrorAnalyzationTable」は、3つの変数b1、b2、b3すべてを詳細な変数情報と共に表示します。
  - 図1も参照してください。
- ケース2: 変数b1がTRUEにセットされている。分析は以下のように変化します。
  - 「SFCErrorAnalyzation」は、分析結果「(b2 OR b3)」を表示します。
  - 「SFCErrorAnalyzationTable」は、2つの変数b2およびb3を対応する変数情報と共に表示します。
  - 図2も参照してください。

図1:

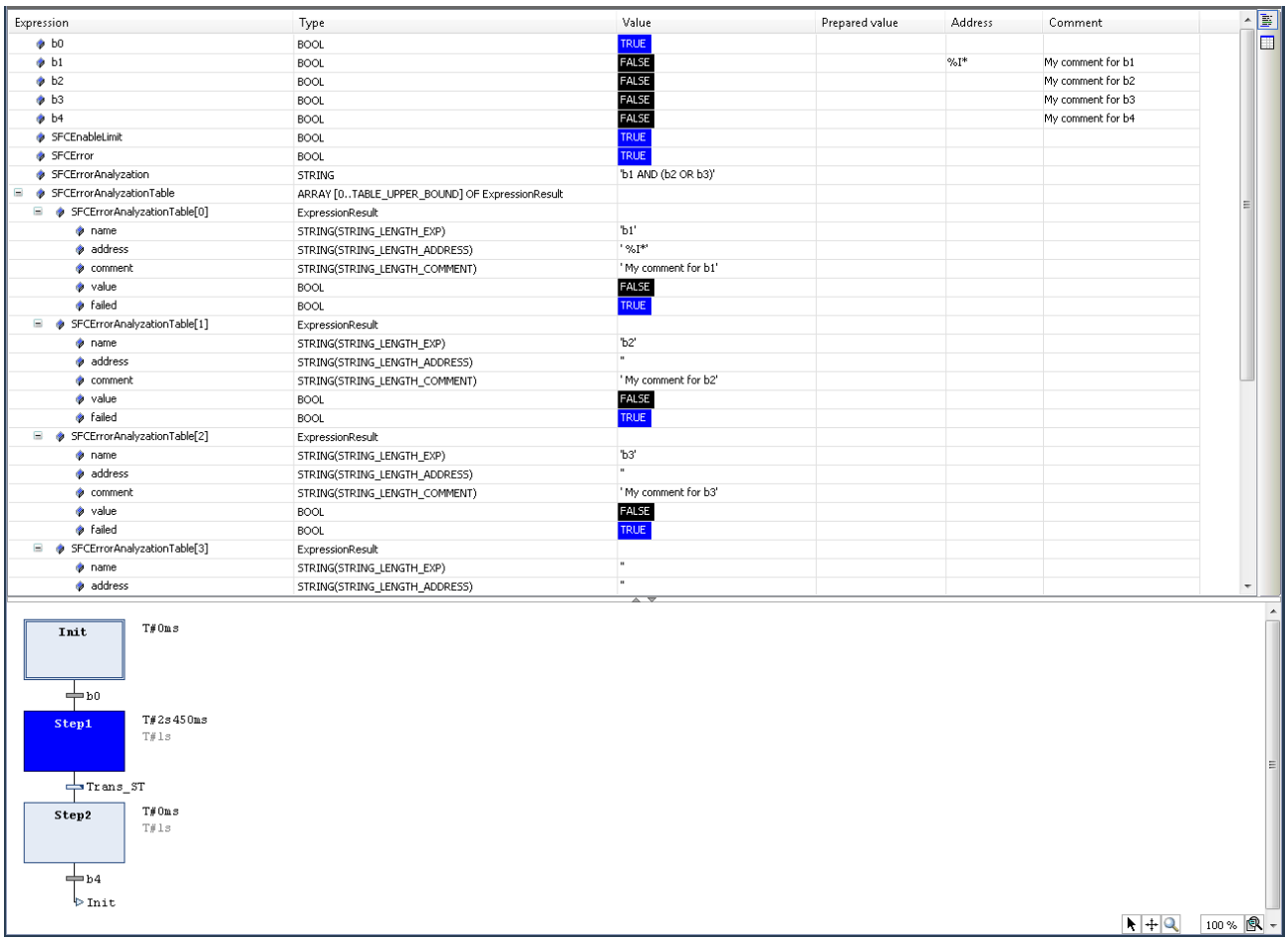
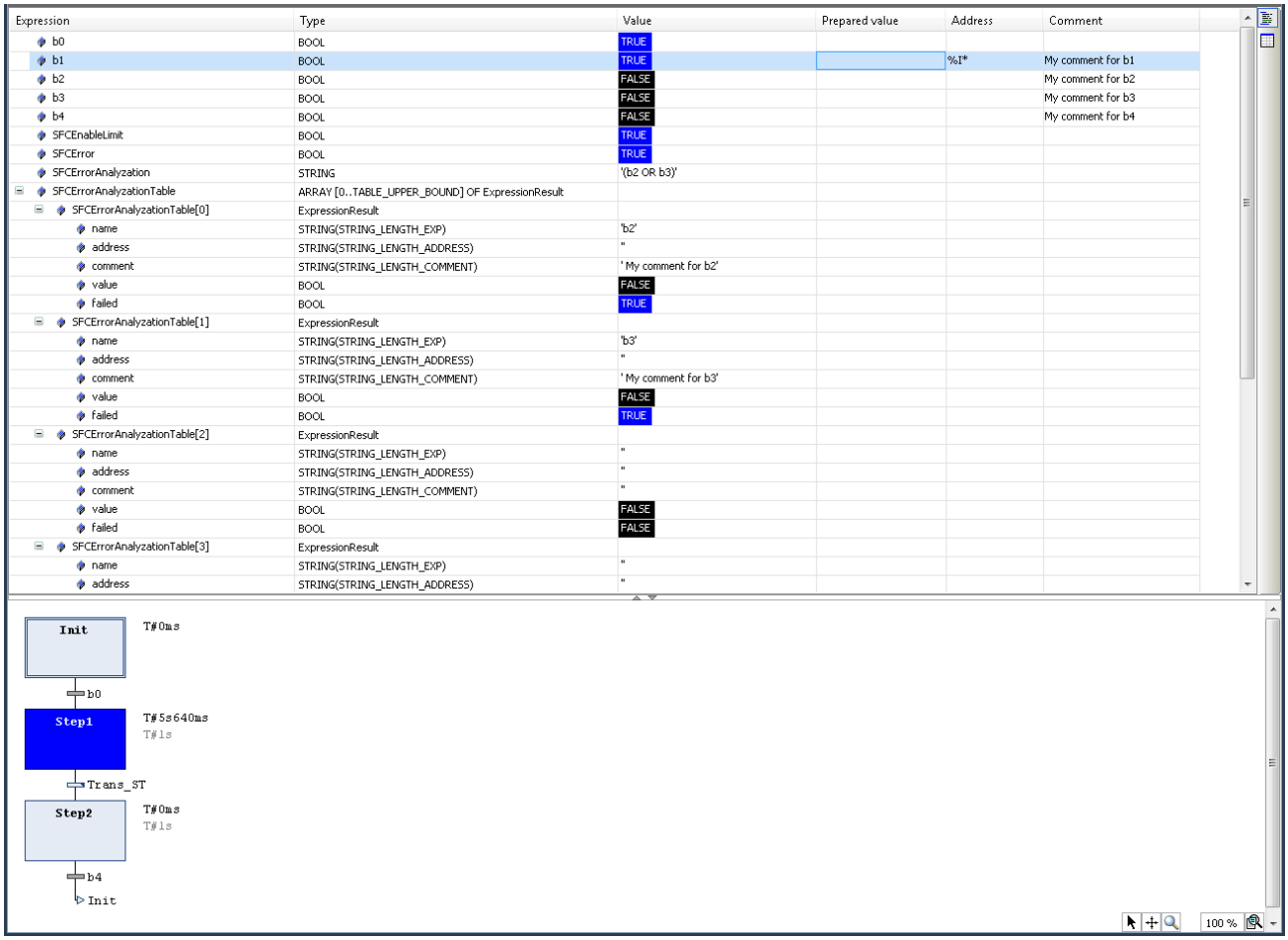

Expression
Type
Value
Prepared value
Address
Comment
b0 BOOL TRUE
b1 BOOL FALSE %I* My comment for b1
b2 BOOL FALSE My comment for b2
b3 BOOL FALSE My comment for b3
b4 BOOL FALSE My comment for b4
SFCEnableLimit BOOL TRUE
SFCError BOOL TRUE
SFCErrorAnalyzation STRING 'b1 AND (b2 OR b3)'
SFCErrorAnalyzationTable ARRAY [0..TABLE_UPPER_BOUND] OF ExpressionResult
SFCErrorAnalyzationTable[0] ExpressionResult
name STRING(STRING_LENGTH_EXP) 'b1'
address STRING(STRING_LENGTH_ADDRESS) '%I*'
comment STRING(STRING_LENGTH_COMMENT) ' My comment for b1'
value BOOL FALSE
failed BOOL TRUE
SFCErrorAnalyzationTable[1] ExpressionResult
name STRING(STRING_LENGTH_EXP) 'b2'
address STRING(STRING_LENGTH_ADDRESS) ''
comment STRING(STRING_LENGTH_COMMENT) ' My comment for b2'
value BOOL FALSE
failed BOOL TRUE
SFCErrorAnalyzationTable[2] ExpressionResult
name STRING(STRING_LENGTH_EXP) 'b3'
address STRING(STRING_LENGTH_ADDRESS) ''
comment STRING(STRING_LENGTH_COMMENT) ' My comment for b3'
value BOOL FALSE
failed BOOL TRUE
SFCErrorAnalyzationTable[3] ExpressionResult
name STRING(STRING_LENGTH_EXP) ''
address STRING(STRING_LENGTH_ADDRESS) ''
Init
T#0ms
b0
Step1
T#2s450ms
T#1s
Trans_ST
Step2
T#0ms
T#1s
b4
Init
100 %

図2:


Expression
Type
Value
Prepared value
Address
Comment
b0 BOOL TRUE
b1 BOOL TRUE %I* My comment for b1
b2 BOOL FALSE My comment for b2
b3 BOOL FALSE My comment for b3
b4 BOOL FALSE My comment for b4
SFCEnableLimit BOOL TRUE
SFCError BOOL TRUE
SFCErrorAnalyzation STRING '(b2 OR b3)'
SFCErrorAnalyzationTable ARRAY [0..TABLE_UPPER_BOUND] OF ExpressionResult
SFCErrorAnalyzationTable[0] ExpressionResult
name STRING(STRING_LENGTH_EXP) 'b2'
address STRING(STRING_LENGTH_ADDRESS) ''
comment STRING(STRING_LENGTH_COMMENT) ' My comment for b2'
value BOOL FALSE
failed BOOL TRUE
SFCErrorAnalyzationTable[1] ExpressionResult
name STRING(STRING_LENGTH_EXP) 'b3'
address STRING(STRING_LENGTH_ADDRESS) ''
comment STRING(STRING_LENGTH_COMMENT) ' My comment for b3'
value BOOL FALSE
failed BOOL TRUE
SFCErrorAnalyzationTable[2] ExpressionResult
name STRING(STRING_LENGTH_EXP) ''
address STRING(STRING_LENGTH_ADDRESS) ''
comment STRING(STRING_LENGTH_COMMENT) ''
value BOOL FALSE
failed BOOL FALSE
SFCErrorAnalyzationTable[3] ExpressionResult
name STRING(STRING_LENGTH_EXP) ''
address STRING(STRING_LENGTH_ADDRESS) ''
Init
T#0ms
b0
Step1
T#5s640ms
T#1s
Trans_ST
Step2
T#0ms
T#1s
b4
Init
100 %

要件

開発環境	ターゲットシステム	PLCライブラリ(カテゴリグループ)
TwinCAT v3.1.0	PCまたはCX (x86、x64、ARM)	Tc2_System (システム)

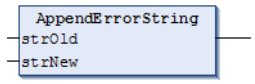
3.6.3 AnalyzeExpressionCombined

このファンクションブロックは、SFCフラグを使用するPLCプロジェクトで必要となります。インスタンスは生成されません。対応するPLCライブラリが、プロジェクトに含まれている必要があります。

要件

開発環境	ターゲットシステム	PLCライブラリ(カテゴリグループ)
TwinCAT v3.1.0	PCまたはCX (x86、x64、ARM)	Tc2_System (システム)

3.6.4 AppendErrorString

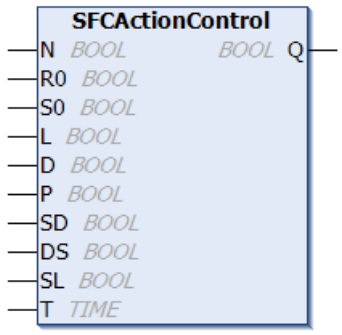


このファンクションは、SFCフラグを使用するPLCプロジェクトで必要となります。このファンクションはプロジェクトでコールしてはいけません。対応するPLCライブラリのみが、プロジェクトに含まれている必要があります。

要件

開発環境	ターゲットシステム	PLCライブラリ(カテゴリグループ)
TwinCAT v3.1.0	PCまたはCX (x86、x64、ARM)	Tc2_System (システム)

3.6.5 SFCActionControl



このファンクションは、SFCプログラム/プロジェクトでIECステップを使用する際に必要となります。FBを含むライブラリのみがプロジェクトに含まれている必要がありますが、インスタンスは不要です。

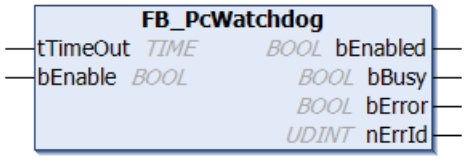
要件

開発環境	ターゲットシステム	PLCライブラリ(カテゴリグループ)
TwinCAT v3.1.0	PCまたはCX (x86、x64、ARM)	Tc2_System (システム)

3.7 ウォッチドッグファンクションブロック

3.7.1 FB_PcWatchdog

この機能は、次のメインボードを実装したIPCでのみ使用可能です: IP-4GVI63、CB1050、CB2050、CB3050、CB1051、CB2051、CB3051。



ファンクションブロックFB_PcWatchdogを使用すると、PCのハードウェアウォッチドッグを有効にすることができます。ウォッチドッグは、bEnable = TRUEおよびタイムアウト時間によって有効になります。タイムアウト時間の範囲は1～255秒です。ウォッチドッグは、bEnable = TRUEおよびtTimeOut >= 1秒という設定によって有効になります。

ウォッチドッグを有効化した場合は、tTimeOut経過時にPCが自動的に再起動されるため、ファンクションブロックをtTimeOutよりも短い間隔で周期的に呼び出す必要があります。このため、ウォッチドッグを使用すると、無限ループが生じたシステムや、PLCがフリーズしたシステムを自動的に再起動することができます。

ウォッチドッグは、bEnable = FALSEまたはtTimeOut = 0によって無効化できます。

注記

PC再起動

タイムアウト時間が経過するとPCが直ちに再起動するため、ブレークポイントを使用する前、PLCまたは全体をリセットする前、TwinCATを停止する前、構成を有効化する前にウォッチドッグを無効化する必要があります。

VAR_INPUT

```
VAR_INPUT
    tTimeOut : TIME;
    bEnable  : BOOL;
END_VAR
```

tTimeOut: 再起動が実行されるまでのウォッチドッグ時間です。

bEnable: ウォッチドッグを有効または無効にします。

VAR_OUTPUT

```
VAR_OUTPUT
    bEnabled : BOOL;
    bBusy    : BOOL;
    bError   : BOOL;
    nErrId   : UDINT;
END_VAR
```

bEnabled: TRUE = ウォッチドッグが有効、FALSE = ウォッチドッグが無効

bBusy: この出力は、ファンクションブロックがコマンドを実行するまでTRUEのままとなります。

bError: この出力は、コマンド実行中にエラーが発生すると直ちにTRUEに切り替わります。nErrIdには、コマンド固有のエラーコードが含まれています。次のコマンド実行により、FALSEにリセットされます。

nErrId: 最後に実行されたコマンドのADSエラーコード [▶ 119]またはコマンド固有のエラーコードです。次のコマンド実行により、0にリセットされます。

STでのファンクションブロックコールの例:

```
PROGRAM MAIN
VAR
    fbPcWatchDog  : FB_PcWatchdog;
```

```
    tWDTime       : TIME := T#10s;
    bEnableWD     : BOOL;
    bWDActive     : BOOL;
END_VAR

IF bEnableWD OR bWDActive THEN
    fbPcWatchDog(tTimeOut := tWDTime, bEnable := bEnableWD);
    bWDActive := fbPcWatchDog.bEnabled;
END_IF
```

要件

開発環境	ターゲットシステム	PLCライブラリ(カテゴリグループ)
TwinCAT v3.1.0	次のメインボードを実装したIPC: IP-4GVI63、CB1050、CB2050、CB3050、CB1051、CB2051、CB3051	PLC Lib Tc2_System

3.7.2 FB_PcWatchDog_BAPI

この機能は、次のメインボードを実装したIPCでのみ使用可能です: BIOSバージョンが0.44以降のCBxx63。

ファンクションブロックFB_PcWatchdog_BAPIを使用すると、PCのハードウェアウォッチドッグを有効にすることができます。ウォッチドッグは、bExecute = TRUEおよびウォッチドッグ時間によって有効になります。ウォッチドッグ時間の範囲は、1～15,300秒(255分)です。ウォッチドッグは、bEnable = TRUEおよびnWatchdogTimeS >= 1秒という設定によって有効になります。

ウォッチドッグを有効化した場合は、nWatchdogTimeS経過時にPCが自動的に再起動されるため、ファンクションブロックをnWatchdogTimeSよりも短い間隔で周期的に呼び出す必要があります。このため、ウォッチドッグを使用すると、無限ループが生じたシステムや、PLCがフリーズしたシステムを自動的に再起動することができます。

注記

PC再起動

nWatchdogTimeSが経過するとPCが直ちに再起動するため、ブレークポイントを使用する前、PLCまたは全体をリセットする前、TwinCATを停止する前、構成を有効化する前にウォッチドッグを無効化する必要があります。

VAR_INPUT

```
VAR_INPUT
    sNetID         : T_AmsNetID;
    nWatchdogTimeS : UDINT;
    bExecute       : BOOL;
    tTimeout       : TIME;
END_VAR
```

sNetID: デバイスのAMSネットワークID(空文字列またはローカルネットワークID)。

nWatchdogTimeS: ウォッチドッグ時間(単位: 秒)、0 = 無効、>0 (最大15300) = 有効。

bExecute: コマンドが立ち上がりで実行されます。

tTimeout: 内部的なADS通信が終了するまでの時間を示します。

VAR_OUTPUT

```
VAR_OUTPUT
    bEnabled : BOOL;
    bBusy    : BOOL;
    bError   : BOOL;
    nErrorId : UDINT;
END_VAR
```

bEnabled: TRUE = ウォッチドッグが有効、FALSE = ウォッチドッグが無効

bBusy: この出力は、ファンクションブロックがコマンドを実行するまでTRUEのままとなります。

bError: この出力は、コマンド実行中にエラーが発生すると直ちにTRUEに切り替わります。入力でコマンドが実行されると、FALSEにリセットされます。

nErrorId: 最後に実行されたコマンドのADSエラーコード [▶ 119]またはコマンド固有のエラーコードです。次のコマンド実行により、0にリセットされます。

STでのファンクションブロックコールの例:

```
PROGRAM MAIN
VAR
    fbWatchdog     : FB_PcWatchdog_BAPI;
    nWatchdogTimeS : UDINT := 10;  (* 10s *)
    bEnabled       : BOOL; (* TRUE: watchdog is activated *)
    bError         : BOOL;
    nErrID         : UDINT;
    fbTimer        : TON := (IN := TRUE, PT := T#0S);
END_VAR
```

```
fbTimer();

(* 1st enable, then refresh watchdog every 1s *)
IF fbTimer.Q THEN
    fbWatchdog(
        sNetID         := '',
        nWatchdogTimeS := nWatchdogTimeS,
        bExecute       := TRUE,
        tTimeout       := T#5S,
    );

    IF NOT fbWatchdog.bBusy THEN
        bEnabled   := fbWatchdog.bEnabled;
        bError     := fbWatchdog.bError;
        nErrID     := fbWatchdog.nErrID;

        fbWatchdog(bExecute := FALSE);

        (* restart timer*)
        fbTimer(IN := FALSE);
        fbTimer(IN := TRUE, PT := T#1S); (* refresh watchdog every s *)
    END_IF
END_IF
```

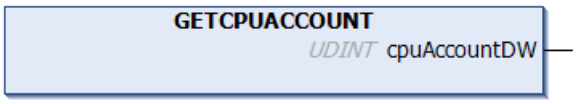
要件

開発環境	ターゲットシステム	PLCライブラリ(カテゴリグループ)
TwinCAT v3.1.0	CBxx63メインボードおよびバージョン0.44以降のBIOSを実装したIPC	PLC Lib Tc2_System 3.4.14.0以降

3.8 タイムファンクションブロック

3.8.1 GETCPUACCOUNT

この機能は、Windows CEのPLCランタイムシステムでは使用できません。



このファンクションブロックを使用して、PLCタスクサイクルティッカを読み込みできます。PLCタスクサイクルティッカは、タスクの実行中のみインクリメントされます。この数値は32ビット整数であり、CPUの内部クロックレートに依存せずに100 nsティックに変換された形式で出力されます。この数値はPLCタスクが呼び出されるたびに100 nsの精度で更新され、たとえばタイミングを計る場合などに使用できます。1単位は100 nsです。

VAR_INPUT

```
VAR_INPUT
(*none*)
END_VAR
```

VAR_OUTPUT

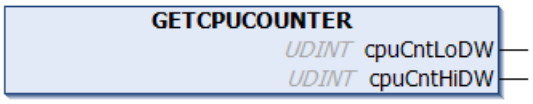
```
VAR_OUTPUT
    cpuAccountDW : UDINT;
END_VAR
```

cpuAccountDW: PLCタスクティッカの現在値です。

要件

開発環境	ターゲットシステム	PLCライブラリ
TwinCAT v3.1.0	PCまたはCX（x86、x64）	Tc2_System（システム）

3.8.2 GETCPUCOUNTER



このファンクションブロックを使用して、CPUサイクルカウンタを読み込みできます。この数値は相対64ビット整数であり、CPUの内部クロックレートに依存せずに100 nsティックに変換された形式で出力されます。この数値はPLCシステムによるコールのたびに100 nsの精度で更新され、たとえばタイミングを計る場合などに使用できます。1単位は100 nsです。このサービスがファンクションではなくブロックとして実装されるのは、単に2つの値を返す必要があるためです。これは、原則としてファンクションでは行えません。

VAR_INPUT

```
VAR_INPUT
(*none*)
END_VAR
```

VAR_OUTPUT

```
VAR_OUTPUT
    cpuCntLoDW : UDINT;
    cpuCntHiDW : UDINT;
END_VAR
```

cpuCntLoDW: 数値の下位4バイトです。

cpuCntHiDW: 数値の上位4バイトです。

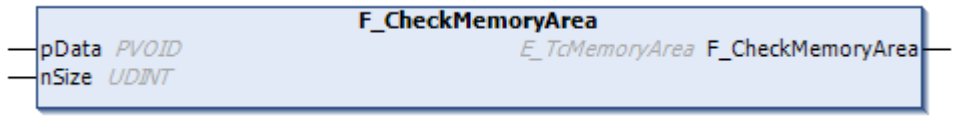
要件

開発環境	ターゲットシステム	PLCライブラリ(カテゴリグループ)
TwinCAT v3.1.0	PCまたはCX (x86、x64、ARM)	Tc2_System (システム)

4 ファンクション

4.1 汎用ファンクション

4.1.1 F_CheckMemoryArea



このファンクションは、指定されたサイズのリクエストされた変数が位置するメモリ領域に関する情報を返します。これには、型がE_TcMemoryArea [▶ 98]の戻り値が使用されます。

FUNCTION F_CheckMemoryArea: E_TcMemoryArea

```
VAR_INPUT
    pData : PVOID;
    nSize : UDINT;
END_VAR
```

pData: 変数のメモリアドレス

nSize: バイト単位での変数サイズ

例

```
PROGRAM MAIN
VAR
    nCounter : USINT;
    eMemAreaStatic : E_TcMemoryArea;
    pDynamicVariable : POINTER TO LREAL;
    eMemAreaDynamic : E_TcMemoryArea;
    pNull : PVOID := 0;
    eMemAreaUnknown : E_TcMemoryArea;
END_VAR
-----------------------------------------------------------------------

nCounter := nCounter + 1;
eMemAreaStatic := F_CheckMemoryArea( pData:=ADR(nCounter), nSize:=SIZEOF(nCounter) );

IF nCounter = 100 THEN
    pDynamicVariable := __NEW(LREAL);
    IF pDynamicVariable <> 0 THEN
        pDynamicVariable^ := 7 * 4.5;
        eMemAreaDynamic := F_CheckMemoryArea( pData:=pDynamicVariable, nSize:=SIZEOF(LREAL) );
        __DELETE(pDynamicVariable);
    END_IF
END_IF

eMemAreaUnknown := F_CheckMemoryArea( pData:=pNull, nSize:=1 );
```

開発環境	ターゲットシステム	PLCライブラリ(カテゴリグループ)
TwinCAT v3.1.4022	PCまたはCX (x86、x64、ARM)	Tc2_System (システム)

4.1.2 F_CmpLibVersion

ファンクションF_CmpLibVersionは、既存のライブラリのバージョンと要求したバージョンを比較します。各ライブラリには、型がST_LibVersionの定数として個々のバージョン情報が付与されています。「stLibVersion_ライブラリ名」がこの定数名の形式です。

FUNCTION F_CmpLibVersion: DINT

```
VAR_INPUT
    stVersion   : ST_LibVersion;
    iMajor      : UINT;
    iMinor      : UINT;
    iBuild      : UINT;
    iRevision   : UINT;
END_VAR
```

stVersion: 既存のライブラリのバージョンです(型: ST_LibVersion型)。

iMajor: 比較するメジャー番号です。

iMinor: 比較するマイナー番号です。

iBuild: 比較するビルド番号です。

iRevision: 比較するリビジョン番号です。

戻りパラメータ	バージョンの相関関係
-1	現在のバージョンが要求しているバージョンよりも古いものである
0	現在のバージョンが要求しているバージョンである
+1	現在のバージョンが要求しているバージョンよりも新しいものである

STの例:

```
IF F_CmpLibVersion( stLibVersion_Tc2_System, 3, 3, 8, 0) >= 0 THEN
    (* newer lib ...*)
ELSE
    (* older lib... *)
END_IF
```

要件

開発環境	ターゲットシステム	PLCライブラリ(カテゴリグループ)
TwinCAT v3.1.0	PCまたはCX (x86、x64、ARM)	Tc2_System (システム)

4.1.3 F_CreateIPv4Addr

このファンクションは、フォーマットされた(IPv4)インターネットプロトコルネットワークアドレスを生成し、文字列型のパラメータとして返します(172.16.7.199など)。

FUNCTION F_CreateIPv4Addr : T_IPv4Addr

```
VAR_INPUT
    nIds : T_IPv4AddrArr;
END_VAR
```

nIds: バイト列です。各バイトが、(IPv4)インターネットプロトコルネットワークアドレスの1つの数値を表します。アドレスバイトは、ネットワークバイトオーダで表されます(型: T_IPv4AddrArr [▶ 101]型)。

ストラクチャードテキストの例:

```
PROGRAM MAIN
VAR
    ids   : T_IPv4AddrArr := 172, 16, 7, 199;
    sIPv4 : T_IPv4Addr := '';
END_VAR

sIPv4 := F_CreateIPv4Addr( ids );(* Result: '172.16.7.199' *)
```

要件

開発環境	ターゲットシステム	PLCライブラリ(カテゴリグループ)
TwinCAT v3.1.0	PCまたはCX (x86、x64、ARM)	Tc2_System (システム)

4.1.4 F_ScanIPv4AddrIds

ファンクションF_ScanIPv4AddrIdsは、(IPv4)インターネットプロトコルネットワークアドレスの文字列を単一のアドレスバイトに変換します。単一のアドレスバイトは、左から右に変換されます。これらのバイトは、バイト配列として返されます。アドレスバイトは、ネットワークバイトオーダで表されます。

FUNCTION F_ScanIPv4AddrIds: T_IPv4AddrArr

```
VAR_INPUT
    sIPv4 : T_IPv4Addr;
END_VAR
```

sIPv4: 文字列としてのインターネットプロトコルネットワークアドレスです(型: T_IPv4Addr [▶ 101]型)。例: 172.16.7.199

入力値	戻り値	説明
sIPv4 ≠ '' (空文字列) およびsIPv4 ≠ '0.0.0.0'	すべてのバイトがゼロ	変換中のエラー。sIPv4入力文字列のフォーマットをチェックしてください。

ストラクチャードテキストの例:

インターネットプロトコル(IPv4)ネットワークアドレス文字列「172.16.7.199」が、アドレスバイト配列に変換されます。

```
PROGRAM MAIN
VAR
    ids    : T_IPv4AddrArr;
    sIPv4  : T_IPv4Addr := '172.16.7.199';
END_VAR

ids := F_ScanIPv4AddrIds( sIPv4 ); (* Result: ids[0]:=172, ids[1]:=16, ids[2]:=7, ids[3]:=199 *)
```

要件

開発環境	ターゲットシステム	PLCライブラリ(カテゴリグループ)
TwinCAT v3.1.0	PCまたはCX (x86、x64、ARM)	Tc2_System (システム)

4.1.5 F_GetMappingPartner

ファンクションF_GetMappingPartnerは、マッピングのパートナ側のオブジェクトID（データ型: OTCID)を返します。

FUNCTION F_GetMappingPartner: OTCID

```
VAR_INPUT
    p : PVOID;
    n : UDINT;
END_VAR
```

p: 変数のメモリアドレス

n: バイト単位での変数サイズ

開発環境	ターゲットシステム	PLCライブラリ
TwinCAT v3.1.4020	PCまたはCX（x86、x64、ARM)	Tc2_System（システム)

4.1.6 F_GetMappingStatus

ファンクションF_GetMappingStatusは、PLC変数の現在のマッピングステータスを返します。このファンクションは、ENUM値(データ型: EPlcMappingStatus [▶ 99]型)と共に、値MS_Unmapped、MS_Mapped、またはMS_Partialを返します。

FUNCTION F_GetMappingStatus: EPlcMappingStatus

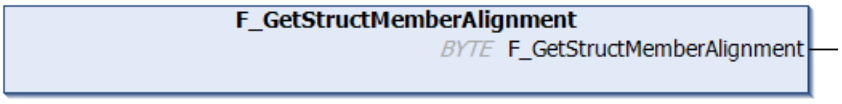
```
VAR_INPUT
    p : PVOID;
    n : UDINT;
END_VAR
```

p: 変数のメモリアドレス

n: バイト単位での変数サイズ

開発環境	ターゲットシステム	PLCライブラリ
TwinCAT v3.1.4020	PCまたはCX（x86、x64、ARM)	Tc2_System（システム)

4.1.7 F_GetStructMemberAlignment



このファンクションは、使用されているデータ構造メンバのアライメント設定に関する情報を返します。アライメントは、コンピュータメモリでのデータ構造要素の配置方法に影響を与えます。

FUNCTION F_GetStructMemberAlignment : BYTE

```
VAR_INPUT
    (* no input parameter *)
END_VAR
```

戻り値	説明
1	1バイトアラインメント(TwinCAT v2.11、x86ターゲットシステムなど)
2	2バイトアラインメント
4	4バイトアラインメント(TwinCAT v2.11、ARMターゲットシステムなど)
8	8バイトアラインメント

以下の例は、適用されるメモリアライメントごとの、データ構造要素の配置を示しています。

?? := パディングバイト

例1

```
TYPE ST_TEST1
STRUCT
    ui8 : BYTE := 16#FF;(* FF *)
    f64 : LREAL := 1234.5678;(* AD FA 5C 6D 45 4A 93 40 *)
END_STRUCT
END_TYPE

test1 : ST_TEST1;
```

アライメント	SIZEOF(test1)	メモリの内容
1 バイト	9	FF AD FA 5C 6D 45 4A 93 40
2 バイト	10	FF ?? AD FA 5C 6D 45 4A 93 40
4 バイト	12	FF ?? ?? ?? AD FA 5C 6D 45 4A 93 40
8 バイト	16	FF ?? ?? ?? ?? ?? ?? ?? AD FA 5C 6D 45 4A 93 40

例2

構造要素の順序を変換すると、パディングバイトの配置が変化します。この例では、パディングバイトが終端に追加されます。

```
TYPE ST_TEST2
STRUCT
    f64 : LREAL := 1234.5678;(* AD FA 5C 6D 45 4A 93 40 *)
    ui8 : BYTE := 16#FF;(* FF *)
END_STRUCT
END_TYPE

test2 : ST_TEST2;
```

アライメント	SIZEOF(test2)	メモリの内容
1 バイト	9	AD FA 5C 6D 45 4A 93 40 FF
2 バイト	10	AD FA 5C 6D 45 4A 93 40 FF ??
4 バイト	12	AD FA 5C 6D 45 4A 93 40 FF ?? ?? ??
8 バイト	16	AD FA 5C 6D 45 4A 93 40 FF ?? ?? ?? ?? ?? ?? ?? ??

例3

2、4、および8バイトアライメントの場合、要素ui32およびf64は既に適切にアライメントされているため、パディングバイトを追加する必要はありません。

```
TYPE ST_TEST3
STRUCT
    ui8  : BYTE := 16#FF;(* FF *)
    ui16 : WORD := 16#1234;(* 34 12 *)
    ui32 : DWORD := 16#AABBCCDD;(* DD CC BB AA *)
    f64  : LREAL := 1234.5678;(* AD FA 5C 6D 45 4A 93 40 *)
END_STRUCT
END_TYPE

test3 : ST_TEST3;
```

アライメント	SIZEOF(test3)	メモリの内容
1 バイト	15	FF 34 12 DD CC BB AA AD FA 5C 6D 45 4A 93 40
2 バイト	16	FF ?? 34 12 DD CC BB AA AD FA 5C 6D 45 4A 93 40
4 バイト	16	FF ?? 34 12 DD CC BB AA AD FA 5C 6D 45 4A 93 40
8 バイト	16	FF ?? 34 12 DD CC BB AA AD FA 5C 6D 45 4A 93 40

例4

```
TYPE ST_A1
STRUCT
    ui8  : BYTE := 16#FF;(* FF *)
    ui32 : DWORD := 16#AABBCCDD;(* DD CC BB AA *)
    rsv  : BYTE := 16#EE;(* EE *)
END_STRUCT
END_TYPE

TYPE ST_A2
STRUCT
    ui16 : WORD := 16#1234;(* 34 12 *)
    ui8  : BYTE := 16#55;(* 55 *)
END_STRUCT
END_TYPE

TYPE ST_TEST4
STRUCT
    a1 : ST_A1;
    a2 : ST_A2;
END_STRUCT
END_TYPE

test4 : ST_TEST4;
```

アライメント	SIZEOF(test4)	SIZEOF(test4.a1)	a1/a2パディングバイト	SIZEOF(test4.a2)	メモリの内容
1 バイト	9	6	-	3	FF DD CC BB AA EE 34 12 55
2 バイト	12	8	-	4	FF ?? DD CC BB AA EE ?? 34 12 55 ??
4 バイト	16	12	-	4	FF ?? ?? ?? DD CC BB AA EE ?? ?? ?? 34 12 55 ??
8 バイト	16	12	-	4	FF ?? ?? ?? DD CC BB AA EE ?? ?? ?? 34 12 55 ??

例5

```
TYPE ST_D1
STRUCT
    ui16 : WORD := 16#1234;(* 34 12 *)
    ui8 : BYTE := 16#55;(* 55 *)
END_STRUCT
END_TYPE

TYPE ST_D2
STRUCT
    ui8 : BYTE := 16#FF;(* FF *)
    f64 : LREAL := 1234.5678;(* AD FA 5C 6D 45 4A 93 40 *)
    rsv : BYTE := 16#EE;(* EE *)
END_STRUCT
```

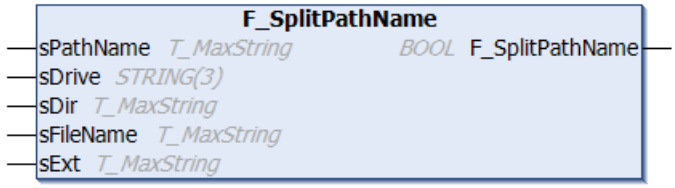
```
END_TYPE

TYPE ST_TEST5
STRUCT
    d1 : ST_D1;
    d2 : ST_D2;
END_STRUCT
END_TYPE

test5 : ST_TEST5;
```

アライメント	SIZEOF(test5)	SIZEOF(test5.d1)	d1/d2パディングバイト	SIZEOF(test5.d2)	メモリの内容
1 バイト	13	3	-	10	34 12 55 FF AD FA 5C 6D 45 4A 93 40 EE
2 バイト	16	4	-	12	34 12 55 ?? FF ?? AD FA 5C 6D 45 4A 93 40 EE ??
4 バイト	20	4	-	16	34 12 55 ?? FF ?? ?? ?? AD FA 5C 6D 45 4A 93 40 EE ?? ?? ??
8 バイト	32	4	4	24	34 12 55 ?? ?? ?? ?? ?? FF ?? ?? ?? ?? ?? ?? AD FA 5C 6D 45 4A 93 40 EE ?? ?? ?? ?? ?? ?? ??

要件

開発環境	ターゲットシステム	PLCライブラリ(カテゴリグループ)
TwinCAT v3.1.0	PCまたはCX (x86、x64、ARM)	Tc2_System (システム)

4.1.8 F_SplitPathName



このファンクションは、完全なパス名を4つの部分に分割します。分割された部分は、sDrive、sDir、sFileName、およびsExtという名前の文字列変数に格納されます。

FUNCTION F_SplitPathName : BOOL

```
VAR_INPUT
    sPathName : T_MaxString;
END_VAR
```

sPathName: 「X:¥DIR¥SUBDIR¥FILENAME.EXT」という形式の文字列としての完全なファイル名(型: T_MaxString [▶ 102]型)。

```
VAR_IN_OUT
    sDrive    : STRING(3);
    sDir      : T_MaxString;
```

```
    sFileName : T_MaxString;
    sExt      : T_MaxString;
END_VAR
```

sDrive: コロン付き(「C:」、「A:」など)のドライブID (型: T_MaxString [▶ 102]型)。

sDir: 前後のバックスラッシュを含む(「¥BC ¥INCLUDE¥」、「¥SOURCE¥」など)ディレクトリ名(型: T_MaxString [▶ 102]型)。

sFileName: ファイル名です(型: T_MaxString [▶ 102]型)。

sExt: ドットおよびファイル拡張子(型: T_MaxString [▶ 102]型) (例: 「.C」、「.EXE」など)。

戻りパラメータ	説明
TRUE	エラーなし
FALSE	エラーです。ファンクションのパラメータをチェックしてください。

STでのコールの例:

パス名「C:¥TwinCAT¥Plc¥Project01¥Data.txt」が、以下の部分に分割されます。

sDrive: = 「C:」
sDir: 「¥TwinCAT¥Plc¥Project01¥」
sFileName: 「Data」
sExt: 「.txt」

```
PROGRAM MAIN
VAR
    bSplit     : BOOL;
    sPathName  : T_MaxString := 'C:\TwinCAT\Plc\Project01\Data.txt';
    sDrive     : STRING(3);
    sDir       : T_MaxString;
    sFileName  : T_MaxString;
    sExt       : T_MaxString;
    bSuccess   : BOOL;
END_VAR

IF bSplit THEN
    bSplit := FALSE;
    bSuccess := F_SplitPathName( sPathName := sPathName,
                sDrive := sDrive,
                sDir := sDir,
                sFileName := sFileName,
                sExt := sExt );
END_IF
```

要件

開発環境	ターゲットシステム	PLCライブラリ(カテゴリグループ)
TwinCAT v3.1.0	PCまたはCX (x86、x64、ARM)	Tc2_System (システム)

4.1.9 SETBIT32

このファンクションは、渡された32ビット値内のビット番号によって指定されたビットをセットし、その結果の値を返します。

FUNCTION SETBIT32 : DWORD

```
VAR_INPUT
    inVal32 : DWORD;
    bitNo   : SINT;
END_VAR
```

inVal32: 変更される32ビット値

bitNo: セットするビット番号(0～31)。この数は、実行の前に内部的にモジュロ32値に変換されます。

FBDでのファンクションコールの例:

これにより、入力値0のビット31がセットされます。結果は(16進数)値「80000000」です。

要件

開発環境	ターゲットシステム	PLCライブラリ(カテゴリグループ)
TwinCAT v3.1.0	PCまたはCX (x86、x64、ARM)	Tc2_System (システム)

4.1.10 CSETBIT32

このファンクションは、渡された32ビット値のビット番号によって指定されたビットをセット/リセットし、その結果の値を返します。

FUNCTION CSETBIT32 : DWORD

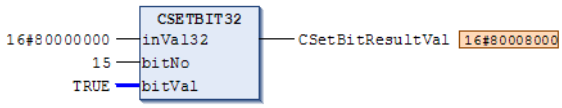
```
VAR_INPUT
    inVal32 : DWORD;
    bitNo   : SINT;
    bitVal  : BOOL;
END_VAR
```

inVal32: 32ビット値

bitNo: セットまたはリセットされるビット数(0～31)。この数は、実行の前に内部的にモジュロ32値に変換されます。

bitVal: ビットがセットまたはリセットされる値です(TRUE = 1、FALSE = 0)。

FBDでのファンクションコールの例:



入力値16#80000000のビット15が1にセットされます。この結果(16#80008000)は、変数CSetBitResultValに割り当てられます。

要件

開発環境	ターゲットシステム	PLCライブラリ(カテゴリグループ)
TwinCAT v3.1.0	PCまたはCX (x86、x64、ARM)	Tc2_System (システム)

4.1.11 GETBIT32

このファンクションは、渡された32ビット値のビット番号によって指定されたビットのステータスをブール値として返します。入力値は変更されません。

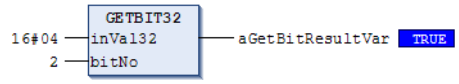
FUNCTION GETBIT32 : BOOL

```
VAR_INPUT
    inVal32 : DWORD;
    bitNo   : SINT;
END_VAR
```

inVal32: 32ビット値

bitNo: 読み込みするビット数(0～31)。この数は、実行の前に内部的にモジュロ32値に変換されます。

FBDでのファンクションコールの例:

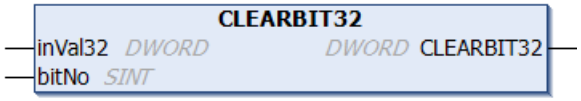


入力値16#04のビット2が読み込みされ、ブール値変数aGetBitResultVarに割り当てられます。この例では、クエリはTRUEを返します。

要件

開発環境	ターゲットシステム	PLCライブラリ(カテゴリグループ)
TwinCAT v3.1.0	PCまたはCX (x86、x64、ARM)	Tc2_System (システム)

4.1.12 CLEARBIT32



このファンクションは、渡された32ビット値のビット番号によって指定されたビットをゼロにリセットし、その結果の値を返します。

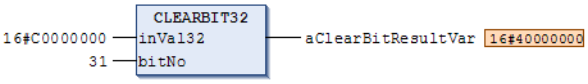
FUNCTION CLEARBIT32 : DWORD

```
VAR_INPUT
    inVal32    : DWORD;
    bitNo      : SINT;
END_VAR
```

inVal32: 変更される32ビット値

bitNo: ゼロにセットするビット番号(0～31)。この数は、実行の前に内部的にモジュロ32値に変換されます。

FBDでのファンクションコールの例:

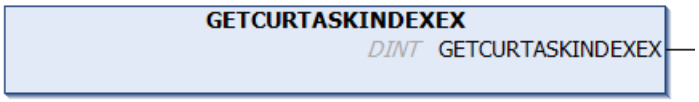


これにより、入力値「C0000000」のビット31がリセットされます。結果は(16進数)値「40000000」となります。

要件

開発環境	ターゲットシステム	PLCライブラリ(カテゴリグループ)
TwinCAT v3.1.0	PCまたはCX (x86、x64、ARM)	Tc2_System (システム)

4.1.13 GETCURTASKINDEXEX



このファンクションは、コール元のタスクのタスクインデックスを判定します。ファンクションブロック GETCURTASKINDEX [▶ 16]とは対照的に、ファンクションが周期的なリアルタイムコンテキストで呼び出されたかどうかを区別できます。

FUNCTION GETCURTASKINDEXEX : DINT

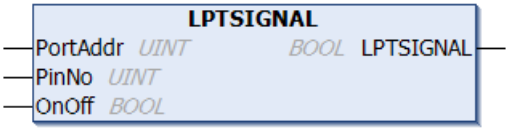
```
VAR_INPUT
    (*keine*)
END_VAR
```

戻りパラメータ	説明
-1	このファンクションは、Windowsコンテキストから呼び出されます。
0	このファンクションは周期的なPLCタスクからではなく、リアルタイムコンテキストからコールします。たとえば、初期化中のFB_initメソッドの自動コールがこれにあたります。
1～n	このファンクションは、周期PLCタスクから呼び出されます。戻り値はタスクインデックスです。

要件

開発環境	ターゲットシステム	PLCライブラリ(カテゴリグループ)
TwinCAT v3.1.0	PCまたはCX (x86、x64、ARM)	Tc2_System (システム)

4.1.14 LPTSIGNAL



このファンクションは、Centronicsインターフェイスの定義済み出力ビットをハイ、またはローレベルにセットします。このファンクションを使用して、オシロスコープでのランタイム測定などが可能です。

FUNCTION LPTSIGNAL: BOOL

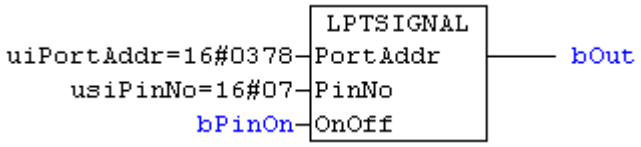
```
VAR_INPUT
    PortAddr : UINT;
    PinNo    : INT;
    OnOff    : BOOL;
END_VAR
```

PortAddr: 目的のLPTインターフェイスで使用可能なポートのアドレスです。

PinNo: PLCが書き込みするピンの番号(ピン0～7)です。

OnOff: そのピンに書き込みされる状態です。

FBDでのファンクションコールの例:



たとえば、ポート378 (16進数)のビット7が1にセットされます。

要件

開発環境	ターゲットシステム	PLCライブラリ(カテゴリグループ)
TwinCAT v3.1.0	PCまたはCX (x86、x64、ARM)	Tc2_System (システム)

4.1.15 TestAndSet

このファンクションを使用して、フラグのチェックおよびセットが可能です。この処理を中断するオプションはありません。これにより、データアクセスを同期できます。排他処理の操作モードには、TestAndSetで切り替えられます。

このファンクションが正常に呼び出されるとTRUEが返され、目的のデータにアクセスできます。このファンクションのコールが失敗するとFALSEが返され、目的のデータにアクセスできません。この場合、代替策を講じる必要があります。

VAR_IN_OUT

```
VAR_IN_OUT
    Flag : BOOL; (* Flag to check if TRUE or FALSE *)
END_VAR
```

Flag: チェックされるブール値フラグです。

- このフラグがFALSEの場合、ファンクションはアクセス可の状態です。アクセス後に(他からのアクセスをブロックするために)フラグをセットして、ファンクションがTRUEを返します。
- このフラグがTRUEの場合、フラグは既に他でアクセス中(ブロック済み)であり、ファンクションがFALSEを返します。

サンプル

```
VAR_GLOBAL
    bGlobalTestFlag : BOOL;
END_VAR
```

```
VAR
    nLocalBlockedCounter : DINT;
END_VAR
```

```
IF TestAndSet(GVL.bGlobalTestFlag) THEN
    (* bGlobalTestFlag was FALSE, nobody was blocking, NOW
    bGlobalTestFlag is set to TRUE and blocking others *)

    (* ... *)

    (* remove blocking by resetting the flag *)
    GVL.bGlobalTestFlag := FALSE;
ELSE
    (* bGlobalTestFlag was TRUE, somebody is blocking *)
    nLocalBlockedCounter := nLocalBlockedCounter + 1;

    (* ... *)
END_IF
```

NEGATIVEサンプル

ファンクションブロックなどでさらにカプセル化する場合、これによって目的の不可分操作が破壊される可能性があるため、注意が必要です。データアクセスのセキュアな同期が行われなくなります。以下は、ファンクションの不正な使用方法を示した例です。この実装で2つのコンテキストが同時にアクセスをリクエストすると、アクセスが許可されたと双方が認識し、データへのセキュアではないアクセスが行われます。

```
FUNCTION_BLOCK FB_MyGlobalLock
VAR_INPUT
    bLock    : BOOL; // set TRUE to lock & set FALSE to unlock
```

```
END_VAR
VAR_OUTPUT
    bLocked : BOOL;
END_VAR

IF bLock THEN
    TestAndSet(bLocked);
ELSE // unlock
    bLocked := FALSE;
END_IF
```

```
IF NOT GVL.fbGlobalLock.bLocked THEN
    GVL.fbGlobalLock(bLock := TRUE);

    (* ... *)

    GVL.fbGlobalLock(bLock := FALSE);
END_IF
```

ファンクションブロックFB_IecCriticalSection [▶ 13]を使用すると、代替Mutexメソッドとしてクリティカルセクションを適用することが可能です。

要件

開発環境	ターゲットシステム	PLCライブラリ(カテゴリグループ)
TwinCAT v3.1.0	PCまたはCX (x86、x64、ARM)	Tc2_System (システム)

4.2 ADSファンクション

4.2.1 ADSLOGDINT

このファンクションは、指定したテキストを画面に表示するメッセージボックスを呼び出して、システムログにエントリを書き込みします。PLCプラグラムは周期的に処理されるため、メッセージボックスなどの項目はエッジ制御下で出力する必要があります。これは、R_TRIGまたはF_TRIGファンクションブロックを連続して配置することで最も簡単に実現できます(以下の例も参照)。

ADSLOGDINTファンクションを使用すると、テキストのユーザが指定した位置に出力するDINT値(4バイト符号付き整数)を挿入できます。これを行うには、格納されているフォーマット文字列の目的の場所に文字「%d」が含まれている必要があります。戻りパラメータには、処理が失敗した場合はファンクションエラーコードが、正常に処理された場合は0が含まれます。

FUNCTION ADSLOGDINT : DINT

```
VAR_INPUT
    msgCtrlMask : DWORD;
    msgFmtStr   : T_MaxString;
    dintArg     : DINT;
END_VAR
```

msgCtrlMask: メッセージ出力のタイプおよび効果を決定する制御マスクです(下表を参照)。

定数	説明
ADSLOG_MSGTYPE_HINT	メッセージタイプは推奨です。
ADSLOG_MSGTYPE_WARN	メッセージタイプは警告です。
ADSLOG_MSGTYPE_ERROR	メッセージタイプはエラーです。
ADSLOG_MSGTYPE_LOG	メッセージはログに書き込みします。
ADSLOG_MSGTYPE_MSGBOX	メッセージはメッセージボックスに出力されます。

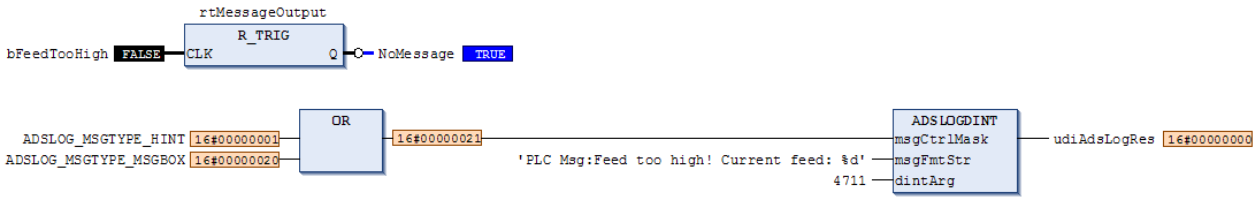
定数	説明
ADSLOG_MSGTYPE_RESOURCE	メッセージはリソースファイルからフェッチされます。(現在未サポート)
ADSLOG_MSGTYPE_STRING	メッセージは直接渡される文字列です(デフォルト)。

この制御マスクは、任意にOR結合できます。

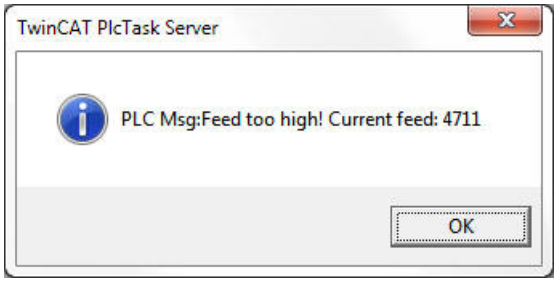
msgFmtStr: 発行されるメッセージを含みます(型: T_MaxString [▶ 102]型)。このパラメータには、DINT値の出力用フォーマットコード%dを任意の位置に含めることができます。

dintArg: メッセージに挿入される数値を含みます。

FBDでのファンクションコールの例:



表示されるメッセージボックス:



ここでは、DINT値4711がメッセージに挿入されます。挿入点は、フォーマット文字列で文字%dによってマークされます。

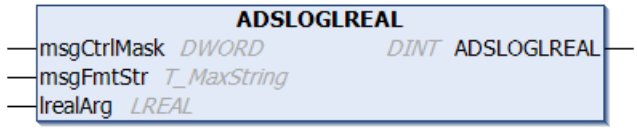
STでのファンクションコールの例:

```
PROGRAM MAIN
VAR
    rtMessageOutput: R_TRIG; (* Declaration *)
    bFeedTooHigh: BOOL;
    udiAdsLogRes: UDINT;
END_VAR

rtMessageOutput(CLK := bFeedTooHigh);
IF rtMessageOutput.Q THEN
    UdiAdsLogRes := ADSLOGDINT( msgCtrlMask := ADSLOG_MSGTYPE_HINT OR ADSLOG_MSGTYPE_MSGBOX,
                    msgFmtStr := 'PLC Msg: Feed too high! Current feed: %d', dintArg:= 4711);
END_IF
```

要件

開発環境	ターゲットシステム	PLCライブラリ(カテゴリグループ)
TwinCAT v3.1.0	PCまたはCX (x86、x64、ARM)	Tc2_System (システム)

4.2.2 ADSLOGLREAL



このファンクションは、指定したテキストを画面に表示するメッセージボックスを呼び出して、システムログにエントリを書き込みします。出力するテキストのユーザが指定した位置に、LREAL値(浮動小数点値)を挿入できます。これを行うには、格納されているフォーマット文字列の目的の場所に文字「%f」が含まれている必要があります。例示するとおり、この場合も必ずエッジ制御を使用してファンクションを呼び出す必要があります(ADSLOGDINTの説明の注意事項も参照)。戻り値には、処理が失敗した場合はファンクションエラーコードが、正常に処理された場合は0が含まれます。

FUNCTION ADSLOGLREAL : DINT

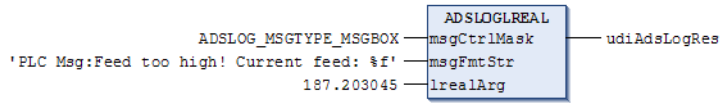
```
VAR_INPUT
    msgCtrlMask    : DWORD;
    msgFmtStr      : T_MaxString;
    lrealArg       : LREAL;
END_VAR
```

msgCtrlMask: メッセージ出力のタイプおよび効果を決定する制御マスクです。メッセージ出力用の制御マスクは、現在ライブラリにグローバル定数としてすべて実装されています(ファンクションADSLOGDINT [▶ 80]の説明を参照)。

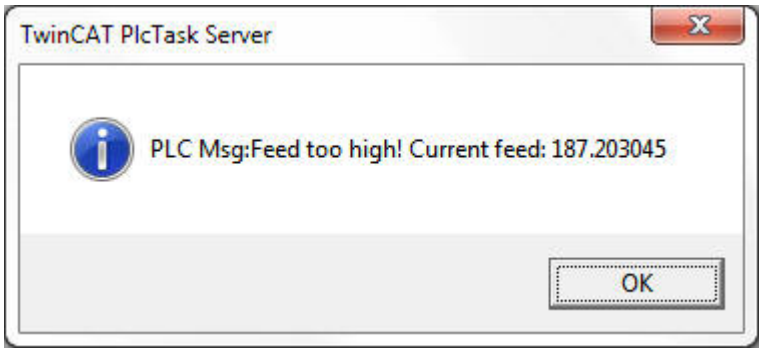
msgFmtStr: 発行されるメッセージを含みます(型: T_MaxString [▶ 102]型)。このパラメータには、DINT値の出力用フォーマットコード%dを任意の位置に含めることができます。

lrealArg: メッセージに挿入される数値を含みます。

FBDでのファンクションコールの例:



表示されるメッセージボックス:



ここでは、LREAL値187.203045がメッセージに挿入されます。挿入点は、フォーマット文字列で文字%fによってマークされます。出力時、小数点以下7桁以降は切り捨てられます。

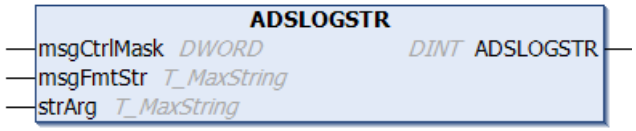
STでのファンクションコールの例:

```
PROGRAM MAIN
VAR
    rtMessageOutput: R_TRIG; (* Declaration *)
    bTemperatureTooHigh: BOOL;
    udiAdsLogRes: UDINT;
END_VAR

rtMessageOutput(CLK := bTemperatureTooHigh);
IF rtMessageOutput.Q THEN
    udiAdsLogRes := ADSLOGLREAL( msgCtrlMask := ADSLOG_MSGTYPE_HINT OR ADSLOG_MSGTYPE_MSGBOX, msgFmt
Str := 'PLC Msg.: Max Temp. reached ! Temperature: %f', lrealArg := 187.203045);
END_IF;
```

要件

開発環境	ターゲットシステム	PLCライブラリ(カテゴリグループ)
TwinCAT v3.1.0	PCまたはCX (x86、x64、ARM)	Tc2_System (システム)

4.2.3 ADSLOGSTR



このファンクションは、指定したテキストを画面に表示するメッセージボックスを呼び出して、システムログにエントリを書き込みします。出力するテキストのユーザが指定した位置に、文字列を挿入できます。これを行うには、格納されているフォーマットの目的の場所に文字「%s」が含まれている必要があります。例示するとおり、この場合も必ずエッジ制御を使用してファンクションを呼び出す必要があります（ ADSLOGDINT [▶ 80]の説明の注意事項も参照）。戻り値には、処理が失敗した場合はファンクションエラーコードが、正常に処理された場合は0が含まれます。

FUNCTION ADSLOGSTR : DINT

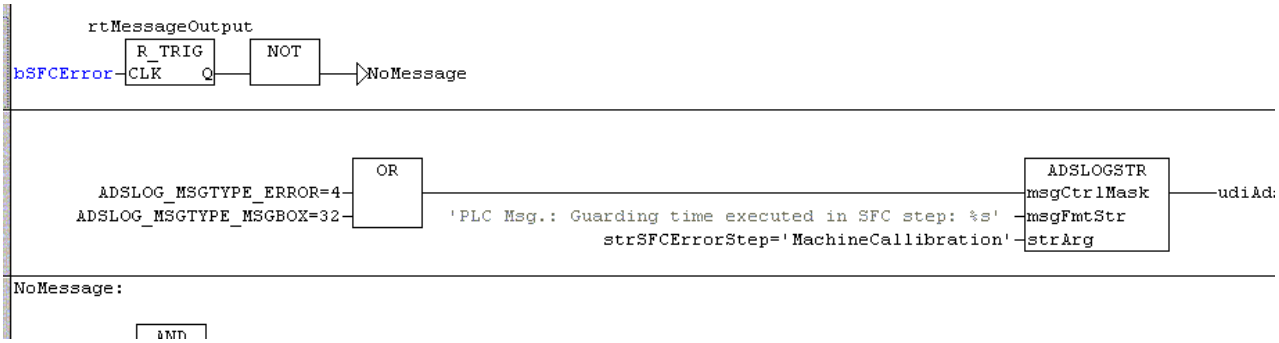
```
VAR_INPUT
    msgCtrlMask   : DWORD;
    msgFmtStr     : T_MaxString;
    strArg        : T_MaxString;
END_VAR
```

msgCtrlMask: メッセージ出力のタイプおよび効果を決定する制御マスクです(ADSLOGDINT [▶ 80]の表を参照)。

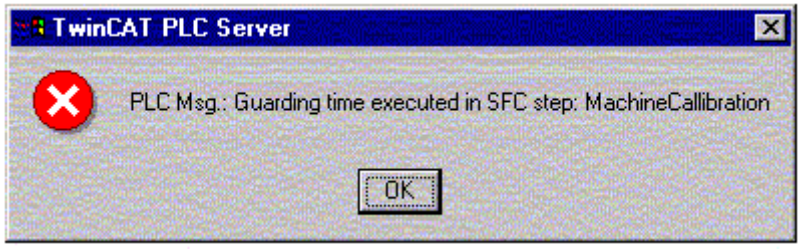
msgFmtStr: 発行されるメッセージを含みます(型: T_MaxString [▶ 102]型)。このパラメータには、テキスト引数の出力用フォーマットコード%sを任意の位置に含めることができます。

strArg: メッセージに挿入される文字列を含みます(型: T_MaxString [▶ 102]型)。

FBDでのファンクションコールの例:



表示されるメッセージボックス:



これにより、PLCプログラマは変数strSFCErrorStepに格納された文字列をメッセージに挿入できます。挿入点は、フォーマット文字列で文字%sによってマークされます。

STでのファンクションコールの例:

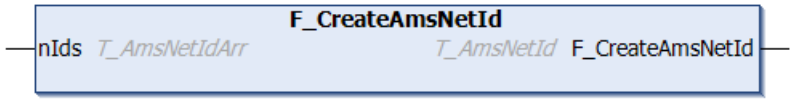
```
PROGRAM MAIN
VAR
    strSFCErrorStep : STRING; (* Declaration*)
    rtMessageOutput: R_TRIG;
    bSFCError: BOOL;
```

```
END_VAR

rtMessageOutput(CLK := bSFCError);
IF rtMessageOutput.Q THEN
    udiAdsLogRes := ADSLOGSTR( msgCtrlMask := ADSLOG_MSGTYPE_ERROR OR ADSLOG_MSGTYPE_MSGBOX, msgFmtS
tr := 'PLC Msg.: Guarding time executed in SFC step: %s', strArg := strSFCErrorStep);
END_IF;
```

要件

開発環境	ターゲットシステム	PLCライブラリ(カテゴリグループ)
TwinCAT v3.1.0	PCまたはCX (x86、x64、ARM)	Tc2_System (システム)

4.2.4 F_CreateAmsNetId



このファンクションは、フォーマットされたNetID文字列を(型: T_AmsNetID [▶ 99]型)を生成し、戻りパラメータとして返します(「127.16.17.3.1.1」など)。

FUNCTION F_CreateAmsNetId : T_AmsNetId

```
VAR_INPUT
    nIds : T_AmsNetIdArr;
END_VAR
```

nIds: バイト列です(型: T_AmsNetIdArr [▶ 100]型)。各バイトは、ネットワークアドレスの番号に対応します。アドレスバイトは、ネットワークバイトオーダで表されます。

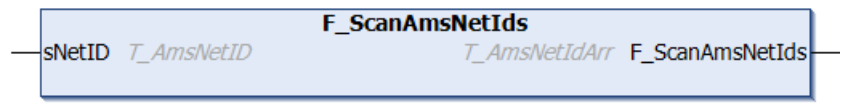
STでのコールの例:

```
PROGRAM MAIN
VAR
    ids    : T_AmsNetIdArr := 127, 16, 17, 3, 1, 1;
    sNetID : T_AmsNetID := '';
END_VAR

sNetID := F_CreateAmsNetId( ids );(* Result: '127.16.17.3.1.1' *)
```

要件

開発環境	ターゲットシステム	PLCライブラリ(カテゴリグループ)
TwinCAT v3.1.0	PCまたはCX (x86、x64、ARM)	Tc2_System (システム)

4.2.5 F_ScanAmsNetIds



ファンクションF_ScanAmsNetIdsを使用して、TwinCATネットワークアドレスを含む文字列を個々のアドレスバイトに変換できます。個々のアドレスバイトは左から右に変換され、バイト配列として返されます(型: T_AmsNetIdArr [▶ 100]型)。アドレスバイトは、ネットワークバイトオーダで表されます。

FUNCTION F_ScanAmsNetIds : T_AmsNetIdArr

```
VAR_INPUT
    sNetID : T_AmsNetID;
END_VAR
```

sNetID: 文字列としてのTwinCATネットワークアドレスです(型: T_AmsNetId [▶ 99]型)。例: 127.16.17.3.1.1

入力パラメータ	戻りパラメータ	説明
sNetID ≠ '' (空文字列) およびsNetID ≠ '0.0.0.0.0.0'	すべてのバイトがnull	変換中にエラーが発生しました。 sNetID文字列のフォーマットをチェックしてください。

STでのコールの例:

以下の例では、ネットワークアドレス文字列「127.16.17.3.1.1」が、アドレスバイト配列に変換されます。

```
PROGRAM MAIN
VAR
    ids    : T_AmsNetIDArr;
    sNetID : T_AmsNetID := '127.16.17.3.1.1';
END_VAR

ids := F_ScanAmsNetIds( sNetID );
(* Result: ids[0]:=127, ids[1]:=16, ids[2]:=17, ids[3]:=3, ids[4]:=1, ids[5]:=1 *)
```

要件

開発環境	ターゲットシステム	PLCライブラリ(カテゴリグループ)
TwinCAT v3.1.0	PCまたはCX (x86、x64、ARM)	Tc2_System (システム)

4.3 文字ファンクション

4.3.1 F_ToCHR

このファンクションは、ASCIIコードを文字列に変換します。

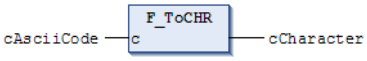
FUNCTION F_ToCHR: STRING

```
VAR_INPUT
    c : BYTE;
END_VAR
```

c: 変換されるASCIIコード

FBDでのコールの例:

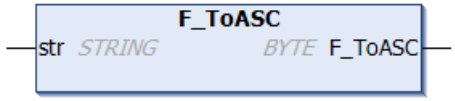
```
PROGRAM P_TEST
VAR
    sCharacter : STRING(1) := '';
    cAsciiCode : BYTE := 16#31;
END_VAR
```



要件

開発環境	ターゲットシステム	PLCライブラリ(カテゴリグループ)
TwinCAT v3.1.0	PCまたはCX (x86、x64、ARM)	Tc2_System (システム)

4.3.2 F_ToASC



このファンクションは、文字列をASCIIコードに変換します。文字列の最初の符号のみ変換されます。空文字列の場合はゼロが返されます。

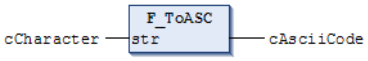
FUNCTION F_ToASC : BYTE

```
VAR_INPUT
    str : STRING;
END_VAR
```

str: 変換される文字列

FBDでのコールの例:

```
PROGRAM P_TEST
VAR
    sCharacter : STRING(1) := '1';
    cAsciiCode : BYTE := 0;
END_VAR
```

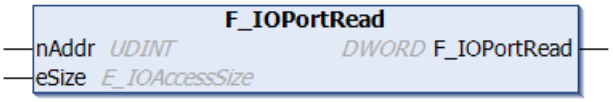


要件

開発環境	ターゲットシステム	PLCライブラリ(カテゴリグループ)
TwinCAT v3.1.0	PCまたはCX (x86、x64、ARM)	Tc2_System (システム)

4.4 I/Oポートアクセス

4.4.1 F_IOPortRead



デジタルI/Oポートは通常1バイト幅のI/O位置であり、メモリにマッピングされ、ポートとしてマッピングされます。この位置に値が書き込みされると、書き込みするビットに応じて出力ピンの電気信号が変更されます。値が入力位置から読み込みされると、この入力ピンの電流論理レベルが個別のビット値として返されます。

ファンクションF_IOPortReadを使用して、eSizeの幅のI/O位置を読み込みできます。このファンクションは、読み込みした値を戻りパラメータとして返します。F_IOPortWrite [▶ 87]ファンクションの説明も参照してください。

FUNCTION F_IOPortRead : DWORD

```
VAR_INPUT
    nAddr : UDINT;
    eSize : E_IOAccessSize;
END_VAR
```

nAddr: I/Oポートアドレスです。

eSize: 読み込みされるデータのバイト数です(型: E_IOAccessSize [▶ 96]型)。

要件

開発環境	ターゲットシステム	PLCライブラリ(カテゴリグループ)
TwinCAT v3.1.0	PCまたはCX (x86、x64、ARM)	Tc2_System (システム)

4.4.2 F_IOPortWrite

ファンクションF_IOPortWriteを使用して、eSizeの幅のI/O位置に書き込みできます。F_IOPortRead [▶ 86] ファンクションの説明も参照してください。

注記

ハードウェアの損傷

データの読み込みのみを行う場合は、ハードウェアへの直接アクセスは問題とはなりません。書き込みアクセスは、ハードウェアや記憶媒体上のデータのクラッシュや破損を招く可能性があります。このファンクションが招く破損により、ハードウェアが起動できなくなる可能性があります。

FUNCTION F_IOPortWrite : BOOL

```
VAR_INPUT
    nAddr  : UDINT;
    eSize  : E_IOAccessSize;
    nValue : DWORD;
END_VAR
```

nAddr: I/Oポートアドレスです。

eSize: 書き込みするデータのバイト数 [▶ 96]です。

nValue: 書き込みする値です。

戻り値	説明
TRUE	エラーなし
FALSE	エラー

STでのサンプルコード:

以下の例では、PCスピーカの直接制御を行うPLCファンクションブロックが、I/Oポートファンクションによって実装されています。

インターフェイス:

```
FUNCTION_BLOCK FB_Speaker
(* Sample code from: "PC INTERN 2.0", ISBN 3-89011-331-1, Data Becker *)
VAR_INPUT
    freq     : DWORD := 10000; (* Frequency [Hz] *)
 tDuration   : TIME := T#1s; (* Tone duration *)
    bExecute : BOOL; (* Rising edge starts function block execution *)
END_VAR
VAR_OUTPUT
    bBusy  : BOOL;
    bError : BOOL;
    nErrID : UDINT;
END_VAR
VAR
    fbTrig : R_TRIG;
    nState : BYTE;
    sts61H : DWORD;
    cnt42H : DWORD;
    cntLo  : DWORD;
```

```
    cntHi  : DWORD;
    timer  : TON;
END_VAR
```

実装:

```
fbTrig( CLK := bExecute );
CASE nState OF
0:
IF fbTrig.Q THEN
    bBusy := TRUE;
    bError := FALSE;
    nErrID := 0;
    timer( IN := FALSE );

    IF F_IOPortWrite( 16#43, NoOfByte_Byte, 182 ) THEN

        cnt42H := 1193180 / freq;
        cntLo := cnt42H AND 16#FF;
        cntHi := SHR( cnt42H, 8 ) AND 16#FF;

        F_IOPortWrite( 16#42, NoOfByte_Byte, cntLo ); (* LoByte *)
        F_IOPortWrite( 16#42, NoOfByte_Byte, cntHi ); (* HiByte *)

        timer( IN := TRUE, PT := tDuration );

        sts61H := F_IOPortRead( 16#61, NoOfByte_Byte );
        sts61H := sts61H OR 2#11;
        F_IOPortWrite( 16#61, NoOfByte_Byte, sts61H ); (* speaker ON *)

        nState := 1;
    ELSE
        nState := 100;
    END_IF
END_IF

1:
timer( );
IF timer.Q THEN

    sts61H := F_IOPortRead( 16#61, NoOfByte_Byte );
    sts61H := sts61H AND 2#11111100;
    F_IOPortWrite( 16#61, NoOfByte_Byte, sts61H ); (* speaker off *)
    bBusy := FALSE;
    nState := 0;
END_IF

100:
bBusy := FALSE;
bError := TRUE;
nErrID := 16#8000;
nState := 0;

END_CASE

Test application:

PROGRAM MAIN
VAR
    fbSpeaker : FB_Speaker;
    bStart : BOOL;
END_VAR


fbSpeaker( freq:= 5000,
```

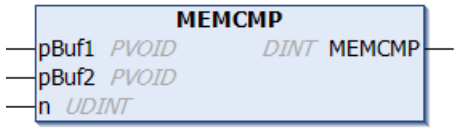
```
tDuration:= t#1s,
bExecute:= bStart );
```

要件

開発環境	ターゲットシステム	PLCライブラリ(カテゴリグループ)
TwinCAT v3.1.0	PCまたはCX (x86、x64、ARM)	Tc2_System (システム)

4.5 メモリファンクション

4.5.1 MEMCMP



ファンクションMEMCMPにより、2つの異なるメモリ領域のPLC変数の値を比較できます。

FUNCTION MEMCMP : DINT

```
VAR_INPUT
    pBuf1 : PVOID;
    pBuf2 : PVOID;
    n     : UDINT;
END_VAR
```

pBuf1: 1番目のメモリ領域(1番目のデータバッファ)の開始アドレスです。

pBuf2: 2番目のメモリ領域(2番目のデータバッファ)の開始アドレスです。

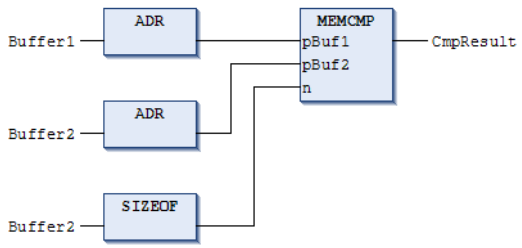
n: 比較されるバイト数です。

このファンクションは、2つのデータバッファの先頭nバイトを比較し、その関係を示す値を返します。

戻りパラメータ	1番目と2番目のデータバッファで異なる先頭バイトの関係
-1	pBuf1がpBuf2よりも小さい
0	pBuf1とpBuf2が等しい
1	pBuf1がpBuf2よりも大きい
0xFF	パラメータ値が間違っています。pBuff1 = 0 または pBuff2 = 0 または n = 0

FBDでのコールの例:

```
PROGRAM MAIN
VAR
    Buffer1   : ARRAY[0..3] OF BYTE;
    Buffer2   : ARRAY[0..3] OF BYTE;
    CmpResult : DINT;
END_VAR
```

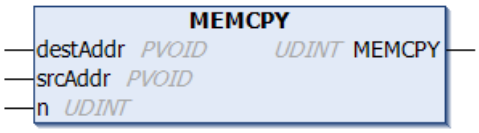


この例では、Buffer2のデータの4バイトがBuffer1のデータの4バイトと比較されます。Buffer1のバイトの方がBuffer2のバイトよりも大きな値となっている箇所が、最初の相違点です。

要件

開発環境	ターゲットシステム	PLCライブラリ(カテゴリグループ)
TwinCAT v3.1.0	PCまたはCX (x86、x64、ARM)	Tc2_System (システム)

4.5.2 MEMCPY



ファンクションMEMCPYを使用して、PLC変数の値を現在のメモリ領域から他のメモリ領域にコピーできます。

FUNCTION MEMCPY : UDINT

```
VAR_INPUT
    destAddr  : PVOID;
    srcAddr   : PVOID;
    n         : UDINT;
END_VAR
```

destAddr: ターゲットメモリ領域の開始アドレスです。

srcAddr: ソースメモリ領域の開始アドレスです。

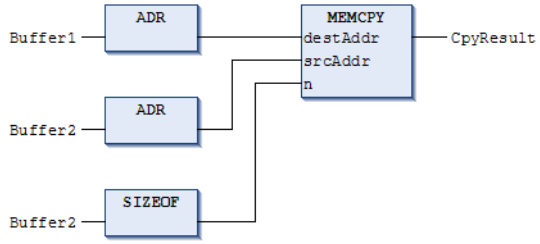
n: コピーされるバイト数です。

このファンクションは、srcAddrで始まるメモリ領域からdestAddrで始まるメモリ領域にnバイトコピーします。

戻りパラメータ	意味
0	パラメータ値が間違っています。destAddr == 0 または srcAddr==0 または n == 0
> 0	正常に実行された場合は、コピーされたバイト数(n)となります。

FBDでのコールの例:

```
PROGRAM MAIN
VAR
    Buffer1   : ARRAY[0..3] OF BYTE;
    Buffer2   : ARRAY[0..3] OF BYTE;
    CpyResult : UDINT;
END_VAR
```

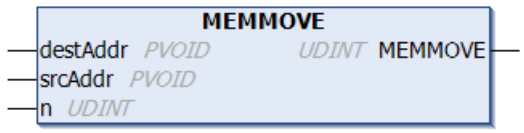


この例では、4バイトがBuffer2からBuffer1にコピーされます。

要件

開発環境	ターゲットシステム	PLCライブラリ(カテゴリグループ)
TwinCAT v3.1.0	PCまたはCX (x86、x64、ARM)	Tc2_System (システム)

4.5.3 MEMMOVE



ファンクションMEMMOVEを使用して、PLC変数の値を現在のメモリ領域から他のメモリ領域に移動できます。

FUNCTION MEMMOVE : UDINT

```
VAR_INPUT
    destAddr : PVOID;
    srcAddr  : PVOID;
    n        : UDINT;
END_VAR
```

destAddr: ターゲットメモリ領域の開始アドレスです。

srcAddr: ソースメモリ領域の開始アドレスです。

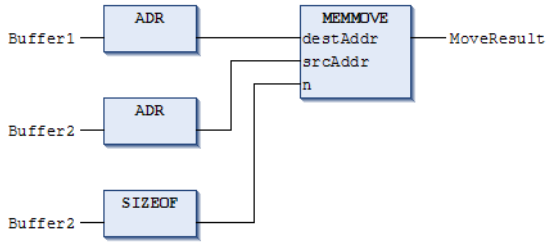
n: コピーされるバイト数です。

このファンクションは、srcAddrで始まるメモリ領域からdestAddrで始まるメモリ領域にnバイトコピーします。

戻りパラメータ	意味
0	パラメータ値が間違っています。destAddr == 0 または srcAddr==0 または n == 0
> 0	正常に実行された場合は、コピーされたバイト数(n)となります。

FBDでのコールの例:

```
PROGRAM MAIN
VAR
    Buffer1    : ARRAY[0..3] OF BYTE;
    Buffer2    : ARRAY[0..3] OF BYTE;
    MoveResult : UDINT;
END_VAR
```

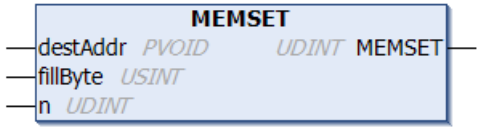


この例では、4バイトがBuffer2からBuffer1に移動されます。

要件

開発環境	ターゲットシステム	PLCライブラリ(カテゴリグループ)
TwinCAT v3.1.0	PCまたはCX (x86、x64、ARM)	Tc2_System (システム)

4.5.4 MEMSET



ファンクションMEMSETによって、特定のメモリ領域のPLC変数を特定の値にセットできます。

FUNCTION MEMSET : UDINT

```
VAR_INPUT
    destAddr : PVOID;
    fillByte : USINT;
    n        : UDINT;
END_VAR
```

destAddr: セットされるメモリ領域の開始アドレスです。

fillByte: 書き込みするバイトの値です。

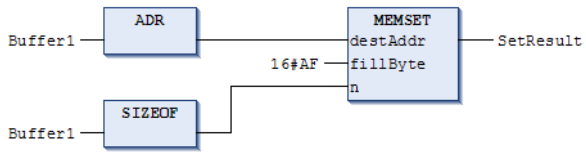
n: セットされるバイト数です。

このファンクションは、アドレスdestAddrから始まるメモリ領域をfillByteによって指定された値でnバイト埋めます。

戻りパラメータ	意味
0	パラメータ値が間違っています。destAddr == 0 または n == 0
> 0	正常に実行された場合は、セットされたバイト数(n)となります。

FBDでのコールの例:

```
PROGRAM MAIN
VAR
    Buffer1 : ARRAY[0..3] OF BYTE;
    SetResult : UDINT;
END_VAR
```



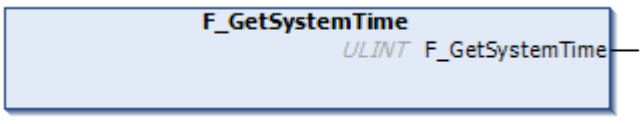
この例では、Buffer1の4バイトが値0xAFにセットされます。

要件

開発環境	ターゲットシステム	PLCライブラリ(カテゴリグループ)
TwinCAT v3.1.0	PCまたはCX (x86、x64、ARM)	Tc2_System (システム)

4.6 タイムファンクション

4.6.1 F_GetSystemTime



このファンクションを使用して、オペレーティングシステムのタイムスタンプを読み込みできます。タイムスタンプは精度100 nsの64ビット整数値で、PLCのコールのたびに更新されます。タイムスタンプはタイミングタスクや時間測定などに利用できます。1単位は100 nsです。時刻は、1601年1月1日からの100 ns間隔の数で表されます。

VAR_OUTPUT

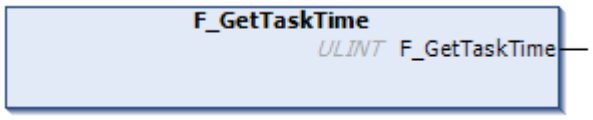
```
VAR_OUTPUT
    F_GetSystemTime    : ULINT;
END_VAR
```

戻り値にはタイムスタンプが含まれます。

要件

開発環境	ターゲットシステム	PLCライブラリ(カテゴリグループ)
TwinCAT v3.1	PCまたはCX (x86、x64、ARM)	Tc2_System (システム) >= 3.4.17.0

4.6.2 F_GetTaskTime



このファンクションを使用して、タスクの開始時刻(タスクを開始する必要がある時刻)を読み込みできますみ取れます。このファンクションは、必ずファンクションが呼び出されたタスクの開始時刻を返します。タイムスタンプは、精度100 nsの64ビット整数値です。タイムスタンプはタイミングタスクや時間測定などに利用できます。1単位は100 nsです。時刻は、1601年1月1日からの100 ns間隔の数で表されます。

VAR_OUTPUT

```
VAR_OUTPUT
    F_GetTaskTime : ULINT;
END_VAR
```

戻り値にはタイムスタンプが含まれます。

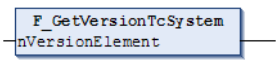
要件

開発環境	ターゲットシステム	PLCライブラリ(カテゴリグループ)
TwinCAT v3.1	PCまたはCX (x86、x64、ARM)	Tc2_System (システム) >= 3.4.17.0

4.7 [廃止]

4.7.1 F_GetVersionTcSystem

このファンクションは廃止されているため、使用しないでください。PLCライブラリからバージョン情報を読み込みするには、グローバル定数stLibVersion_Tc2_System [▶ 107]を使用してください。



このファンクションを使用して、PLCライブラリのバージョン情報を読み込みできます。

FUNCTION F_GetVersionTcSystem : UINT

```
VAR_INPUT
    nVersionElement : INT;
END_VAR
```

nVersionElement: 読み込みするバージョン要素です。使用可能なパラメータ:

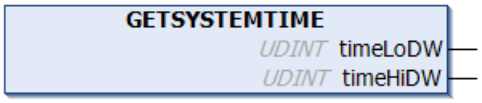
- 1 : メジャー番号
- 2 : マイナー番号
- 3 : リビジョン番号

要件

開発環境	ターゲットシステム	PLCライブラリ(カテゴリグループ)
TwinCAT v3.1.0	PCまたはCX (x86、x64、ARM)	Tc2_System (システム)

4.7.2 GETSYSTEMTIME

このファンクションブロックは、戻り値が2つではなく、1つだけを必要とする新しいファンクションF_GetSystemTime()に変更されます。



このファンクションブロックを使用して、オペレーティングシステムのタイムスタンプを読み込みできます。タイムスタンプは精度100 nsの64ビット整数値で、PLCのコールのたびに更新されます。タイムスタンプはタイミングタスクや時間測定などに利用できます。1単位は100 nsです。時刻は、1601年1月1日からの100 ns間隔の数で表されます。

F_GetSystemTime [▶ 92]を参照してください。

VAR_INPUT

```
VAR_INPUT
(*none*)
END_VAR
```

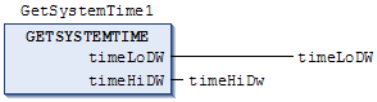
VAR_OUTPUT

```
VAR_OUTPUT
    timeLoDW    : UDINT;
    timeHiDW    : UDINT;
END_VAR
```

timeLoDW: タイムスタンプの下位4バイトが含まれます。

timeHiDW: タイムスタンプの上位4バイトが含まれます。

FBDでのファンクションブロックコールの例:



この例は、インスタンス「GetSystemTime1」によるファンクションブロックのコールを表しており、タイムスタンプとして64ビットの整数値(16進数) 1BCD6EAB05C4E60を出力します。

要件

開発環境	ターゲットシステム	PLCライブラリ(カテゴリグループ)
TwinCAT v3.1.0	PCまたはCX (x86、x64、ARM)	Tc2_System (システム)

4.7.3 GETTASKTIME

このファンクションブロックは、戻り値が2つではなく、1つだけを必要とする新しいファンクションF_GetTaskTime()に変更されます。

このファンクションブロックを使用して、タスクの開始時刻(タスクを開始する必要がある時刻)を読み込みできます。このファンクションブロックは、必ずファンクションブロックインスタンスが呼び出されたタスクの開始時刻を返します。タイムスタンプは、精度100 nsの64ビット整数値です。タイムスタンプはタイミングタスクや時間測定などに利用できます。1単位は100 nsです。時刻は、1601年1月1日からの100 ns間隔の数で表されます。

F_GetTaskTime [▶ 93]を参照してください。

VAR_INPUT

```
VAR_INPUT
(*none*)
END_VAR
```

VAR_OUTPUT

```
VAR_OUTPUT
    timeLoDW : UDINT;
    timeHiDW : UDINT;
END_VAR
```

timeLoDW: タイムスタンプの下4バイトが含まれます。

timeHiDW: タイムスタンプの上4バイトが含まれます。

要件

開発環境	ターゲットシステム	PLCライブラリ(カテゴリグループ)
TwinCAT v3.1.0	PCまたはCX (x86、x64、ARM)	Tc2_System (システム)

5 データ型

5.1 E_IOAccessSize

I/O位置のバイトサイズ(書き込みする、または読み込みするバイト数)です。

```
TYPE E_IOAccessSize :
(
    NoOfByte_Byte  := 1,
    NoOfByte_Word  := 2,
    NoOfByte_DWord := 4
);
END_TYPE
```

要件

開発環境	ターゲットシステム	PLCライブラリ(カテゴリグループ)
TwinCAT v3.1.0	PCまたはCX (x86、x64、ARM)	Tc2_System (システム)

5.2 E_OpenPath

この型の変数は、汎用パス、またはファイルを開く操作を実行するターゲットデバイス上のTwinCATシステムパスのいずれかを選択します。

```
TYPE E_OpenPath :
(
    PATH_GENERIC :=1, (* search/open/create files in selected/generic folder *)
    PATH_BOOTPRJ,     (* search/open/create files in the TwinCAT/
Boot directory (adds the extension .wbp) *)
    PATH_BOOTDATA,    (* reserved for future use*)
    PATH_BOOTPATH,    (* refers to the TwinCAT/Boot directory without adding an extension (.wbp) *)
    PATH_USERPATH1 :=11, (*reserved for future use*)
    PATH_USERPATH2,   (*reserved for future use*)
    PATH_USERPATH3,   (*reserved for future use*)
    PATH_USERPATH4,   (*reserved for future use*)
    PATH_USERPATH5,   (*reserved for future use*)
    PATH_USERPATH6,   (*reserved for future use*)
    PATH_USERPATH7,   (*reserved for future use*)
    PATH_USERPATH8,   (*reserved for future use*)
    PATH_USERPATH9    (*reserved for future use*)
);
END_TYPE
```

要件

開発環境	ターゲットシステム	PLCライブラリ(カテゴリグループ)
TwinCAT v3.1.0	PCまたはCX (x86、x64、ARM)	Tc2_System (システム)

5.3 E_SeekOrigin

この型の変数は、ポインタ移動の原点を示します。

```
TYPE E_SeekOrigin :
(
    SEEK_SET   := 0, (* Seek from beginning of file *)
    SEEK_CUR,        (* Seek from current position of file pointer *)
    SEEK_END         (* Seek from the end of file *)
);
END_TYPE
```

値	説明
SEEK_SET	ファイルの先頭から検索

値	説明
SEEK_CUR	ファイルポインタの現在位置から検索
SEEK_END	ファイルの終端から検索

要件

開発環境	ターゲットシステム	PLCライブラリ(カテゴリグループ)
TwinCAT v3.1.0	PCまたはCX (x86、x64、ARM)	Tc2_System (システム)

5.4 E_TcEventClass

```
TYPE E_TcEventClass :
(
    TCEVENTCLASS_NONE        :=0, (* No class *)
    TCEVENTCLASS_MAINTENANCE :=1, (* Maintenance hint *)
    TCEVENTCLASS_MESSAGE     :=2, (* Message *)
    TCEVENTCLASS_HINT        :=3, (* Hint *)
    TCEVENTCLASS_STATEINFO   :=4, (* State information *)
    TCEVENTCLASS_INSTRUCTION :=5, (* Instruction *)
    TCEVENTCLASS_WARNING     :=6, (* Warning *)
    TCEVENTCLASS_ALARM       :=7, (* Alarm *)
    TCEVENTCLASS_PARAMERROR  :=8 (* Parameter error *)
);
END_TYPE
```

要件

開発環境	ターゲットシステム	PLCライブラリ(カテゴリグループ)
TwinCAT v3.1.0	PCまたはCX (x86、x64、ARM)	Tc2_System (システム)

5.5 E_TcEventClearModes

```
TYPE E_TcEventClearModes :
(
    TCEVENTLOGIOFFS_CLEARACTIVE := 1,
    TCEVENTLOGIOFFS_CLEARLOGGED ,
    TCEVENTLOGIOFFS_CLEARALL
);
END_TYPE
```

要件

開発環境	ターゲットシステム	PLCライブラリ(カテゴリグループ)
TwinCAT v3.1.0	PCまたはCX (x86、x64、ARM)	Tc2_System (システム)

5.6 E_TcEventPriority

```
TYPE E_TcEventPriority :
(
    TCEVENTPRIO_IMPLICIT := 0
);
END_TYPE
```

要件

開発環境	ターゲットシステム	PLCライブラリ(カテゴリグループ)
TwinCAT v3.1.0	PCまたはCX (x86、x64、ARM)	Tc2_System (システム)

5.7 E_TcEventStreamType

```
TYPE E_TcEventStreamType :
(
    TCEVENTSTREAM_INVALID       := 0, (* no source name, no prog id *)
    TCEVENTSTREAM_SIMPLE,             (* no source name, no prog id *)
    TCEVENTSTREAM_NORMAL,             (* source name AND prog id *)
    TCEVENTSTREAM_NOSOURCE,           (* no source name, but prog id *)
    TCEVENTSTREAM_CLASSID,            (* source name AND class id *)
    TCEVENTSTREAM_CLSNOSRC,           (* no source name but class id *)
    TCEVENTSTREAM_READCLASSCOUNT,     (* *)
    TCEVENTSTREAM_MAXTYPE             (* no source name but class id *)
);
END_TYPE
```

要件

開発環境	ターゲットシステム	PLCライブラリ(カテゴリグループ)
TwinCAT v3.1.0	PCまたはCX (x86、x64、ARM)	Tc2_System (システム)

5.8 E_TcMemoryArea

CheckMemoryArea [▶ 68]ファンクションは、指定されたサイズのリクエストされた変数が位置するメモリ領域に関する情報を返します。これには、型がE_TcMemoryAreaの戻り値が使用されます。

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_TcMemoryArea :
(
    Unknown := 0,
    Static  := 1, // static PLC memory
    Dynamic := 2, // dynamic memory
    CNC     :=3
)UDINT;
END_TYPE
```

Unknown: メモリ領域が不明です。たとえば、Windowsコンテキストのメモリである可能性があります。指定されたメモリサイズが2つの異なるメモリ領域にまたがる場合は、このメモリ領域が「不明」として出力されます。さらに、メモリ領域がスタックメモリである場合も「不明」として出力されます。

Static: 静的PLCメモリです。

Dynamic: ランタイム中またはPLCの初期化フェーズ中に動的に割り当てられたメモリです。

CNC: CNCドライバのメモリです。

開発環境	ターゲットシステム	PLCライブラリ(カテゴリグループ)
TwinCAT v3.1.4022	PCまたはCX (x86、x64、ARM)	Tc2_System (システム)

5.9 E_UsrLED_Color

```
TYPE E_UsrLED_Color :
(
    eUsrLED_Off   := 0,
    eUsrLED_Red   := 1,
    eUsrLED_Blue  := 2,
    eUsrLED_Green := 3
);
END_TYPE
```

5.10 EPlcMappingStatus

```
Type EPlcMappingStatus :
(
    MS_Unmapped,
    MS_Mapped,
    MS_Partial
);
End_TYPE
```

このデータ型は、グローバルタイプシステムで定義されます。

開発環境	ターゲットシステム	PLCライブラリ
TwinCAT v3.1.4020	PCまたはCX (x86、x64、ARM)	Tc2_System (システム)

5.11 ST_AmsAddr

この型の変数には、TwinCAT AMSネットワークIDアドレスが含まれます。

```
TYPE ST_AmsAddr :
STRUCT
    netId   : T_AmsNetIdArr;
    port    : T_AmsPort;
END_STRUCT
END_TYPE
```

netId: AMSネットワークIDアドレスです(型: T_AmsNetIdArr [▶ 100]型)。

port: AMSポート番号です(型: T_AmsPort [▶ 100]型)。

要件

開発環境	ターゲットシステム	PLCライブラリ(カテゴリグループ)
TwinCAT v3.1.0	PCまたはCX (x86、x64、ARM)	Tc2_System (システム)

5.12 SYSTEMINFOTYPE

このTwinCAT2データ型は、TwinCAT 3には存在していません。

SystemInfoTypeは、グローバルデータ型であるPlcAppSystemInfoに変更されます。

5.13 SYSTEMTASKINFOTYPE

このTwinCAT2データ型は、TwinCAT 3には存在していません。

SystemTaskInfoTypeは、グローバルデータ型であるPlcTaskSystemInfoに変更されます。

5.14 T_AmsNetId

この型の変数は、ADSコマンドの送信先であるターゲットデバイスのAMSネットワーク識別子の文字列です(型: T_AmsNetId)。文字列はドットで区切られた6の数値フィールドで構成されます。各数値フィールドには、1～254の数値が含まれます。有効なAMSネットワークアドレスは、「1.1.1.2.7.1」や「200.5.7.170.1.7」などです。空文字列が渡されると、ローカルデバイスのAMSネットワーク識別子が自動的に割り当てられます。

```
TYPE T_AmsNetId : STRING(23);
END_TYPE
```

要件

開発環境	ターゲットシステム	PLCライブラリ(カテゴリグループ)
TwinCAT v3.1.0	PCまたはCX (x86、x64、ARM)	Tc2_System (システム)

5.15 T_AmsNetIdArr

```
TYPE T_AmsNetIdArr : ARRAY[0..5] OF BYTE;
END_TYPE
```

この型の変数は、AMSネットワーク識別子を含むバイト配列です。アドレスバイトは、ネットワークバイトオーダで表されます。例: 「127.16.17.3.1.1」の表記

```
byte[0] = 127
byte[1] = 16
byte[2] = 17
byte[3] = 3
byte[4] = 1
byte[5] = 1
```

例: F_ScanAmsNetIds [▶ 84]

要件

開発環境	ターゲットシステム	PLCライブラリ(カテゴリグループ)
TwinCAT v3.1.0	PCまたはCX (x86、x64、ARM)	Tc2_System (システム)

5.16 T_AmsPort

この型の変数には、ADSポート番号が含まれます。TwinCATネットワークのADSデバイスは、AMSネットワークアドレスおよびポート番号によって識別されます。以下のポート番号は、すべてのTwinCATシステムにおいて固定で指定されています。

```
TYPE T_AmsPort : UINT;
END_TYPE
```

指定されたADSポート番号の表:

ADSデバイス	ポート番号
カムコントローラ	900
ランタイムシステム1 ランタイムシステム2 ランタイムシステム3 ランタイムシステム4 ランタイムシステム5 ランタイムシステムn	ランタイムシステム1: 851 (TwinCAT 2では: 801) ランタイムシステム2: 852 (TwinCAT 2では: 811) ランタイムシステム3: 853 (TwinCAT 2では: 821) ランタイムシステム4: 854 (TwinCAT 2では: 831) ランタイムシステム5: 855 ランタイムシステムn: 850 + n、以下同様
NC	500
予約	400
I/O	300
リアルタイムコア	200
イベントシステム(ロガー)	100

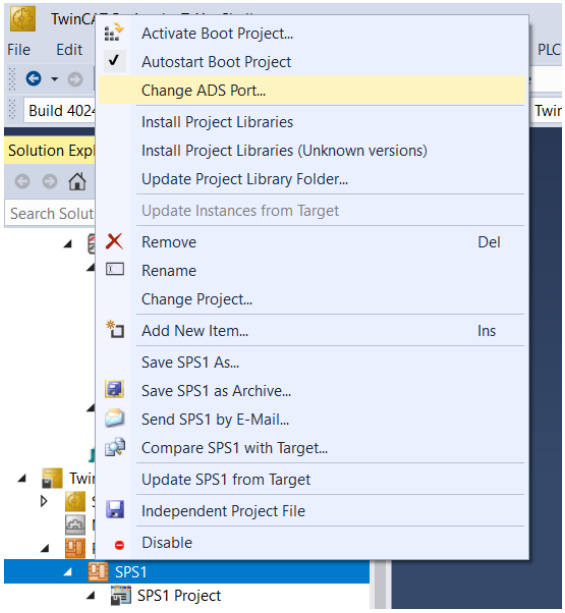
TwinCAT 2システム上では、最大4つの独立したPLCランタイム(PLCプロジェクト)が動作できます。各PLCプロジェクトは、スタンドアロンPLCとみなされます。ADSブロックのコール時には、入力パラメータとしてポート番号とネットアドレスの両方が必要です。

通常、TwinCAT3では800～899のADSポートをPLC用に使用できます。ただし、TwinCAT 2システムとTwinCAT 3システムを分離する場合は、851～899のポートのみを使用することを推奨します。

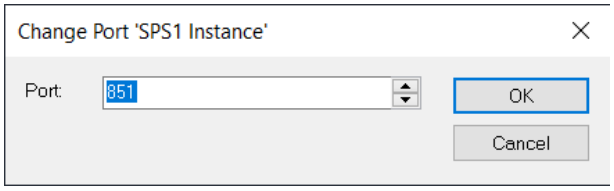
ダイアログを使用してADSポートを設定する場合は、ポート851が設定可能な最小の番号のポートとして表示されます。800～850の範囲を使用するには、ポート番号を入力する必要があります。

以下の手順にしたがって、ダイアログボックスでポート番号を入力します。

1. 目的のPLCプロジェクトを右クリックします。
2. [**Change ADS Port**]をクリックします。



⇨ ダイアログボックスが開きます。



3. 矢印キーを使用するか、ポート番号を入力して、目的のポート番号を選択します。
4. [OK]で入力を確定します。

⇨ これで、ポート番号がシステムに入力されます。

TwinCAT 2システムとTwinCAT 3システムを確実に分離するには、851～899のADSポートのみを使用することを推奨します。
800～899の範囲外のADSポートは入力できません。

要件

開発環境	ターゲットシステム	PLCライブラリ(カテゴリグループ)
TwinCAT v3.1.0	PCまたはCX (x86、x64、ARM)	Tc2_System (システム)

5.17 T_IPv4Addr

この型の変数は、(Ipv4)インターネットプロトコルネットワークアドレスの文字列です。例: 172.16.7.199

```
TYPE T_IPv4Addr : STRING(15);
END_TYPE
```

要件

開発環境	ターゲットシステム	PLCライブラリ(カテゴリグループ)
TwinCAT v3.1.0	PCまたはCX (x86、x64、ARM)	Tc2_System (システム)

5.18 T_IPv4AddrArr

この型の変数は、(IPv4)インターネットプロトコルネットワークアドレスを含むバイト配列です。

```
TYPE T_IPv4AddrArr: ARRAY[0..3] OF BYTE;
END_TYPE
```

アドレスバイトは、ネットワークバイトオーダで表されます。
例: 「172.16.7.199」の表記

```
byte[0] := 172
byte[1] := 16
byte[2] := 7
byte[3] := 199
```

例: F_ScanIPv4AddrIds [▶ 70]

要件

開発環境	ターゲットシステム	PLCライブラリ(カテゴリグループ)
TwinCAT v3.1.0	PCまたはCX (x86、x64、ARM)	Tc2_System (システム)

5.19 T_MaxString

この型の変数は、最大長のPLC文字列です。より長い文字列も使用できますが、この文字列ファンクションでは255文字までに制限されています。

```
TYPE T_MaxString : STRING(MAX_STRING_LENGTH);
END_TYPE
```

```
VAR_GLOBAL CONSTANT
    MAX_STRING_LENGTH : UDINT := 255;
ENd_VAR
```

要件

開発環境	ターゲットシステム	PLCライブラリ(カテゴリグループ)
TwinCAT v3.1.0	PCまたはCX (x86、x64、ARM)	Tc2_System (システム)

5.20 TcEvent

```
TYPE TcEvent
STRUCT
    Class                : UDINT;
    Prio                 : UDINT;
    Id                   : UDINT;
    bQuitRequired        : BOOL;
    DataFormatStrAddress : PVOID;
    UserFlags            : DWORD;
    Flags                : DWORD;
    StreamType           : UDINT;
    SourceString         : STRING[15]; (* TCEVENT_SRCNAMESIZE *)
    SourceId             : UDINT;
    ProgId               : STRING[31]; (* TCEVENT_FMTPRGSIZE *)
END_STRUCT
END_TYPE
```

Class: イベントクラス。Enum型のE_TcEventClass [▶ 97]から値を取得します。

Prio: クラスのイベントの優先度。自由に選択可能なカウントです(1～MaxUDINT)。

Id: イベントのIDです。イベントロガーでの明示的な識別に使用されます。

bQuitRequired: 確認レスポンス必須のオン/オフ切り替えです(TRUE → 確認レスポンス必須)。

DataFormatStrAddress: 文字列のアドレスです。文字列には、フォーマット命令が含まれます(例: %d%f は、IntegerおよびReal (浮動小数点)値をフォーマット)。

UserFlags: 32ビットを自由に使用できます。

Flags: イベント識別用の32ビットです。各ビットの意味は、ライブラリのグローバル変数 [▶ 104]で宣言されています。

StreamType: イベントのタイプです。Enum型のE_TcEventStreamType [▶ 98]から値を取得します。

SourceString: ソース名の文字列です(最大15文字 [▶ 104])。

SourceId: ソースIDです。

ProgId: フォーマッタ名の文字列(Prog-Id)です(最大31文字 [▶ 104])。

要件

開発環境	ターゲットシステム	PLCライブラリ(カテゴリグループ)
TwinCAT v3.1.0	PCまたはCX (x86、x64、ARM)	Tc2_System (システム)

6 グローバル定数

6.1 定数

ポート番号

ポート番号	値	説明
AMSPORT_LOGGER	100	標準ロガーのポート番号です。
AMSPORT_EVENTLOG	110	TwinCATイベントロガーのポート番号です。
AMSPORT_R0_RTIME	200	TwinCATリアルタイムサーバのポート番号です。
AMSPORT_R0_IO	300	TwinCAT I/Oサーバのポート番号です。
AMSPORT_R0_NC	500	TwinCAT NCサーバのポート番号です。
AMSPORT_R0_NCSAF	501	TwinCAT NCサーバ(タスクSAF)のポート番号です。
AMSPORT_R0_NCSVB	511	TwinCAT NCサーバ(タスクSVB)のポート番号です。
AMSPORT_R0_ISG	550	内部的に使用
AMSPORT_R0_CNC	600	TwinCAT NC Iサーバのポート番号です。
AMSPORT_R0_LINE	700	内部的に使用
AMSPORT_R0_PLC	800	TwinCAT PLCサーバのポート番号です(バスコントローラの場合のみ)。
AMSPORT_R0_PLC_RTS1	801	TwinCAT 2.xx PLCサーバランタイム1のポート番号です。
AMSPORT_R0_PLC_RTS2	811	TwinCAT 2.xx PLCサーバランタイム2のポート番号です。
AMSPORT_R0_PLC_RTS3	821	TwinCAT 2.xx PLCサーバランタイム3のポート番号です。
AMSPORT_R0_PLC_RTS4	831	TwinCAT 2.xx PLCサーバランタイム4のポート番号です。
AMSPORT_R0_CAM	900	TwinCAT CAMサーバのポート番号です。
AMSPORT_R0_CAMTOOL	950	TwinCAT CAMTOOLサーバのポート番号です。
AMSPORT_R3_SYSSERV	10000	TwinCATシステムサービスのポート番号です。
AMSPORT_R3_SCOPESERVER	14001	TwinCATスコープサーバのポート番号です。

ADSのステータス

ADSのステータス	値	説明
ADSSTATE_INVALID	0	ADSステータス: 無効
ADSSTATE_IDLE	1	ADSステータス: アイドル
ADSSTATE_RESET	2	ADSステータス: リセット
ADSSTATE_INIT	3	ADSステータス: 初期化
ADSSTATE_START	4	ADSステータス: 開始
ADSSTATE_RUN	5	ADSステータス: 実行
ADSSTATE_STOP	6	ADSステータス: 停止
ADSSTATE_SAVECFG	7	ADSステータス: 設定保存
ADSSTATE_LOADCFG	8	ADSステータス: 設定ロード
ADSSTATE_POWERFAILURE	9	ADSステータス: 電源不良
ADSSTATE_POWERGOOD	10	ADSステータス: 電源良好
ADSSTATE_ERROR	11	ADSステータス: エラー
ADSSTATE_SHUTDOWN	12	ADSステータス: シャットダウン
ADSSTATE_SUSPEND	13	ADSステータス: 待機
ADSSTATE_RESUME	14	ADSステータス: 再開
ADSSTATE_CONFIG	15	ADSステータス: コンフィグレーション(システムがコンフィグレーションモード)
ADSSTATE_RECONFIG	16	ADSステータス: 再コンフィグレーション(システムがコンフィグモードで再起動)
ADSSTATE_STOPPING	17	
ADSSTATE_INCOMPATIBLE	18	
ADSSTATE_EXCEPTION	19	
ADSSTATE_MAXSTATES	20	使用可能なADSステータスの最大数

ADS/システムサービス

予約済みのインデックスグループ	値	説明
ADSIGRP_SYMTAB	16#F000	
ADSIGRP_SYMNAME	16#F001	
ADSIGRP_SYMVAL	16#F002	
ADSIGRP_SYM_HNDBYNAME	16#F003	
ADSIGRP_SYM_VALBYNAME	16#F004	
ADSIGRP_SYM_VALBYHND	16#F005	
ADSIGRP_SYM_RELEASEHND	16#F006	
ADSIGRP_SYM_INFOBYNAME	16#F007	
ADSIGRP_SYM_VERSION	16#F008	
ADSIGRP_SYM_INFOBYNAMEEX	16#F009	
ADSIGRP_SYM_DOWNLOAD	16#F00A	
ADSIGRP_SYM_UPLOAD	16#F00B	
ADSIGRP_SYM_UPLOADINFO	16#F00C	
ADSIGRP_SYMNOTE	16#F010	
ADSIGRP_IOIMAGE_RWIB	16#F020	
ADSIGRP_IOIMAGE_RWIX	16#F021	
ADSIGRP_IOIMAGE_RISIZE	16#F025	
ADSIGRP_IOIMAGE_RWOB	16#F030	
ADSIGRP_IOIMAGE_RWOX	16#F031	
ADSIGRP_IOIMAGE_RWOSIZE	16#F035	
ADSIGRP_IOIMAGE_CLEARI	16#F040	
ADSIGRP_IOIMAGE_CLEARO	16#F050	
ADSIGRP_IOIMAGE_RWIOB	16#F060	
ADSIGRP_DEVICE_DATA	16#F100	
ADSIOFFS_DEVDATA_ADSSTATE	16#0000	
ADSIOFFS_DEVDATA_DEVSTATE	16#0002	

システムサービス/ファイルサービス

システムサービスインデックスグループ	値	説明
SYSTEMSERVICE_OPENCREATE	100	
SYSTEMSERVICE_OPENREAD	101	
SYSTEMSERVICE_OPENWRITE	102	
SYSTEMSERVICE_CREATEFILE	110	
SYSTEMSERVICE_CLOSEHANDLE	111	
SYSTEMSERVICE_FOPEN	120	
SYSTEMSERVICE_FCLOSE	121	
SYSTEMSERVICE_FREAD	122	
SYSTEMSERVICE_FWRITE	123	
SYSTEMSERVICE_FSEEK	124	
SYSTEMSERVICE_FTELL	125	
SYSTEMSERVICE_FGETS	126	
SYSTEMSERVICE_FPUTS	127	
SYSTEMSERVICE_FSCANF	128	
SYSTEMSERVICE_FPRINTF	129	
SYSTEMSERVICE_FEOF	130	
SYSTEMSERVICE_FDELETE	131	
SYSTEMSERVICE_FRENAME	132	
SYSTEMSERVICE_REG_HKEYLOCALMACHINE	200	
SYSTEMSERVICE_SENDEMAIL	300	
SYSTEMSERVICE_TIMESERVICES	400	
SYSTEMSERVICE_STARTPROCESS	500	
SYSTEMSERVICE_CHANGENETID	600	

システムサービス/時刻サービス

システムサービスインデックスオフセット(時刻サービス)	値	説明
TIMESERVICE_DATEANDTIME	1	
TIMESERVICE_SYSTEMTIMES	2	
TIMESERVICE_RTCTIMEDIFF	3	
TIMESERVICE_ADJUSTTIMETORTC	4	

ADSLOGメッセージタイプ

ログ出力のマスク	値	説明
ADSLOG_MSGTYPE_HINT	16#01	
ADSLOG_MSGTYPE_WARN	16#02	
ADSLOG_MSGTYPE_ERROR	16#04	
ADSLOG_MSGTYPE_LOG	16#10	
ADSLOG_MSGTYPE_MSGBOX	16#20	
ADSLOG_MSGTYPE_RESOURCE	16#40	
ADSLOG_MSGTYPE_STRING	16#80	

BOOTDATAフラグ

Bootdataフラグのマスク	値	説明
BOOTDATAFLAGS_RETAIN_LOADED	16#01	
BOOTDATAFLAGS_RETAIN_INVALID	16#02	
BOOTDATAFLAGS_RETAIN_REQUESTED	16#04	
BOOTDATAFLAGS_PERSISTENT_LOADED	16#10	
BOOTDATAFLAGS_PERSISTENT_INVALID	16#20	

BSODフラグのマスク	値	説明
SYSTEMSTATEFLAGS_BSOD	16#01	BSOD: ブルースクリーン
SYSTEMSTATEFLAGS_RTVIOLATION	16#02	リアルタイム違反: レイテンシオーバーラン

ファイル出力モード

ファイル出力のマスク	値	説明
FOPEN_MODEREAD	16#0001	r: ファイルを読み込み専用で開きます。
FOPEN_MODEWRITE	16#0002	w: ファイルを読み込み専用で開きます。ファイルが既に存在する場合は上書きします。
FOPEN_MODEAPPEND	16#0004	a: 読み込み用にファイルを開き、ファイルが既に存在する場合は追記します。ファイルが存在しない場合は作成します。
FOPEN_MODEPLUS	16#0008	+: 読み込み/書き込み用にファイルを開きます。
FOPEN_MODEBINARY	16#0010	b: バイナリでの読み込み/書き込み用にファイルを開きます。
FOPEN_MODETEXT	16#0020	t: テキストでの読み込み/書き込み用にファイルを開きます。

イベントロガー定数

イベントロガーフラグのマスク	値	説明
TCEVENTFLAG_PRIOCLASS	16#0010	クラスおよび優先度はフォーマッタによって定義されます。
TCEVENTFLAG_FMTSELF	16#0020	フォーマット情報はイベントに付随しています。
TCEVENTFLAG_LOG	16#0040	ロギング。
TCEVENTFLAG_MSGBOX	16#0080	メッセージボックスを表示します。
TCEVENTFLAG_SRCID	16#0100	ソース名の代わりにソースIDを使用します。

TwinCATイベントロガーステータスメッセージ	値	説明
TCEVENTSTATE_INVALID	16#0000	無効です。イベントがレポートされなかった場合にも発生します。
TCEVENTSTATE_SIGNALED	16#0001	イベントがレポートされていますが、サインオフまたは確認レスポンスされていません。
TCEVENTSTATE_RESET	16#0002	イベントがサインオフされています(「退出」)。
TCEVENTSTATE_CONFIRMED	16#0010	イベントが確認レスポンスされています。
TCEVENTSTATE_RESETCON	16#0012	イベントがサインオフされ、確認レスポンスされています。

TwinCATイベントロガーステータスメッセージ	値	説明
TCEVENT_SRCNAMESIZE	15	ソース名の最大長。
TCEVENT_FMTPRGSIZE	31	フォーマッタ名の最大長。

その他	値	説明
PI	3.14159265358979323846264338327 95	Pi番号
DEFAULT_ADS_TIMEOUT	T#5s	デフォルトのADSタイムアウト
MAX_STRING_LENGTH	255	T_MaxStringデータ型の最大文字列長

要件

開発環境	ターゲットシステム	PLCライブラリ(カテゴリグループ)
TwinCAT v3.1.0	PCまたはCX (x86、x64、ARM)	Tc2_System (システム)

6.2 ライブラリバージョン

すべてのライブラリに対して、特定のバージョンが付与されています。バージョンは、PLCライブラリリポジトリなどで確認できます。グローバル定数には、ライブラリバージョンに関する情報が含まれています。

Global_Version

```
VAR_GLOBAL CONSTANT
    stLibVersion_Tc2_System : ST_LibVersion;
END_VAR
```

stLibVersion_Tc2_System: Tc2_Systemライブラリのバージョン番号(型: ST_LibVersion型)。

使用しているバージョンが必要なバージョンであるかどうかをチェックするには、ファンクション F_CmpLibVersion [▶ 69]を使用します。

TwinCAT 2で使用されていたライブラリバージョン比較のその他の方法は、すべて使用できなくなっています。

要件

開発環境	ターゲットシステム	PLCライブラリ(カテゴリグループ)
TwinCAT v3.1.0	PCまたはCX (x86、x64、ARM)	Tc2_System (システム)

7 サンプル

7.1 AdsReadInd/AdsReadResファンクションブロックでの例

この例は、単純なADSサーバアプリケーションのPLCへの実装を表しています。このサーバアプリケーションは、ADSクライアントアプリケーションのADSREADリクエストを処理できます。

サンプルアプリケーションでは、ADSREADリクエストを使用してPLCタスクのPLCカウンタ変数をインクリメント/デクリメント、またはリセットしています。処理が成功すると、カウンタ変数の値がADSクライアントアプリケーションに返されます。

ADSサーバアプリケーションの完全なソースは、こちらから解凍できます: https://infosys.beckhoff.com/content/1033/TcPlcLib_Tc2_System/Resources/zip/9007199965315979.zip

このADSサーバに適したADSクライアントアプリケーションは、こちらに記載されています: ADSREADファンクションブロックの例 [▶ 111]

ADSクライアントアプリケーション

PLCタスクで必要なサービスは、インデックスグループパラメータに符号化されます。

IG:0x80000001 → カウンタ変数をインクリメント

IG:0x80000002 → カウンタ変数をデクリメント

IG:0x80000003 → カウンタ変数を0にセット

インデックスオフセットパラメータはゼロです。

リクエストがPLCタスクにルーティングされるように、最高値ビットがインデックスグループパラメータ内にセットされる必要があります。

PLCプログラム

ADSREADリクエストは、ADSREADIND [▶ 28]ファンクションブロックのインスタンスによってPLCタスクの指示として取得されます。その後、インデックスグループとインデックスオフセットパラメータ、および必要なデータ長と妥当性がチェックされます。CASE命令では、PLC変数による任意の操作が実行されます。処理が成功すると、ADSREADRES [▶ 31]ファンクションブロックのインスタンスによって、レスポンスがPLC変数の現在値と共にコール元に返されます。エラーの場合、該当するエラーメッセージが出力されます。次のサイクルでは、他の指示を処理できるようにファンクションブロックのCLEARおよびRESPONDフラグがリセットされます。

宣言部分

```
PROGRAM MAIN
VAR
    fbReadInd    : ADSREADIND;(* Indication function block instance *)
    fbReadRes    : ADSREADRES;(* Response function block instance *)
    sNetId       : T_AmsNetID;
    nPort        : T_AmsPort;
    nInvokeId    : UDINT;
    nIdxGrp      : UDINT;
    nIdxOffs     : UDINT;
    cbLength     : UDINT;(* Requested read data/buffer byte size *)
    cbRead       : UDINT;(* Returned read data/buffer byte size *)
    pRead        : PVOID;(* Pointer to returned read data/buffer *)
    nErrID       : UDINT;(* Read indication result error code *)
    nCounter     : INT;(* Server data *)
END_VAR
```

実装

```
fbReadRes( RESPOND := FALSE );(* Reset response function block *)
 fbReadInd( CLEAR := FALSE );(* Trigger indication function block *)
 IF fbReadInd.VALID THEN(* Check for new indication *)
```

```
    sNetID := fbReadInd.NETID;
    nPort := fbReadInd.PORT;
    nInvokeID := fbReadInd.INVOKEID;
    nIdxGrp := fbReadInd.IDXGRP;
    nIdxOffs := fbReadInd.IDXOFFS;
    cbLength := fbReadInd.LENGTH;

    cbRead := 0;
    pRead := 0;
    nErrID     := DEVICE_SRVNOTSUPP;


CASE nIdxGrp OF
     (*-----------------------------------------------------------------------------------------*)
     16#80000001:
          CASE nIdxOffs OF
               0:(* Increment counter value *)
                    IF cbLength >= SIZEOF(nCounter) THEN
                         nCounter := nCounter + 1;
                         cbRead := SIZEOF(nCounter);
                         pRead := ADR(nCounter);
                         nErrID := NOERR;
                    ELSE (* ADS error (example): Invalid size *)
                         nErrID := DEVICE_INVALIDSIZE;
                    END_IF
          ELSE (* ADS error (example): Invalid index offset *)
               nErrID := DEVICE_INVALIDOFFSET;
          END_CASE
     (*-----------------------------------------------------------------------------------------*)
     16#80000002:
          CASE nIdxOffs OF
               0:(* Decrement counter value *)
                    IF cbLength >= SIZEOF(nCounter) THEN
                         nCounter := nCounter - 1;
                         cbRead := SIZEOF(nCounter);
                         pRead := ADR(nCounter);
                         nErrID := NOERR;
                    ELSE(* ADS error (example): Invalid size *)
                         nErrID := DEVICE_INVALIDSIZE;
                    END_IF
          ELSE (* ADS error (example): Invalid index offset *)
               nErrID := DEVICE_INVALIDOFFSET;
          END_CASE
     (*-----------------------------------------------------------------------------------------*)
     16#80000003:
          CASE nIdxOffs OF
               0:(* Reset counter value *)
                    IF cbLength >= SIZEOF(nCounter) THEN
                         nCounter := 0;
                         cbRead := SIZEOF(nCounter);
                         pRead := ADR(nCounter);
                         nErrID := NOERR;
                    ELSE(* ADS error (example): Invalid size *)
                         nErrID := DEVICE_INVALIDSIZE;
                    END_IF
          ELSE (* ADS error (example): ervice is not supported by server *)
               nErrID := DEVICE_SRVNOTSUPP;
          END_CASE

ELSE (* ADS error (example): Invalid index group *)
     nErrID := DEVICE_INVALIDGRP;
END_CASE

fbReadRes(     NETID := sNetID,
               PORT := nPort,
               INVOKEID := nInvokeID,
```

```
                    LEN := cbRead,
                    DATAADDR := pRead,
                    RESULT := nErrID,
                    RESPOND := TRUE );(* Send read response *)

    fbReadInd( CLEAR := TRUE ); (* Clear indication entry *)
END_IF
```

7.2 AdsWriteInd/AdsWriteResファンクションブロックでの例

この例は、単純なADSサーバアプリケーションのPLCへの実装を表しています。このサーバアプリケーションは、ADSクライアントアプリケーションのADSWRITEリクエストを処理できます。

このサンプルアプリケーションでは、ADSWRITEリクエストを使用して整数値の配列をPLCタスクに送信しています。受信データは、PLCで適切な配列変数にコピーされます。

ADSサーバアプリケーションの完全なソースは、こちらから解凍できます: https://infosys.beckhoff.com/content/1033/TcPlcLib_Tc2_System/Resources/zip/9007199965323915.zip

このADSサーバに適したADSクライアントアプリケーションは、こちらに記載されています: ADSWRITEファンクションブロックの例 [▶ 112]

ADSクライアントアプリケーション

PLCタスクからの目的のサービス/コマンドは、インデックスグループおよびインデックスオフセットパラメータに符号化されます。例:

IG:0x80000005 and IO:0x00000007→ 送信データをPLCの配列にコピー。

リクエストがPLCタスクにルーティングされるように、最高値ビットがインデックスグループパラメータ内にセットされる必要があります。

PLCプログラム

リクエストは、ADSWRITEIND [▶ 29]ファンクションブロックのインスタンスによってPLCタスクの指示として取得されます。その後、インデックスグループ、インデックスオフセット、および送信されたデータ長パラメータの妥当性がチェックされ、PLC変数で目的の操作が実行されます。次のステップで、ADSWRITERES [▶ 32]ファンクションブロックによって、コール元にレスポンス(エラーの場合はエラーコードを含む)が返されます。次のサイクルでは、他の指示を処理できるようにCLEARおよびRESPONDフラグがリセットされます。

ADSWRITEINDファンクションブロックのCLEAR入力の立ち上がりで、最後に送信されたデータへのアドレスポインタが無効(== ゼロ)になります。
このため、送信データはCLEAR入力がTRUEにセットされる前に、PLC変数にコピーされます。

宣言部分

```
PROGRAM MAIN
VAR
   fbWriteInd  : ADSWRITEIND;
   fbWriteRes  : ADSWRITERES;
   sNetId      : T_AmsNetID;
   nPort       : T_AmsPort;
   nInvokeId   : UDINT;
   nIdxGrp     : UDINT;
   nIdxOffs    : UDINT;
   cbWrite     : UDINT;(* Byte size of written data *)
   pWrite      : PVOID;(* Pointer to written data buffer *)
   nResult     : UDINT;(* Write indication result error code *)
   arrInt      : ARRAY[0..9] OF INT;(* Server data *)
END_VAR
```

実装

```
fbWriteRes( RESPOND := FALSE );(* Reset response function block *)
fbWriteInd( CLEAR := FALSE );(* Trigger indication function block *)
IF ( fbWriteInd.VALID ) THEN

   sNetId       := fbWriteInd.NETID;
   nPort        := fbWriteInd.PORT;
   nInvokeId    := fbWriteInd.INVOKEID;
   nIdxGrp      := fbWriteInd.IDXGRP;
   nIdxOffs     := fbWriteInd.IDXOFFS;
   cbWrite      := fbWriteInd.LENGTH;
   pWrite       := fbWriteInd.DATAADDR;
   nResult      := DEVICE_SRVNOTSUPP;

   CASE nIdxGrp OF
      16#80000005:
         CASE nIdxOffs OF
            16#00000007:
               IF cbWrite <= SIZEOF( arrInt ) THEN
                  MEMCPY( ADR( arrInt ), pWrite, MIN( cbWrite, SIZEOF(arrInt) ) );
                  nResult := NOERR;
               ELSE(* ADS error (example): Invalid size *)
                  nResult := DEVICE_INVALIDSIZE;
               END_IF
         ELSE(* ADS error (example): Invalid index offset *)
            nResult := DEVICE_INVALIDOFFSET;
         END_CASE
   ELSE(* ADS error (example): Invalid index group *)
      nResult := DEVICE_INVALIDGRP;
   END_CASE

   fbWriteRes( NETID := sNetId,
            PORT := nPort,
            INVOKEID := nInvokeId,
            RESULT := nResult,
            RESPOND := TRUE ); (* Send write response *)

   fbWriteInd( CLEAR := TRUE ); (* Clear indication entry *)
END_IF
```

7.3 AdsReadファンクションブロックでの例

以下は、ADSクライアントアプリケーションでのADSREADファンクションブロックの使用例です。

ADSクライアントアプリケーションの完全なソースは、こちらから解凍できます: https://infosys.beckhoff.com/content/1033/TcPlcLib_Tc2_System/Resources/zip/9007199965319819.zip

宣言部分

```
PROGRAM MAIN
VAR
    fbReadReq  : ADSREADEX := ( NETID := '', PORT := 851, TMOUT := DEFAULT_ADS_TIMEOUT );
    bIncrement : BOOL;(* Rising edge at this variable starts command execution *)
    bDecrement : BOOL;(* Rising edge at this variable starts command execution *)
    bReset     : BOOL;(* Rising edge at this variable starts command execution *)
    bOther     : BOOL;(* Rising edge at this variable starts command execution *)

    nState     : BYTE;
    bBusy      : BOOL;
    bError     : BOOL;
    nErrID     : UDINT;
    cbRead     : UDINT;

    nCounter   : INT;(* Server data to be read *)
END_VAR
```

実装

```
CASE nState OF
    0:
        IF bIncrement OR bDecrement OR bReset OR bOther THEN
            bBusy := TRUE;
            bError := FALSE;
            nErrID := 0;

            fbReadReq( READ := FALSE );

            IF bIncrement THEN(* Incement counter value *)
                bIncrement := FALSE;
                fbReadReq( IDXGRP := 16#80000001, IDXOFFS := 0, LEN := SIZEOF(nCounter), DESTADDR :=
 ADR(nCounter), READ := TRUE );
            ELSIF bDecrement THEN(* Decrement counter value *)
                bDecrement := FALSE;
                fbReadReq( IDXGRP := 16#80000002, IDXOFFS := 0, LEN := SIZEOF(nCounter), DESTADDR :=
 ADR(nCounter), READ := TRUE );
            ELSIF bReset THEN(* Reset counter value *)
                bReset := FALSE;
                fbReadReq( IDXGRP := 16#80000003, IDXOFFS := 0, LEN := SIZEOF(nCounter), DESTADDR :=
 ADR(nCounter), READ := TRUE );
            ELSIF bOther THEN(* Call unsupported function *)
                bOther := FALSE;
                fbReadReq( IDXGRP := 16#80000004, IDXOFFS := 0, LEN := SIZEOF(nCounter), DESTADDR :=
 ADR(nCounter), READ := TRUE );
            END_IF

            nState := 1;
        END_IF
    1:
        fbReadReq( READ := FALSE, BUSY=>bBusy, ERR=>bError, ERRID=>nErrID, COUNT_R=>cbRead );
        IF NOT bBusy THEN
            IF NOT bError THEN
                nState := 0;(* Success *)
            ELSE
                nState := 100;(* Error *)
            END_IF
        END_IF

    100:(* TODO::Implement error handler *)
        nState := 0;

END_CASE
```

7.4 AdsWriteファンクションブロックでの例

以下は、ADSクライアントアプリケーションでのADSWRITEファンクションブロックの使用例です。

ADSクライアントアプリケーションの完全なソースは、こちらから解凍できます: https://infosys.beckhoff.com/content/1033/TcPlcLib_Tc2_System/Resources/zip/9007199965327755.zip

宣言部分

```
PROGRAM MAIN
VAR
   fbWriteReq   : ADSWRITE := ( NETID := '', PORT := 851, TMOUT := DEFAULT_ADS_TIMEOUT );
   bWrite       : BOOL;(* Rising edge at this variable starts command execution *)
   nState       : BYTE;
   bBusy        : BOOL;
   bError       : BOOL;
   nErrID       : UDINT;
   arrInt       : ARRAY[0..9] OF INT;(* Server data to be written *)
   i            : INT;
END_VAR
```

実装

```
FOR i:=0 TO 9 BY 1 DO (* modify/simulate new data *)
   arrInt[i] := arrInt[i] + 1;
END_FOR


CASE nState OF
   0:
      IF bWrite THEN
         bWrite := FALSE;

         bBusy := TRUE;
         bError := FALSE;
         nErrID := 0;

         fbWriteReq( WRITE := FALSE );
         fbWriteReq( IDXGRP := 16#80000005, IDXOFFS := 7,
                  LEN := SIZEOF( arrInt ), SRCADDR := ADR( arrInt ),
                  WRITE := TRUE );
         nState := 1;
      END_IF

   1:
      fbWriteReq( WRITE := FALSE, BUSY=>bBusy, ERR=>bError, ERRID=>nErrID );
      IF NOT bBusy THEN
         IF NOT bError THEN
            nState := 0;(* Success *)
         ELSE
            nState := 100;(* Error *)
         END_IF
      END_IF

   100:(* TODO::Implement error handler *)
      nState := 0;

END_CASE
```

7.5 PLCからのイベントログ信号の送信/確認レスポンス

以下は、ADSLOGEVENTファンクションブロックの使用例です。

サンプルアプリケーションの完全なソースは、こちらから解凍できます: https://infosys.beckhoff.com/content/1033/TcPlcLib_Tc2_System/Resources/zip/711421451.zip

ステップシーケンス

イベントの設定:

EventConfigData [▶ 102]構造体のパラメータ設定

パラメータの送信:

EventDataAddressで、構造体、配列、またはADR演算子付きの単一の変数のアドレスを生成します。SIZEOF演算子を使用して判定した構造体、配列、または単一の変数の長さをEventDataLength入力に適用します。たとえば、INTおよびLREAL型変数の構造体をイベントで送信する場合、構造体をこれら2つのコンポーネントによって作成し、インスタンス化する必要があります。このインスタンスのアドレスおよび長さを送信する必要があります。

イベントのセット:

Event入力の立ち上がり

イベントのリセット:

Event入力の立ち下がり

イベントの確認レスポンス:

Quit入力の立ち上がり

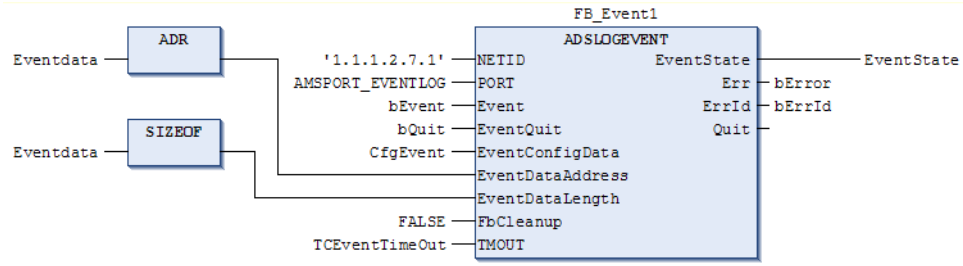
インスタンスの完全削除:

FbCleanup入力の立ち上がりで、インスタンスの内容が完全に削除されます。イベントロガーの既存のイベントは、この方法では直接削除できません。

イベントロガーへのイベント送信後、イベントのステータス [▶ 104]がEventstate出力で明示的に変化します。

ADSLOGEVENTファンクションブロックのコール

```
PROGRAM MAIN
VAR
    FB_Event1    : ADSLOGEVENT;
    CfgEvent     : TcEvent;
    Eventdata    : ParaStruct;
    EventState   : UDINT;
    bEvent       : BOOL;
    bQuit        : BOOL;
END_VAR
VAR CONSTANT
    TCEventDataFormatString : STRING:='%f%d';
    TcEventTimeOut      : TIME:=T#1s;
END_VAR
```



宣言部分

```
PROGRAM MAIN
VAR
    CfgEvent                : TcEvent;
    fbEvent                 : ADSLOGEVENT;
    bSetEvent               : BOOL;(* Rising edge sets event *)

    eventData               : ST_EventData;
    TcEventDataFormatString : STRING    := '%f%d';
END_VAR
```

実装

```
CfgEvent.Class := TCEVENTCLASS_ALARM;
CfgEvent.Prio := 2;
CfgEvent.Id := 1;
CfgEvent.SourceId := 100;
CfgEvent.bQuitRequired := TRUE;
CfgEvent.DataFormatStrAddress := ADR(TcEventDataFormatString);
CfgEvent.Flags := TCEVENTFLAG_LOG OR TCEVENTFLAG_SRCID OR TCEVENTFLAG_AUTOFMTALL;
CfgEvent.StreamType := TCEVENTSTREAM_SIMPLE;
CfgEvent.ProgId :='TcEventFormatter.TcXMLFormatter' ;

eventData.rVal     := 2.65;
eventData.iVal    := 3;

fbEvent( NETID:= '',
        PORT:= 110,
        Event:= bSetEvent,
        EventConfigData:= CfgEvent,
```

```
        EventDataAddress := ADR(eventData) ,
        EventDataLength := SIZEOF(eventData),
        TMOUT:= t#3s);
```

7.6 PLCからのファイルアクセス

この例では、Tc2_systemライブラリのデータアクセス用PLCファンクションブロックが使用されています。新しいファンクションブロックFB_FileCopyが、既存のファンクションブロックを使用して実装されています。FB_FileCopyファンクションブロックを使用すると、バイナルファイルのローカルTwinCATシステムでのコピー、またはローカルとリモートTwinCATシステム間でのコピーが可能です。

サンプルプロジェクトの完全なソースコードは、こちらから解凍できます: https://infosys.beckhoff.com/content/1033/TcPlcLib_Tc2_System/Resources/zip/9007199962636171.zip

FB_FileCopyファンクションブロックを使用してネットワークドライブにアクセスすることはできません。

FB_FileCopyブロックのbExecute入力での立ち上がりで、以下のステップが実行されます。

a) ソースおよびターゲットファイルを開きます。

b) ソースファイルをバッファに読み込みします。

c) バッファから読み込みしたバイトをターゲットファイルに書き込みします。

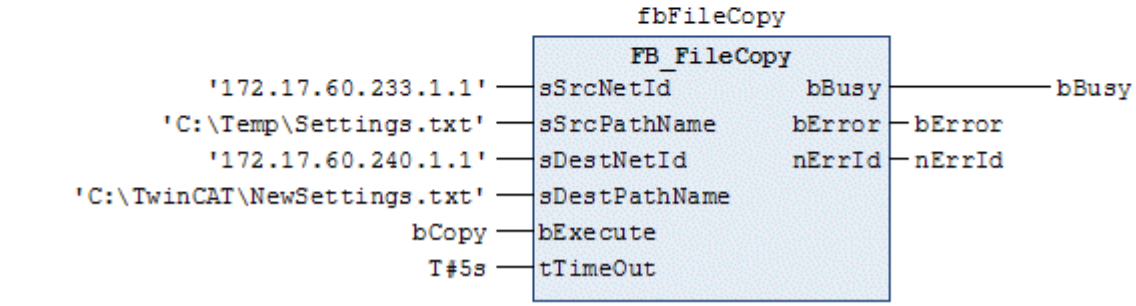
d) ソースファイルの終端に達したかどうかをチェックします。達していなければ、b)およびc)を繰り返します。達していれば、e)に移ります。

e) ソースおよびターゲットファイルを閉じます。

ファイルが1つのセグメントに一度にコピーされます。この例では、バッファサイズが1000バイトに指定されていますが、バッファサイズは変更可能です。

PLCプログラム

```
PROGRAM MAIN
VAR
    fbFileCopy  : FB_FileCopy;
    bCopy       : BOOL;
    bBusy       : BOOL;
    bError      : BOOL;
    nErrId      : UDINT;
END_VAR
```



宣言部分

```
FUNCTION_BLOCK FB_FileCopy
VAR_INPUT
    sSrcNetId           : T_AmsNetId;
    sSrcPathName        : T_MaxString;
    sDestNetId          : T_AmsNetId;
    sDestPathName       : T_MaxString;
```

```
    bExecute            : BOOL;
    tTimeOut            : TIME := DEFAULT_ADS_TIMEOUT;
END_VAR
VAR_OUTPUT
    bBusy               : BOOL;
    bError              : BOOL;
    nErrId              : UDINT;
END_VAR
VAR
    fbFileOpen          : FB_FileOpen;
    fbFileClose         : FB_FileClose;
    fbFileRead          : FB_FileRead;
    fbFileWrite         : FB_FileWrite;
    hSrcFile            : UINT := 0;(* File handle of the source file *)
    hDestFile           : UINT := 0;(* File handle of the destination file *)

    Step                : DWORD;
    RisingEdge          : R_TRIG;
    buffRead            : ARRAY[1..1000] OF BYTE;(* Buffer *)
    cbReadLength        : UDINT := 0;
END_VAR
```

実装

```
RisingEdge(CLK:=bExecute);

CASE Step OF
     0:     (* Idle state *)
          IF RisingEdge.Q THEN
               bBusy := TRUE;
               bError:= FALSE;
               nErrId:=0;
               Step := 1;
               cbReadLength:=0;
               hSrcFile:=0;
               hDestFile:=0;
          END_IF

     1:     (* Open source file *)
          fbFileOpen( bExecute := FALSE );
          fbFileOpen( sNetId := sSrcNetId, sPathName := sSrcPathName,
                         nMode := FOPEN_MODEREAD OR FOPEN_MODEBINARY,
                         ePath := PATH_GENERIC, tTimeout := tTimeOut, bExecute := TRUE );
          Step := Step + 1;
     2:
          fbFileOpen( bExecute := FALSE );
          IF NOT fbFileOpen.bBusy THEN
               IF fbFileOpen.bError THEN
                    nErrId := fbFileOpen.nErrId;
                    bError := TRUE;
                    Step := 50;
               ELSE
                    hSrcFile := fbFileOpen.hFile;
                    Step := Step + 1;
               END_IF
          END_IF

     3:     (* Open destination file *)
          fbFileOpen( bExecute := FALSE );
          fbFileOpen( sNetId := sDestNetId, sPathName := sDestPathName,
                         nMode := FOPEN_MODEWRITE OR FOPEN_MODEBINARY,
                         ePath := PATH_GENERIC, tTimeout := tTimeOut, bExecute := TRUE );
          Step := Step+1;
     4:
          fbFileOpen( bExecute := FALSE );
          IF NOT fbFileOpen.bBusy THEN
               IF fbFileOpen.bError THEN
```

```
                nErrId := fbFileOpen.nErrId;
                bError := TRUE;
                Step := 50;
            ELSE
                hDestFile := fbFileOpen.hFile;
                Step := Step + 1;
            END_IF
        END_IF

5:      (* Read data from source file *)
        cbReadLength := 0;
        fbFileRead( bExecute:= FALSE );
        fbFileRead( sNetId:=sSrcNetId, hFile:=hSrcFile,
                    pReadBuff:= ADR(buffRead), cbReadLen:= SIZEOF(buffRead),
                    bExecute:=TRUE, tTimeout:=tTimeOut );
        Step := Step + 1;
6:
        fbFileRead( bExecute:= FALSE );
        IF NOT fbFileRead.bBusy THEN
            IF fbFileRead.bError THEN
                nErrId := fbFileRead.nErrId;
                bError := TRUE;
                Step := 50;
            ELSE
                cbReadLength := fbFileRead.cbRead;
                Step := Step + 1;
            END_IF
        END_IF

7:      (* Write data to destination file *)
        fbFileWrite( bExecute := FALSE );
        fbFileWrite( sNetId:=sDestNetId, hFile:=hDestFile,
                    pWriteBuff:= ADR(buffRead), cbWriteLen:= cbReadLength,
                    bExecute:=TRUE, tTimeout:=tTimeOut );
        Step := Step + 1;
8:
        fbFileWrite( bExecute := FALSE );
        IF NOT fbFileWrite.bBusy THEN
            IF fbFileWrite.bError THEN
                nErrId := fbFileWrite.nErrId;
                bError := TRUE;
                Step := 50;
            ELSE
                IF fbFileRead.bEOF THEN (* Check if the EOF flag ist set *)
                    Step := 50;     (* Cleanup: close the destination and source files *)
                ELSE
                    Step := 5; (* Repeat reading/writing *)
                END_IF
            END_IF
        END_IF

30:     (* Close the destination file *)
        fbFileClose( bExecute := FALSE );
        fbFileClose( sNetId:=sDestNetId, hFile:=hDestFile, bExecute:=TRUE, tTimeout:=tTimeOut );
        Step := Step + 1;
31:
        fbFileClose( bExecute := FALSE );
        IF NOT fbFileClose.bBusy THEN
            IF fbFileClose.bError THEN
                nErrId := fbFileClose.nErrId;
                bError := TRUE;
            END_IF
            Step := 50;
            hDestFile := 0;
        END_IF
```

```
    40: (* Close source file *)
        fbFileClose( bExecute := FALSE );
        fbFileClose( sNetId:=sSrcNetId, hFile:=hSrcFile, bExecute:=TRUE, tTimeout:=tTimeOut );
        Step := Step + 1;
    41:
        fbFileClose( bExecute := FALSE );
        IF NOT fbFileClose.bBusy THEN
            IF fbFileClose.bError THEN
                nErrId := fbFileClose.nErrId;
                bError := TRUE;
            END_IF
            Step := 50;
            hSrcFile := 0;
        END_IF

    50: (* Error or ready => Cleanup *)
        IF ( hDestFile <> 0 ) THEN
            Step := 30; (* Close the destination file*)
        ELSIF (hSrcFile <> 0 ) THEN
            Step := 40; (* Close the source file *)
        ELSE
            Step := 0;      (* Ready *)
            bBusy := FALSE;
        END_IF

END_CASE
```

8 付録

8.1 ADSリターンコード

エラーコードのグループ分け: 0x000 [▶ 119]...、0x500 [▶ 119]...、0x700 [▶ 120]...、0x1000 [▶ 121]...

グローバルエラーコード

Hex	Dec	HRESULT	名前	説明
0x0	0	0x9811 0000	ERR_NOERROR	エラーはありません。
0x1	1	0x9811 0001	ERR_INTERNAL	内部エラーです。
0x2	2	0x9811 0002	ERR_NORTIME	リアルタイムではありません。
0x3	3	0x9811 0003	ERR_ALLOCLOCKEDMEM	ロック済みの割り当て - メモリエラーです。
0x4	4	0x9811 0004	ERR_INSERTMAILBOX	メールボックスの空き容量なし - 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	0x9811 0005	ERR_WRONGRECEIVEHMSG	間違ったHMSG。
0x6	6	0x9811 0006	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サーバに到達できません。
0x7	7	0x9811 0007	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	0x9811 0008	ERR_UNKNOWNCMDID	不明なコマンドIDです。
0x9	9	0x9811 0009	ERR_BADTASKID	タスクIDが無効です。
0xA	10	0x9811 000A	ERR_NOIO	IOがありません。
0xB	11	0x9811 000B	ERR_UNKNOWNAMSCMD	不明なAMSコマンドです。
0xC	12	0x9811 000C	ERR_WIN32ERROR	Win32エラー。
0xD	13	0x9811 000D	ERR_PORTNOTCONNECTED	ポートが接続されていません。
0xE	14	0x9811 000E	ERR_INVALIDAMSLENGTH	AMS長が無効です。
0xF	15	0x9811 000F	ERR_INVALIDAMSNETID	AMS Net IDが無効です。
0x10	16	0x9811 0010	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	0x9811 0011	ERR_NODEBUGINTAVAILABLE	デバッグが利用できません。
0x12	18	0x9811 0012	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	0x9811 0013	ERR_PORTALREADYCONNECTED	ポートが接続済みです。
0x14	20	0x9811 0014	ERR_AMSSYNC_W32ERROR	AMS Sync Win32エラー。
0x15	21	0x9811 0015	ERR_AMSSYNC_TIMEOUT	AMS Syncタイムアウトです。
0x16	22	0x9811 0016	ERR_AMSSYNC_AMSERROR	AMS同期エラーです。
0x17	23	0x9811 0017	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	0x9811 0018	ERR_INVALIDAMSPORT	AMSポートが無効です。
0x19	25	0x9811 0019	ERR_NOMEMORY	メモリがありません。
0x1A	26	0x9811 001A	ERR_TCPSEND	TCP送信エラーです。
0x1B	27	0x9811 001B	ERR_HOSTUNREACHABLE	ホストに到達できません。
0x1C	28	0x9811 001C	ERR_INVALIDAMSFRAGMENT	AMSフラグメントが無効です。
0x1D	29	0x9811 001D	ERR_TLSSEND	TLS送信エラー - セキュアなADS接続に失敗しました。
0x1E	30	0x9811 001E	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

Hex	Dec	HRESULT	名前	説明
0x500	1280	0x9811 0500	ROUTERERR_NOLOCKEDMEMORY	ロックされているメモリを割り当てられません。
0x501	1281	0x9811 0501	ROUTERERR_RESIZEMEMORY	ルータメモリのサイズを変更できませんでした。
0x502	1282	0x9811 0502	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	0x9811 0503	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	0x9811 0504	ROUTERERR_UNKNOWNPORTTYPE	ポートタイプが不明です。
0x505	1285	0x9811 0505	ROUTERERR_NOTINITIALIZED	ルータが初期化されていません。
0x506	1286	0x9811 0506	ROUTERERR_PORTALREADYINUSE	ポート番号が既に割り当てられています。
0x507	1287	0x9811 0507	ROUTERERR_NOTREGISTERED	ポートが登録されていません。

Hex	Dec	HRESULT	名前	説明
0x508	1288	0x9811 0508	ROUTERERR_NOMOREQUEUES	ポートの最大数に達しています。
0x509	1289	0x9811 0509	ROUTERERR_INVALIDPORT	ポートが無効です。
0x50A	1290	0x9811 050A	ROUTERERR_NOTACTIVATED	ルータがアクティブではありません。
0x50B	1291	0x9811 050B	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	0x9811 050C	ROUTERERR_FRAGMENTTIMEOUT	フラグメントタイムアウトが発生しました。
0x50D	1293	0x9811 050D	ROUTERERR_TOBEREMOVED	ポートが削除されました。

一般的なADSエラーコード

Hex	Dec	HRESULT	名前	説明
0x700	1792	0x9811 0700	ADSERR_DEVICE_ERROR	一般的なデバイスエラーです。
0x701	1793	0x9811 0701	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	0x9811 0702	ADSERR_DEVICE_INVALIDGRP	インデックスグループが無効です。
0x703	1795	0x9811 0703	ADSERR_DEVICE_INVALIDOFFSET	インデックスオフセットが無効です。
0x704	1796	0x9811 0704	ADSERR_DEVICE_INVALIDACCESS	読み込み/書き込みが許可されていません。
0x705	1797	0x9811 0705	ADSERR_DEVICE_INVALIDSIZE	パラメータのサイズが正しくありません。
0x706	1798	0x9811 0706	ADSERR_DEVICE_INVALIDDATA	データ値が無効です。
0x707	1799	0x9811 0707	ADSERR_DEVICE_NOTREADY	デバイスの操作準備が完了していません。
0x708	1800	0x9811 0708	ADSERR_DEVICE_BUSY	デバイスがビジーです。
0x709	1801	0x9811 0709	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ファンクション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	0x9811 070A	ADSERR_DEVICE_NOMEMORY	メモリが不足しています。
0x70B	1803	0x9811 070B	ADSERR_DEVICE_INVALIDPARM	パラメータ値が無効です。
0x70C	1804	0x9811 070C	ADSERR_DEVICE_NOTFOUND	見つかりません(ファイル...)。
0x70D	1805	0x9811 070D	ADSERR_DEVICE_SYNTAX	ファイルまたはコマンドの構文エラーです。
0x70E	1806	0x9811 070E	ADSERR_DEVICE_INCOMPATIBLE	オブジェクトが一致しません。
0x70F	1807	0x9811 070F	ADSERR_DEVICE_EXISTS	オブジェクトは既に存在しています。
0x710	1808	0x9811 0710	ADSERR_DEVICE_SYMBOLNOTFOUND	シンボルが見つかりません。
0x711	1809	0x9811 0711	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	0x9811 0712	ADSERR_DEVICE_INVALIDSTATE	デバイス(サーバ)が無効な状態です。
0x713	1811	0x9811 0713	ADSERR_DEVICE_TRANSMODENOTSUPP	AdsTransModeはサポートされていません。
0x714	1812	0x9811 0714	ADSERR_DEVICE_NOTIFYHNDINVALID	通知ハンドルが無効です。
0x715	1813	0x9811 0715	ADSERR_DEVICE_CLIENTUNKNOWN	通知クライアントが登録されていません。
0x716	1814	0x9811 0716	ADSERR_DEVICE_NOMOREHDLS	使用可能な通知ハンドルがこれ以上ありません。
0x717	1815	0x9811 0717	ADSERR_DEVICE_INVALIDWATCHSIZE	通知サイズが大きすぎます。
0x718	1816	0x9811 0718	ADSERR_DEVICE_NOTINIT	デバイスが初期化されていません。
0x719	1817	0x9811 0719	ADSERR_DEVICE_TIMEOUT	デバイスがタイムアウトしています。
0x71A	1818	0x9811 071A	ADSERR_DEVICE_NOINTERFACE	インターフェイス確認に失敗しました。
0x71B	1819	0x9811 071B	ADSERR_DEVICE_INVALIDINTERFACE	間違ったインターフェイスがリクエストされました。
0x71C	1820	0x9811 071C	ADSERR_DEVICE_INVALIDCLSID	クラスIDが無効です。
0x71D	1821	0x9811 071D	ADSERR_DEVICE_INVALIDOBJID	オブジェクトIDが無効です。
0x71E	1822	0x9811 071E	ADSERR_DEVICE_PENDING	リクエストが保留されています。
0x71F	1823	0x9811 071F	ADSERR_DEVICE_ABORTED	リクエストが中止されています。
0x720	1824	0x9811 0720	ADSERR_DEVICE_WARNING	信号警告です。
0x721	1825	0x9811 0721	ADSERR_DEVICE_INVALIDARRAYIDX	配列インデックスが無効です。
0x722	1826	0x9811 0722	ADSERR_DEVICE_SYMBOLNOTACTIVE	シンボルが有効でありません。
0x723	1827	0x9811 0723	ADSERR_DEVICE_ACCESSDENIED	アクセスが拒否されました。
0x724	1828	0x9811 0724	ADSERR_DEVICE_LICENSENOTFOUND	ライセンスが見つかりません。
0x725	1829	0x9811 0725	ADSERR_DEVICE_LICENSEEXPIRED	ライセンスの期限が切れています。
0x726	1830	0x9811 0726	ADSERR_DEVICE_LICENSEEXCEEDED	ライセンスを超えています。
0x727	1831	0x9811 0727	ADSERR_DEVICE_LICENSEINVALID	ライセンスが無効です。
0x728	1832	0x9811 0728	ADSERR_DEVICE_LICENSESYSTEMID	ラインセンスの問題: システムIDが無効です。
0x729	1833	0x9811 0729	ADSERR_DEVICE_LICENSENOTIMELIMIT	ライセンスの期限がありません。
0x72A	1834	0x9811 072A	ADSERR_DEVICE_LICENSEFUTUREISSUE	ライセンスの問題: まだ時刻が経過していません。
0x72B	1835	0x9811 072B	ADSERR_DEVICE_LICENSETIMETOLONG	ライセンス期間が長すぎます。
0x72C	1836	0x9811 072C	ADSERR_DEVICE_EXCEPTION	システムスタートアップ時の例外です。
0x72D	1837	0x9811 072D	ADSERR_DEVICE_LICENSEDUPLICATED	ライセンスファイルを2回読み込みしました。
0x72E	1838	0x9811 072E	ADSERR_DEVICE_SIGNATUREINVALID	シグネチャが無効です。
0x72F	1839	0x9811 072F	ADSERR_DEVICE_CERTIFICATEINVALID	証明書が無効です。
0x730	1840	0x9811 0730	ADSERR_DEVICE_LICENSEOEMNOTFOUND	OEMからの不明な公開キーです。
0x731	1841	0x9811 0731	ADSERR_DEVICE_LICENSERESTRICTED	このシステムIDに対するライセンスが無効です。

Hex	Dec	HRESULT	名前	説明
0x732	1842	0x9811 0732	ADSERR_DEVICE_LICENSEDEMODENIED	デモライセンスは禁止されています。
0x733	1843	0x9811 0733	ADSERR_DEVICE_INVALIDFNCID	ファンクションIDが無効です。
0x734	1844	0x9811 0734	ADSERR_DEVICE_OUTOFRANGE	有効範囲外です。
0x735	1845	0x9811 0735	ADSERR_DEVICE_INVALIDALIGNMENT	アライメントが無効です。
0x736	1846	0x9811 0736	ADSERR_DEVICE_LICENSEPLATFORM	プラットフォームレベルが無効です。
0x737	1847	0x9811 0737	ADSERR_DEVICE_FORWARD_PL	コンテキスト - パッシブレベルに送信。
0x738	1848	0x9811 0738	ADSERR_DEVICE_FORWARD_DL	コンテキスト - ディスパッチレベルに送信。
0x739	1849	0x9811 0739	ADSERR_DEVICE_FORWARD_RT	コンテキスト - リアルタイムに送信。
0x740	1856	0x9811 0740	ADSERR_CLIENT_ERROR	クライアントエラーです。
0x741	1857	0x9811 0741	ADSERR_CLIENT_INVALIDPARM	サービスに無効なパラメータが含まれています。
0x742	1858	0x9811 0742	ADSERR_CLIENT_LISTEMPTY	ポーリングリストが空です。
0x743	1859	0x9811 0743	ADSERR_CLIENT_VARUSED	var接続が既に使用されています。
0x744	1860	0x9811 0744	ADSERR_CLIENT_DUPLINVOKEID	呼び出されたIDは既に使用されています。
0x745	1861	0x9811 0745	ADSERR_CLIENT_SYNCTIMEOUT	タイムアウトが発生しました - 指定されたADSタイムアウトにリモートターミナルがレスポンスしません。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	0x9811 0746	ADSERR_CLIENT_W32ERROR	Win32補助システムのエラーです。
0x747	1863	0x9811 0747	ADSERR_CLIENT_TIMEOUTINVALID	クライアントタイムアウト値が無効です。
0x748	1864	0x9811 0748	ADSERR_CLIENT_PORTNOTOPEN	ポートが開いていません。
0x749	1865	0x9811 0749	ADSERR_CLIENT_NOAMSADDR	AMSアドレスがありません。
0x750	1872	0x9811 0750	ADSERR_CLIENT_SYNCINTERNAL	Ads syncの内部エラーです。
0x751	1873	0x9811 0751	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	0x9811 0752	ADSERR_CLIENT_REMOVEHASH	テーブルにキーが見つかりません。
0x753	1875	0x9811 0753	ADSERR_CLIENT_NOMORESYM	キャッシュにシンボルがありません。
0x754	1876	0x9811 0754	ADSERR_CLIENT_SYNCRESINVALID	無効なレスポンスを受信しました。
0x755	1877	0x9811 0755	ADSERR_CLIENT_SYNCPORTLOCKED	Syncポートがロックされています。

RTimeエラーコード

Hex	Dec	HRESULT	名前	説明
0x1000	4096	0x9811 1000	RTERR_INTERNAL	リアルタイムシステムの内部エラーです。
0x1001	4097	0x9811 1001	RTERR_BADTIMERPERIODS	タイマ値が無効です。
0x1002	4098	0x9811 1002	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	0x9811 1003	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	0x9811 1004	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	0x9811 1005	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	0x9811 1006	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	0x9811 1007	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	0x9811 100D	RTERR_EXTIRQALREADYDEF	外部同期割り込みが既に適用されています。
0x100E	4110	0x9811 100E	RTERR_EXTIRQNOTDEF	外部同期割り込みは適用されていません。
0x100F	4111	0x9811 100F	RTERR_EXTIRQINSTALLFAILED	外部同期割り込みの適用に失敗しました。
0x1010	4112	0x9811 1010	RTERR_IRQLNOTLESSOREQUAL	間違ったコンテキストでのサービス機能のコール
0x1017	4119	0x9811 1017	RTERR_VMXNOTSUPPORTED	Intel VT-x拡張はサポートされていません。
0x1018	4120	0x9811 1018	RTERR_VMXDISABLED	Intel VT-x拡張はBIOSでは有効でありません。
0x1019	4121	0x9811 1019	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	0x9811 101A	RTERR_VMXENABLEFAILS	Intel VT-xの有効化に失敗しました。

TCP Winsockのエラーコード

Hex	Dec	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レスポンスしなかった、または接続されたホストがレスポンス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明示的に接続を拒否したために、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	WSAEHOSTUNREACH	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			