Anybus® CompactCom™ 40

ソフトウェアデザインガイド

HMSI-216-125 JA 3.4 日本語

Connecting Devices™

必ずお読みください

責任の範囲

本ドキュメントは細心の注意を払って作成されています。 誤字や脱字があればHMS Industrial Networks ABまでご連絡ください。 本ドキュメントに記載されているデータや図表は、何ら拘束力を持ちません。 HMS Industrial Networks ABは、製品開発へ継続的に取り組むという自社ポリシーに基づき、製品に変更を加える権利を留保します。 本ドキュメントの内容は予告なく変更される場合があります。また、本ドキュメントの内容はHMS Industrial Networks ABによる何らかの保証を表明するものではありません。HMS Industrial Networks ABは、本ドキュメント内の誤りについて一切の責任を負いません。

本製品は様々な用途に応用可能です。 本装置の使用者は、必要なあらゆる手段を通じて、本装置の用途が適用される法令、規則、規約、規格の定める性能・安全性に関する要件をすべて満たしていることを検証しなければならないものとします。

HMS Industrial Networks ABは、いかなる場合であっても、本製品のドキュメントに記載されていない機能やタイミング、機能の副作用によって生じた不具合について一切の責任を負いません。 本製品のそのような特徴を直接または間接に使用したことで生じる影響（互換性の問題や安定性の問題など）は、本ドキュメントでは定義されていません。

本ドキュメントの例および図表は、説明のみを目的として使用されています。 本製品の個々の使用においては様々なバリエーションや要件が存在するため、本ドキュメントの例や図表に基づいて本製品を使用したことに関して、HMS Industrial Networks ABは一切の責任を負いません。

知的所有権

本ドキュメントに記載されている製品に組み込まれた技術に関連する知的所有権は、HMS Industrial Networks ABに帰属します。 この知的所有権には、米国およびその他の国における特許および出願中の特許出願が含まれる場合があります。

目次

Page

1 まえがき

1.1 本ドキュメントについて

本ドキュメントは、Anybus CompactComプラットフォームの機能を十分理解していただくためのものです。 Anybus CompactCom 40製品が提供するネットワーク固有の機能についての説明は一切含まれていません。ネットワーク固有の機能に関する情報については、当該のネットワークガイドをご覧ください。

本ドキュメントの読者は、ソフトウェア設計に関する高度な知識、および、産業用ネットワーク通信システムに関する一般的な知識を備えていることが求められます。 追加の情報、文書、サポートなどについては、サポートWebサイトwww.anybus.com/supportを参照ください。

1.2 関連ドキュメント

ドキュメント	作成者	ドキュメント**ID**
Anybus CompactCom M40 Hardware Design Guide	HMS	HMSI-216-126
Anybus CompactCom B40 Design Guide	HMS	HMSI-27-230
Anybus CompactCom Host Application Implementation Guide	HMS	HMSI-27-334
Anybus CompactCom 40 Network Guides（サポートされている各フィールドバスまたはネットワークシステムに関するドキュメント）	HMS	

1.3 ドキュメント更新履歴

バージョン	日付	説明
0.50	2013/07/02	新規作成
0.60	2013/12/20	全般的な変更
1.00	2014/03/28	メジャーアップデート
1.10	2014/05/26	メジャーアップデート
1.20	2014/08/15	メジャーアップデート
1.21	2014/08/26	メジャーアップデート
1.20	2014/11/10	メジャーアップデート
1.31	2015/02/06	マイナーアップデート
2.00	2015/09/24	メジャーアップデート
3.0	2016/08/31	FMからDOXへ変更 メジャーアップデート
3.1	2017/04/03	CC-Link IE Field追加 各種更新と訂正
3.2	2017/06/15	BACnet/IP追加 各種更新と訂正
3.3	2017/07/10	アプリケーションオブジェクト（FFh）にattr #7追加 LEDステータスレジスタに関する情報更新 後方互換性に関するAppendix追加
3.4	2017/11/28	アプリケーションオブジェクト（FFh）にattr #8～#11追加 小規模な変更

1.4 表記と用語

順番通りに実行されなくてはならない指示については、番号の付いたリストが使用されます。

1. まずこれを行います
2. その後これを行います

順番のない指示については、番号付けのないリスト (箇条書き) が使用されます。

- 項目化された情報
- 任意の順序で実行できる指示

アクションと結果が対になる指示については、以下のように表記します。

► このアクションは...

➡ この結果につながります

Bold typefaceはコネクタ、スイッチなどハードウェア上のインタラクティブな部品、またはグラフィックユーザーインターフェース上のメニューやボタンを示します。

等幅フォントはプログラムコードやコンフィグレーションスクリプトなどの
データ入出力表示などに使用されます。

これはこの文書内の相互参照です: 表記と用語, ページ 6

これは、外部リンク(URL)です: www.hms-networks.com

これはインストールおよび/または操作を容易にする可能性のある追加情報です。

機能の低減および/または機器への損傷のリスクを避けるため、またはネットワークのセキュリティのリスクを避けるために、この指示には従わなければなりません。

注意
個人の負傷のリスクを避けるため、この指示には従わなければなりません。

警告
死亡または重篤な障害のリスクを避けるため、この指示には従わなければなりません。

1.5 ドキュメント固有の表記規則

- 「Anybus」または「module」（モジュール）という表現はAnybus CompactComモジュールを表します。
- 「host」（ホスト）または「host application」（ホストアプリケーション）という表現はAnybus機器を制御する機器を表します。
- 16進数はNNNNhまたは0xNNNNの形式で表します。ここで、NNNNは16進の値を表します。
- 特に指定がない限り、Intel形式のバイトオーダーを想定します。
- オブジェクトのインスタンスはインスタンス#0です。
- オブジェクトのアトリビュートは、オブジェクトのインスタンスに格納されています。
- 「Anybus implementation」（Anybusの実装）「Anybus version」（Anybusのバージョン）は、通常、Anybusモジュールにおける実装、すなわち、ネットワークタイプおよびファームウェアリビジョンを表します。
- 任意であることが明示されていない限り、必須であるとみなされるものとします。
- 書き込みの際、「reserved」（予約）と宣言されたフィールドは0に設定されます。
- 読み込みの際、「reserved」（予約）と宣言されたビットフィールドは無視されます。
- 「reserved」（予約）と宣言されているフィールドは、ドキュメントに記述された目的以外で使用しないでください。
- 1バイトは常に8ビットで構成されます。
- 1ワードは常に16ビットで構成されます。

1.6 商標について

- Anybus®はHMS Industrial Networks ABの登録商標です。
- EtherNet/IPはODVA, Inc.の商標です。
- DeviceNetはODVA, Inc.の商標です。
- EtherCAT® Conformance tested EtherCAT®は、ドイツのBeckhoff Automation GmbHよりライセンスを受けた登録商標および特許技術です。

その他の商標は、各所有者に帰属します。

2 Anybus CompactCom 40について

2.1 概要

Anybus CompactCom 40ネットワーク通信モジュールは、要求が厳しい産業用途向けの高性能な通信ソリューションです。 このモジュールは、サーボドライブシステムなどの高性能かつ同期が必要な用途に適しています。 代表的な用途は、PLC、HMI、ロボット、AC/DCドライブ、サーボドライブです。

Anybus CompactComのソフトウェアインターフェースは、ネットワークプロトコルとは独立して設計されているため、ホストアプリケーションは、機能を損なうことなく、同じソフトウェアドライバーを使用してあらゆる主要ネットワークシステムに対応することが可能です。

柔軟性と拡張性を提供するため、ホストアプリケーションとAnybusモジュールとの間はオブジェクト指向のアドレス指定方式が使用されています。 この方式では、メモリマップドデータではなくオブジェクトに対して明示的にリクエストするため、Anybusモジュールがホストアプリケーションから直接情報を取得することが可能になり、非常に高いレベルでホストアプリケーション・Anybusモジュールの統合が可能となります。

> **!** Anybus CompactCom 40シリーズはAnybus CompactCom 30シリーズに対し後方互換性がありますが、40シリーズの方が性能が大幅に高く、30シリーズより機能が豊富になっています。 40シリーズは、アプリケーションコードのマイナーチェンジが必要であるものの、30シリーズ用に開発されたアプリケーションを40シリーズで使用できるという意味において、30シリーズとの後方互換性があります。
>
> 従って、40シリーズの製品をそのまま30シリーズの製品の代わりにすることはできません。

2.2 特長

- トリプルバッファによるプロセスデータ転送をハードウェアでサポート
- ブラックチャネルインターフェースによりセーフティ通信に対し透過的なチャネルを提供
- ホストインターフェースは、ネットワークプロトコルに非依存
- 多言語をサポート
- 高レベルの統合化
- 同期機能をサポート
- 8ビットおよび16ビットのパラレルモード
- SPIモード
- スタンドアロンシフトレジスタモード
- シリアルインターフェースモード（UART）
- オプションでネットワーク固有の高度な機能をサポート

3 ソフトウェア概要

3.1 背景

産業用ネットワークインターフェースの主な機能は、ネットワーク上にある他のデバイスとデータ交換を行うことです。 これは多くの場合においてサイクリックI/Oを交換し、入出力用の2つのメモリバッファを介してデータをホストデバイスに提供することで行われてきました。

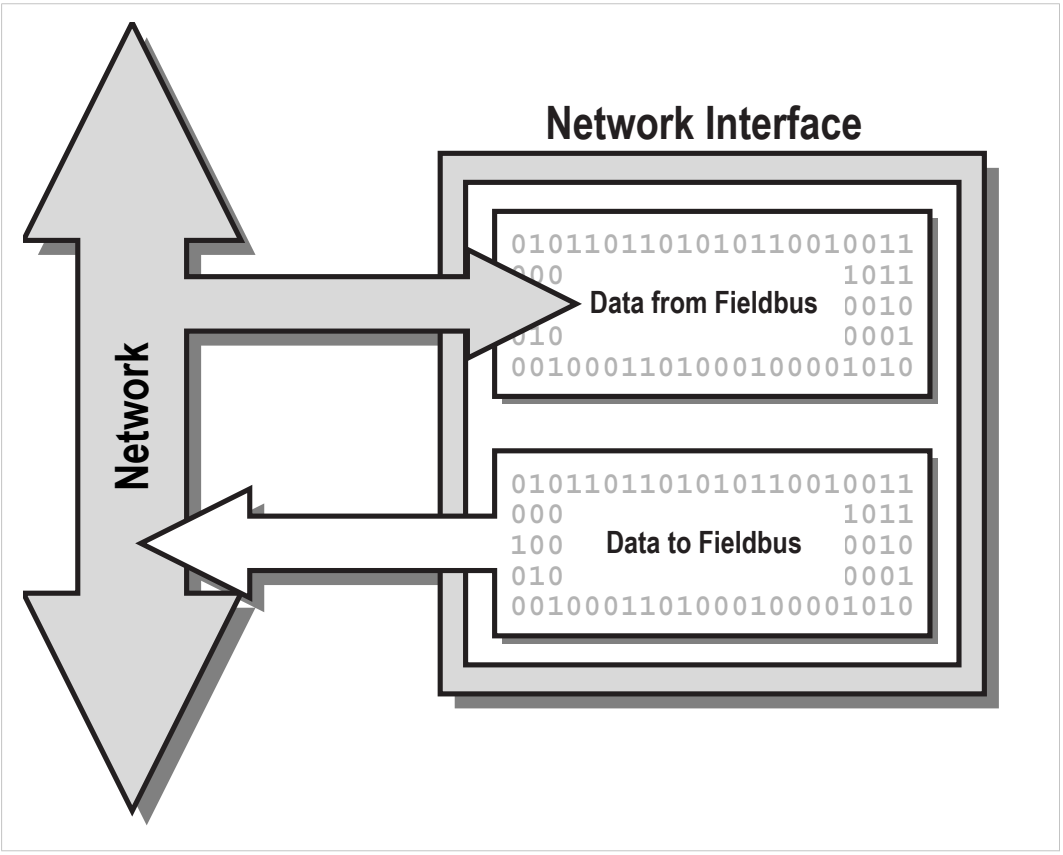

Fig. 1

さらに、ネットワーク機能に対する要求が高まる中でネットワークインターフェースの標準的な役割は、アサイクリックなデータ交換、アラーム処理、診断情報の管理などを含む方向で発展してきました。

通常、これらのネットワーク機能の実装方法は、各ネットワークによって全く異なります。 すなわち、各ネットワーク専用のソフトウェアを実装しない限り、新しい機能をサポートし、その機能を実際に活用することがますます難しくなっていることを意味しています。

Anybus CompactComは、最新のオブジェクト指向技術を採用することによりネットワークごとにソフトウェアドライバーを実装することなしに、ほとんどのネットワークシステムをサポートするとともに、高度なネットワーク機能を活用するための簡単かつ効率的な方法を提供しています。 アサイクリックなデータ交換の要求は統一された方法で変換され、また、専用のオブジェクトにより診断 / アラーム処理を各ネットワークに対応した形で処理する機能を提供します。

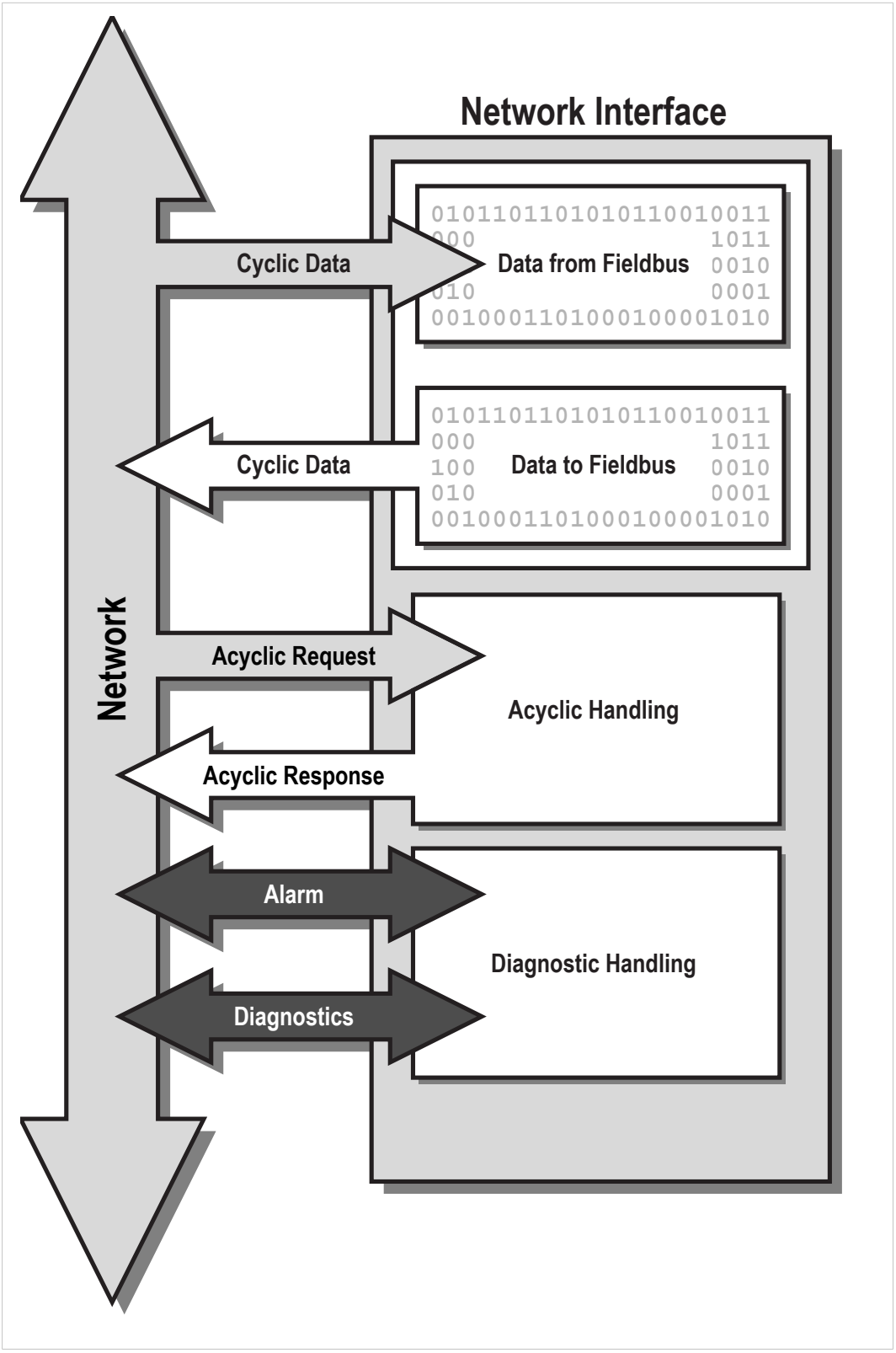
Network Interface
Cyclic Data
Data from Fieldbus
Cyclic Data
Data to Fieldbus
Network
Acyclic Request
Acyclic Handling
Acyclic Response
Alarm
Diagnostic Handling
Diagnostics

Fig. 2

3.2 オブジェクトモデル

3.2.1 基本

このソフトウェアインターフェースは、ホストアプリケーションとAnybusモジュールの両方に対して柔軟かつ論理的なアドレス指定方法を提供できるように、オブジェクト構造化手法で構築されています。最初は、このアプローチに戸惑うかもしれませんが、これは、情報を分類し、アドレス指定する方法の一つにすぎません。

関連する情報とサービスは、"オブジェクト"と呼ばれる実体にグループ化されます。 各オブジェクトは"インスタンス"と呼ばれる下位の実体を持つことができ、インスタンスは、"アトリビュート"と呼ばれるいくつかのフィールドを持つことが可能です。 通常、アトリビュートは、そのオブジェクトに関する情報や設定を表します。 インスタンスは、オブジェクトによっては静的なものまたは実行中に動的に作成されるもののいずれかになります。

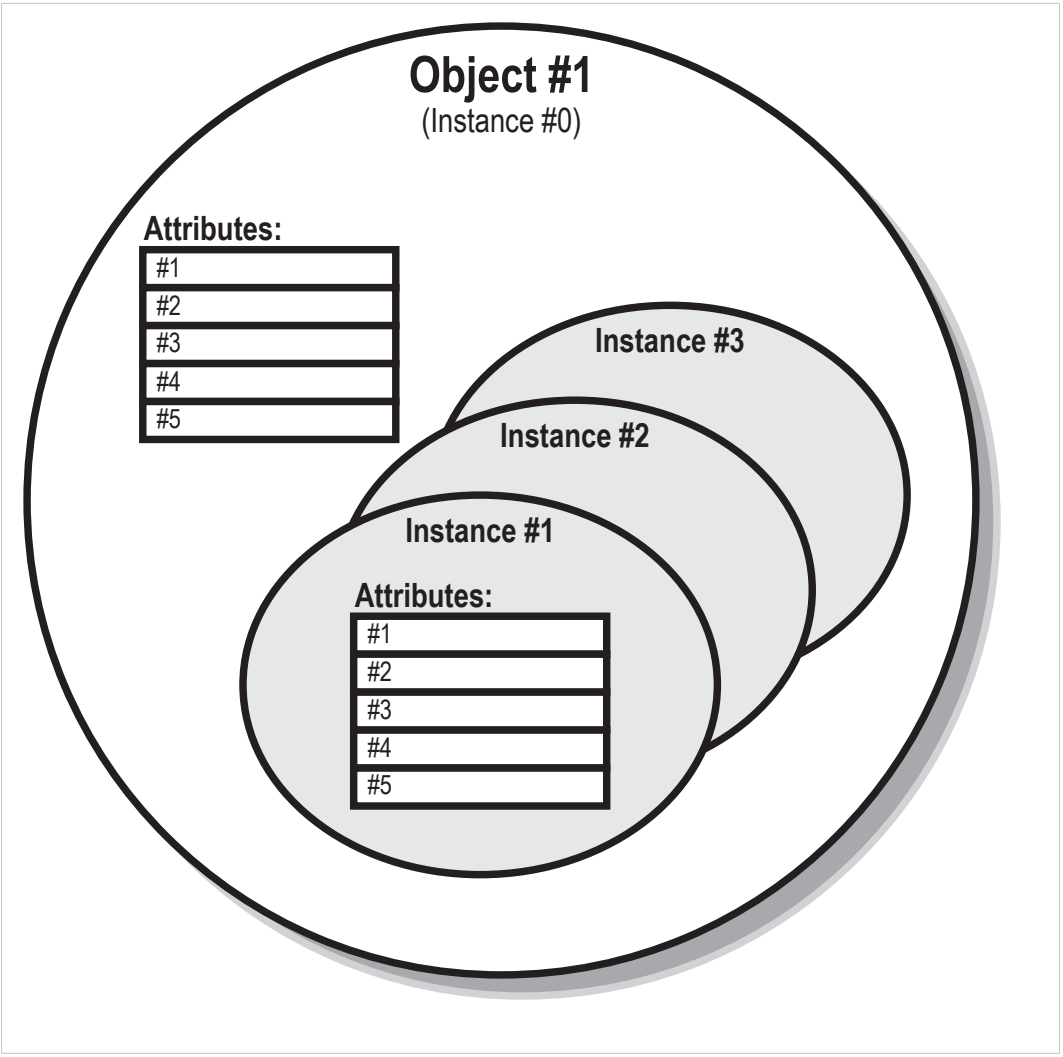

Fig. 3

3.2.2 アドレス指定方式

オブジェクトとその内容には、オブジェクトメッセージングを使用してアクセスします。 各オブジェクトメッセージには、オブジェクト番号、インスタンス、およびアトリビュートが含まれており、これにより、そのメッセージに関連するデータや設定の場所が指定されます。

> **!** このアドレス指定方式は双方向に適用されます。すなわち、Anybusモジュールと同様に、ホストアプリケーションにおいても、送信されてくるオブジェクト要求を解釈して適切なソフトウェア処理に転送する必要があります。

例：

本モジュールには、Anybusモジュール自身に関する共通の設定をグループ化する、「Anybus オブジェクト」と呼ばれるオブジェクトが用意されています。

このオブジェクトには、インスタンス#1 に 「 Firmware version」 （ アトリビュート#2) というアトリビュートが含まれています。 ホストは、オブジェクト#1 (Anybus オブジェクト) のインスタンス#1、アトリビュート#2 (ファームウェアバージョン) に対して**Get Attribute**要求を発行するだけで、本モジュールのファームウェアバージョンを取得できます。

3.2.3 オブジェクトのカテゴリ

オブジェクトは、その物理的な場所に基づいて、大きく2つのカテゴリにグループ化されます。

Anybus モジュールオブジェクト	このオブジェクトはAnybusファームウェアの一部で、通常、モジュールの動作やネットワークにおける作用を制御します。
ホストアプリケーションオブジェクト	このオブジェクトはホストアプリケーションのファームウェア内に置かれ、Anybusモジュールからアクセスされます。 そのため、ホストアプリケーションには、送信されてくるオブジェクト要求に対する適切な処理を実装する必要があります。

3.2.4 標準オブジェクトの実装

あらゆる主要ネットワークシステムの要求に対応できるように、標準オブジェクトが実装されています。大抵の場合、これらのオブジェクトをサポートするだけで、ネットワークの種類に関係なく十分な機能が得られます。

オプションとして、ネットワーク特有の高度な機能を使用できるように、ネットワーク特有のオブジェクトもサポート可能です。 これらのオブジェクトについては、各ネットワークガイドで別途説明しています。

Anybus モジュールオブジェクト

Anybus CompactCom 40の標準のファームウェアでは、以下のオブジェクトが実装されています。

- Anybusオブジェクト
- 診断オブジェクト
- ネットワークオブジェクト
- ネットワーコンフィグレーションオブジェクト
- Anybusファイル システムインターフェース・オブジェクト
- ファンクショナルセーフティモジュール・オブジェクト
- ネットワーク固有オブジェクト

これらのオブジェクトに関してどの程度の実装を行う必要があるかは、ホストアプリケーションの要件により異なります。

下記も参照してください。

Anybus モジュールオブジェクト, ページ 61

ホストアプリケーションオブジェクト

ホストアプリケーションでは以下のオブジェクトの実装が行えます。

- アプリケーションデータオブジェクト（必須）
- アプリケーションオブジェクト（必須）
- SYNCオブジェクト（任意）
- モジュラーデバイスオブジェクト（任意）
- アセンブリマッピングオブジェクト（任意）
- アプリケーション・ファイル システムインターフェース・オブジェクト（任意）
- Energyコントロールオブジェクト（任意）
- ファンクショナルセーフティオブジェクト（任意）
- ネットワーク固有オブジェクト（任意）

アプリケーションデータオブジェクトとアプリケーションオブジェクトは必ず実装する必要があります。ただし、実際にどのオブジェクトを実装すべきかは、ホストアプリケーションの要件に大きく依存します。

下記も参照してください。

ホストアプリケーションオブジェクト, ページ 100

3.3 ネットワークのデータ交換

ネットワーク上で交換されるデータは、アプリケーションデータオブジェクトにグループ化されます。このオブジェクトは、ホストアプリケーションのファームウェアに実装する必要があります。このオブジェクト内のインスタンス（以下、アプリケーションデータインスタンスを「ADI」と表記）は、ネットワークデータのブロックを表します。

通常、ADIはネットワークからアクセスされるアサイクリックパラメータに関連付けられています。 例えば、DeviceNetおよびEtherNet/IPでは、ベンダー固有の専用CIPオブジェクトとしてADIが表されます。一方、PROFIBUSでは、ADIはアサイクリックなDP-V1リード/ライトサービスによりアクセスされます。CANopenオブジェクトディクショナリをベースとするEtherCATおよびその他のプロトコルでは、ADIはオブジェクトディクショナリで定義されたPDOにマップされます。

ADIは、ホストアプリケーションにより、または（可能であれば）ネットワークから、プロセスデータとしてもマッピング可能です。 プロセスデータはAnybus CompactComホストプロトコルの専用データチャネルを介して交換され、通常、高速サイクリックネットワークI/Oと関連付けられます。プロセスデータの厳密な表現は、ネットワークによって異なります。例えばPROFIBUSでは、プロセスデータはIOデータに関連付けられます。

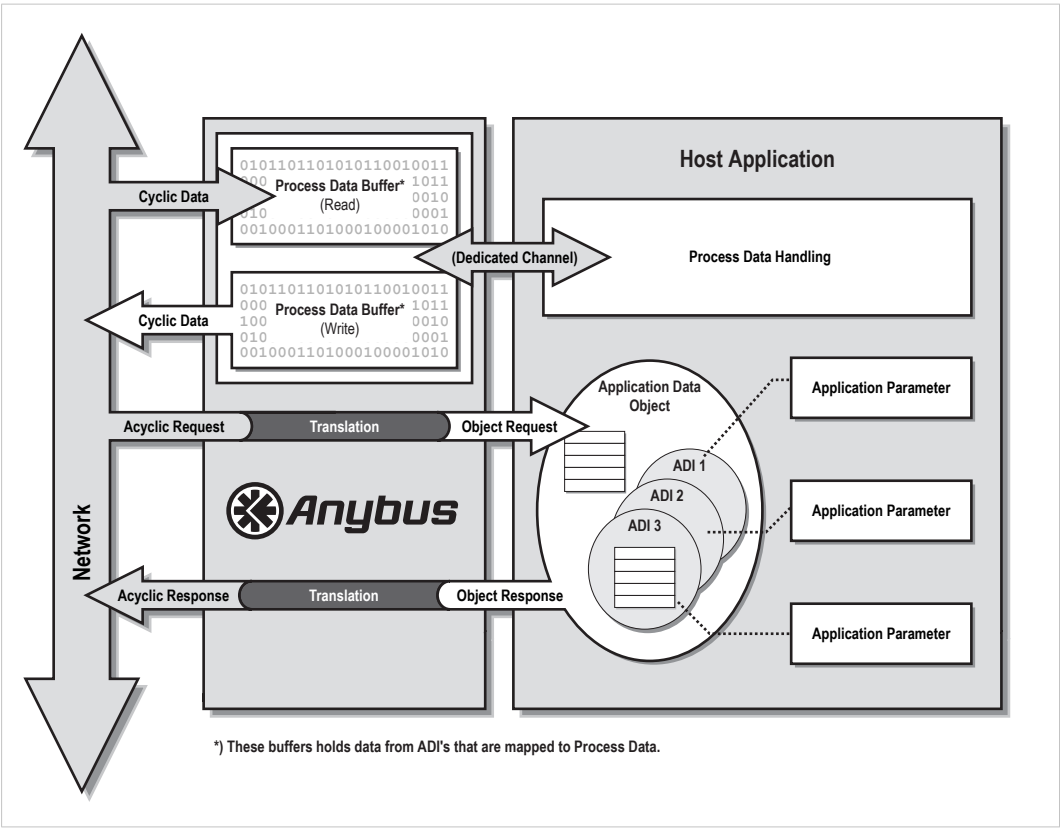

Fig. 4

各ADIには、名前、データ型、値の範囲、デフォルト値を設定できます。これらはすべて、ネットワークからアクセスできます（対象のネットワークによりサポートされている場合）。 これにより、上位のネットワークデバイス（ネットワークマスター、コンフィグレーションツールなど）は実際の名前やタイプ（利用可能な場合）を使用してアサイクリックなパラメータを認識できるため、ネットワークコンフィグレーションプロセスが簡素化されます。

ネットワークシステムによっては、サイクリックとアサイクリックアクセスのどちらでも同じパラメータにアクセスできます。 このようにAnybus CompactCom 40の場合、明示的なオブジェクト要求とプロセスデータの両方で同時にADIにアクセスできることを意味します。 Anybusモジュールは、いかなる方法によってもこのデータを同期させようとはしません。必要に応じてホストアプリケーションにてその仕組みを実装する必要があります。

Anybusインターフェースは、リトルエンディアンのメモリアドレッシング方式を使用します。 そのため、バイトデータは最下位バイト（LSB）から最上位バイト（MSB）に向かって配置されます。 ただし、

Anybusは、ネットワークにおける実際の表現方式に従ってADIの値を透過的に処理します（この表現方式は、初期化時にアプリケーションに通知されます）。 必要な場合には、アプリケーションのドライバーにてバイトの並びを入れ替えます。 これは、以下の理由によります。

- メモリ容量の制限および起動時間の要求が存在するため、AnybusではADIのすべてのデータ型に関する情報を持つことができない。
- それに代えてパラメータのリード/ライト要求の前に毎回データ型を読み込むのでは時間がかかりすぎる。

下記も参照してください。

Anybusステートマシン, ページ 43

ネットワーク オブジェクト (03h), ページ 73

ファンクショナルセーフティオブジェクト (E8h), ページ 103

3.4 診断

Anybus CompactCom 40では、ホストに関する診断を行うための専用のオブジェクトが用意されています。 ホストアプリケーションが診断イベントを通知するには、このオブジェクトに対しインスタンスの生成を行います。 その診断イベントが解決した場合には、ホストアプリケーションからそのインスタンスの削除を行います。

各イベントには、イベントの種類を表すイベントコードや、イベントの重大度を表す重大度コードが設定されます。 この情報が実際にどのように表されるかは、ネットワークにより大きく異なります。

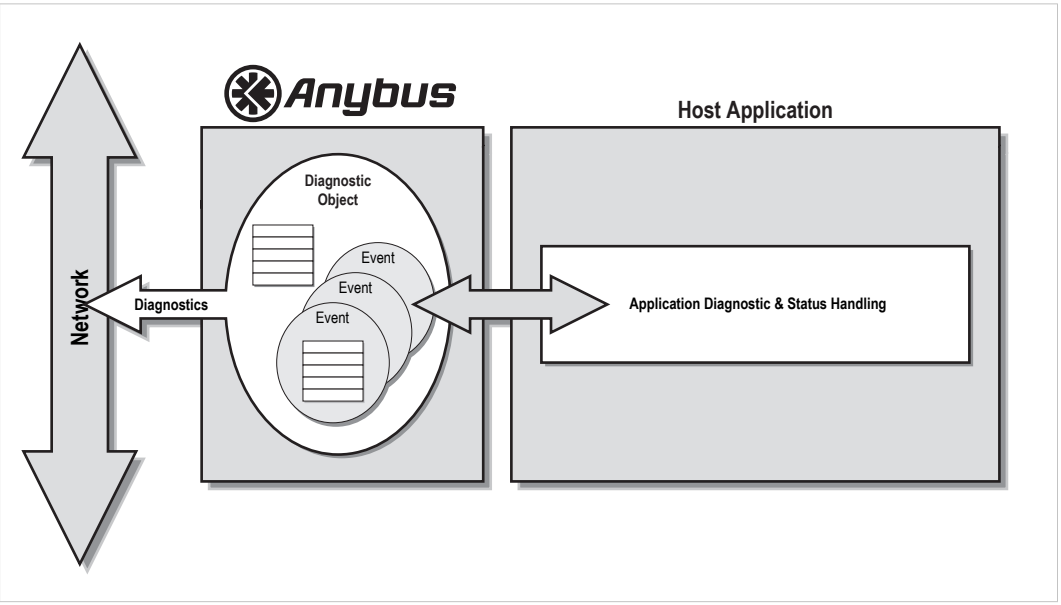

Fig. 5

下記も参照してください。

診断オブジェクト (02h), ページ 68

3.5 ファイルシステム

Anybus CompactCom 40シリーズのモジュールには、ファイルシステムが組み込まれています。

FTPをサポートしていないモジュールでは、Anybusファイルシステムインターフェース・オブジェクト（0Ah）を使用してファームウェアファイルをファームウェアディレクトリに保存できます。 これらのモジュールでは、これ以外の方法/目的でファイルシステムにアクセスすること、またはファイルシステムを使用することはできません。

FTPをサポートしているモジュールでは、アプリケーションおよびネットワークから組み込みファイルシステムにアクセスできます。 以下の3つのディレクトリがあらかじめ定義されています：

VFS	仮想ファイルシステム。例えば、モジュールのWebページを保持します。
Application	アプリケーション・ファイルシステムインターフェース・オブジェクト（EAh）（オプション）を介して、このディレクトリにあるアプリケーションのファイルシステムにアクセスできます。 このディレクトリはアプリケーションからアクセスできません。ネットワークからのみアクセス可能です。
Firmware	ファームウェアの更新ファイルはこのディレクトリに保存されます。

ファームウェアフォルダでは、ファイルの書き込み中に挿入モードは使用できません。 必ず書き込みモードだけ使用してください。

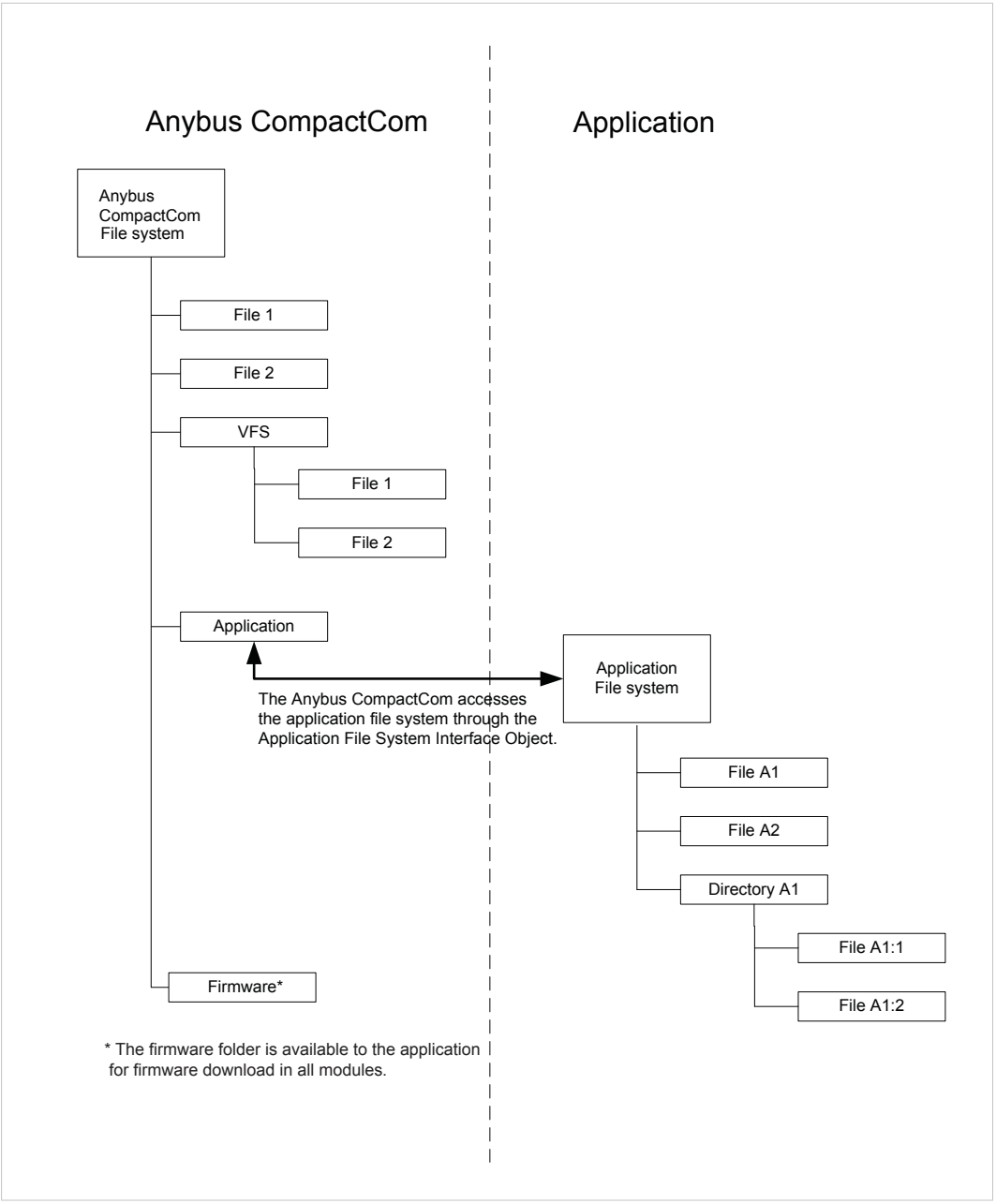

Fig. 6

下記も参照してください。

Anybus ファイルシステムインターフェース・オブジェクト *(0Ah), ページ 81*

アプリケーション・ファイルシステムインターフェース・オブジェクト *(EAh), ページ 122*

ファームウェアダウンロード, *ページ 24*

Anybus CompactCom 40 Network Guides、以下で入手可能 www.anybus.com

3.6 モジュラーデバイス

モジュラーデバイス機能により、アプリケーションの多様なタイプ (デジタル入出力、アナログ入出力、ドライブ、など) の多数のモジュール に対して、プロセスデータを構造化しモデル化することが可能になります。 複数のADIを 1 つのモジュールに割り当てられ、そのADI数は設定可能です。 モジュールは複数のスロットを持つバックプレーンに物理的に接続されます。 最初のスロットはAnybus CompactComモジュールを含む「カプラー」で占められます。 その他のすべてのスロットは、空であるか、またはモジュールで占められています。 ADIをプロセスデータにマッピングするとき、アプリケーションは、各モジュールのプロセスデータをスロット順にマッピングします。

下記も参照してください。

- モジュラーデバイスオブジェクト (*ECh*) , ページ *127*
- Anybus CompactCom 40 Network Guides

3.7 SYNC

3.7.1 概要

オートメーションシステムは数多くのデバイスで構成されており、イベントを同期させる仕組みが必要な場合があります。 これは、システム内のデバイスが共通のタイミング信号を使用することで実現できます。 Anybus CompactCom 40は、SYNCオブジェクトを用いたSYNCメカニズムをサポートしており、必要に応じてアプリケーションに実装できます。

以下のAnybus CompactCom 40モジュールがSYNC機能をサポートしています。

- Ethernet POWERLINK
- PROFINET-IRT
- EtherCAT

下記も参照してください。

- *SYNC* オブジェクト *(EEh)*, ページ *129*.
- アプリケーションステータスレジスタ, ページ *30*
- Anybus CompactComモジュールの各状態に関する情報については、*Anybus*ステートマシン, ページ *43*を参照してください。

3.7.2 機能

SYNCメカニズムを正しく実装するには、いくつかの実装/考慮すべきものがあります。

ネットワークのマスターは、IDLE状態またはPROCESS_ACTIVE状態になる前に、Anybus CompactComモジュールを介してSYNCオブジェクトのアトリビュート#1～3および#7を設定します。IDLE状態またはPROCESS_ACTIVE状態にてモジュールがアトリビュート#1～3を設定しようとした場合、アプリケーションはエラーコード0Dh（Invalid state）を返す必要があります。 アトリビュートでサポートされていない値が指定された場合、アプリケーションは適切なエラーコードを返す必要があります（11h（Value too high）、12h（Value too low）、0Ch（Out of range））。

コンフィグレーション全般の問題が発生した場合、アプリケーションは、アプリケーションステータスレジスタにてその旨を通知する必要があります。 アプリケーションステータスレジスタ, ページ 30を参照してください。

アプリケーションは、サポートする最小サイクル、および入出力処理に要する時間を、SYNCオブジェクトのアトリビュート#4～6にて常に通知する必要があります。 これらのアトリビュートには、現在のプロセスデータマッピングで要求されるタイミングを反映して、固定値と可変値のいずれも設定可能です。

3.7.3 同期ロック

アプリケーションがSYNC信号をロックするのに時間がかかる場合、アプリケーションステータスレジスタに0001h（同期していない）を書き込む必要があります。 同期ロックが確立し、コンフィグレーションエラーが存在しない場合、アプリケーションは、アプリケーションステータスレジスタに0000hを書き込み、PROCESS_ACTIVEへの遷移を許可する必要があります。

アプリケーションがSYNC信号にロックされておらず、SYNCオブジェクトのアトリビュート#7「Sync mode」が 1 に設定されている場合、アプリケーションは、最も適切な0以外のステータスコードをアプリケーションステータスレジスタに書き込む必要があります。

参照先

アプリケーションステータスレジスタ, ページ 30

3.7.4 SYNCパルス

SYNC信号（モジュールのアプリケーションコネクタから取得可能）は、各サイクルの立ち上がりパルスによって、同期イベントをアプリケーションに通知します。 同期イベントは、SYNCパルスの立ち上がりエッジにより通知されます。

SYNCパルスの最小幅は5 µsで、最大幅はサイクルタイムの50%です。

SYNCイベントは、マスカブル割り込みとしてアプリケーションで使用することも可能です。 割り込みステータスレジスタ, ページ 32を参照してください。

3.7.5 ネットワーク変換

Ethernet POWERLINKは、それ自体では同期機能をサポートしていません。 モジュールからのSYNC信号はサイクルごとに1回送信され、そのようなものとしてアプリケーションで使用できます。

Anybus CompactCom 40 EtherCATでは、パラメータと設定はCoEオブジェクトの1C32hと1C33hに格納されます。

Anybus CompactCom 40 PROFINET IRTは、アイソクロナスモードと非アイソクロナスモードの両方をサポートします。

詳細については、それぞれのネットワークガイドを参照してください。

3.7.6 Anybus CompactCom 40 SYNCの実装

SYNCの目標

- 出力データを異なるデバイスに同時に設定する。 言い換えると、これはPLCがネットワーク上のすべてのデバイスに次のように伝えることを意味します。「ここに次の出力データー式があります。SYNC信号が届いたら、これを各自のアプリケーションで設定してください」。

  出力データがアプリケーションで設定される時間のことをOutput Valid Pointと呼びます。

- 異なるデバイスから入力データを同時に取り込む。

  この時間のことをInput Capture Pointと呼びます。

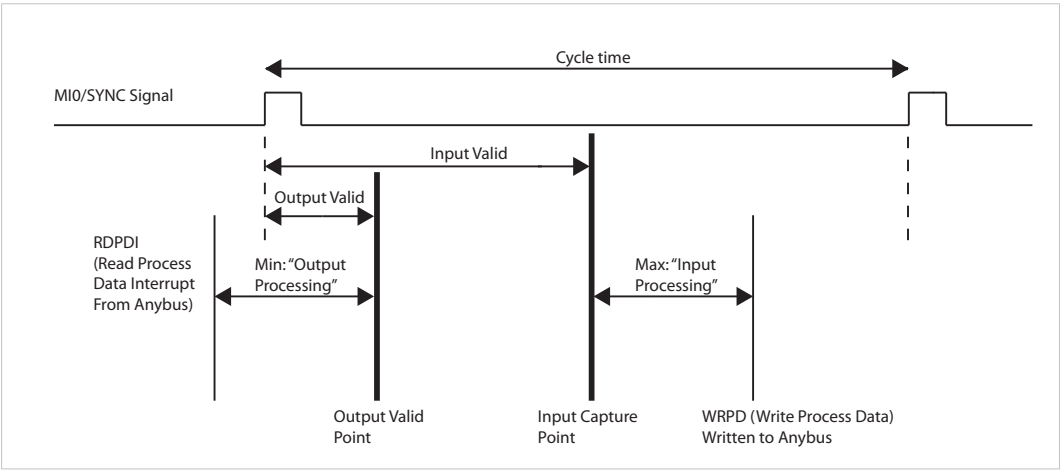

Fig. 7

出力データの処理

各デバイスでは、新しい出力データをアプリケーションで設定する前にそのデータを処理する時間が必要です。 この時間は一定ではなく小刻みに変動します。

デバイスで以下の手順を実行する必要があります。

1. 新しい出力データの通知を待ちます。これはAnybus CompactCom 40によってRDPDI（Read Process Data Interrupt：リードプロセスデータ割り込み）として通知されます。
2. RDPDIを受信したら、Anybus CompactCom 40から出力データを読み取ります。
3. 新しい出力データを処理して、ホストアプリケーションで使用できるように準備します（データをコピーする、出力変数を処理する、計算を行う、など）。
4. SYNC信号を待ちます。
5. Output Valid 時間が0の場合は、SYNC信号の受信時に出力をホストアプリケーションに対して有効化します。
6. あるいは（Output Valid 時間の値が0より大きい場合）、SYNC信号の立ち上がりエッジでハードウェアまたはソフトウェアタイマーを開始し、 output valid event を生成します。 タイマーが Output Valid pointを経過したら、出力をホストアプリケーションに対して有効にします。

RDPD（リードプロセスデータ）とRPDPI（リードプロセスデータ割り込み）の詳細については、バッファコントロールレジスタ, ページ 31および割り込みステータスレジスタ, ページ 32を参照してください。

入力データの処理

各デバイスでは、新しい入力データを取り込み、準備して送信するための時間が必要です。 この時間は一定ではなく小刻みに変動します。

デバイスで以下の手順を実行する必要があります。

1. SYNC信号を待ちます。
2. Input Valid 時間が0の場合は、SYNC信号の受信時に（可能な限り迅速に）ホストアプリケーションの現在の入力プロセス変数を取り込みます（Input Capture Point）。
3. あるいは（Input Valid 時間が0より大きい場合）、SYNC信号の立ち上がりエッジでハードウェアまたはソフトウェアタイマーを開始し、input capture event を作成します。 タイマーが Input Valid ポイントまで経過したら、現在の入力プロセスデータを取り込みます。
4. 新しい入力プロセスデータを処理します（Anybus CompactCom 40に書き込めるように準備します）。
5. 新しいプロセス入力データをAnybus CompactCom 40に書き込みます。

ホストアプリケーションのプログラミングガイドライン

Anybus CompactCom 40を使用してSYNCをサポートするには、いくつか検討して実装すべきことがあります。

詳細は*SYNC オブジェクト (EEh), ページ 129*を参照してください。

1. SYNCオブジェクトの実装（パート1）

7	Sync mode	Get/Set	UINT16	このアトリビュートは、同期モードの選択で使用されます。 これは、アトリビュート8のビットを列挙します。 0： 非同期動作。（同期動作がサポートされていない場合のデフォルト値） 1: 同期動作 2 - 65535: 予約。 サポートされていない同期モードの値に設定しようとすると、エラー応答が生成されます。
8	Supported sync modes	Get	UINT16	アプリケーションがサポートする同期モードのリストです。 各ビットは、アトリビュート7のモードに対応します。 ビット0: 1 = 非同期モードをサポート ビット1: 1 = 同期モードをサポート ビット2～15: 予約（0）

2. SYNCオブジェクトの実装（パート2）

4	Output processing	Get	UINT32	RDPDI割り込みから Output valid までに最低限必要な時間（単位：ナノ秒）。
5	Input processing	Get	UINT32	Input capture からAnybus CompactCom 40へのライトプロセスデータの書き込みが完了するまでに必要な最大時間（単位：ナノ秒）
6	Min cycle time	Get	UINT32	アプリケーションがサポートする最小サイクルタイム。

RDPDI割り込みを受信してからプロセス出力変数（リードプロセスデータ）がアプリケーションに引き継がれるまでに経過した時間を測定する必要があります。 Anybus CompactCom 40からアトリビュート#4 Output processing が要求されたとき、アプリケーションは、この最大時間を提供する必要があります。

入力プロセス変数を取り込んでから入力プロセスデータ（ライトプロセスデータ）がCompactCom 40に書き込まれるまでに経過した時間を測定する必要があります。 Anybus CompactCom 40からアトリビュート#5 Input processing が要求されたとき、アプリケーションは、この最大時間を提供する必要があります。

ホストアプリケーションは、すべてのプロセスデータの処理に要した最大時間（RDPDI割り込みを受信してから入力プロセスデータがAnybus CompactCom 40に書き込まれるまでの時間）を測定する必要があります。 Anybus CompactCom 40からアトリビュート#6 Min cycle time が要求されたとき、アプリケーションはこの値を提供する必要があります。

3. SYNCオブジェクトの実装（パート3）

1	Cycle time	Get/Set	UINT32	アプリケーションのサイクルタイム（単位：ナノ秒）。
2	Output valid	Get/Set	UINT32	SYNCイベント発生からOutput valid pointまでの時間（単位：ナノ秒）。 デフォルト値： 0
3	Input capture	Get/Set	UINT32	SYNCイベント発生からInput capture pointまでの時間（単位：ナノ秒）。 デフォルト値： 0

これら3つのアトリビュートはすべて、Anybus CompactCom 40によって設定されます。

Anybus CompactCom 40はアトリビュート#1 Cycle time を使用して、（Set_Attributeコマンドによって）実際に使用されるサイクルタイムをホストアプリケーションに通知します。 ホストアプリケーションはこの値を確認し、受け入れられない場合（例えば、ホストアプリケーションの他のサイクリックタスクと競合する、定義範囲から外れている、などの理由で値が適していない場合）は拒否する必要があります。 拒否された場合、Anybus CompactCom 40はこれをPLCに報告します。

アトリビュート#2と#3は、一部のネットワーク（例：EtherCAT、SERCOS、PROFINET）に見られるInput Valid pointとOutput Valid pointを微調整できる機能を反映しています。 これを使用して、あるデバイスを別のデバイスに対してわずかな時間オフセットできます。 ゼロ（0）以外の値をサポートするには、アプリケーションにタイマーを実装する必要があります。

4. アプリケーションステータスレジスタの実装

詳細はアプリケーションステータスレジスタ, ページ 30を参照してください。

5. RDPDIおよびSYNC信号を受信したときに正しい処理を行う

RDPDI割り込みを受信したときは、Anybus CompactCom 40から出力プロセスデータ（リードプロセスデータ）を読み取り、SYNC信号の受信時に有効にできるように準備します（出力プロセスデータを処理してプロセス出力変数に割り当てます）。

SYNC信号を受信したときは、以下のことを行います。

a. 出力プロセス変数を直ちにアプリケーションに引き継ぎます。

b. すべての入力プロセス変数を直ちに取り込みます。

c. 入力プロセス変数を処理して入力プロセスデータに割り当てます。

d. 入力プロセスデータ（ライトプロセスデータ）をCompactComに書き込みます。

ステップ2、3、4はアトリビュート#5 Input processing によって指定された時間（Input captureからの時間）内に行う必要があります。

ステップ1と2は Output valid および Input capture がゼロ（0）であることを前提とします。

同期アプリケーションの最も簡単な実装

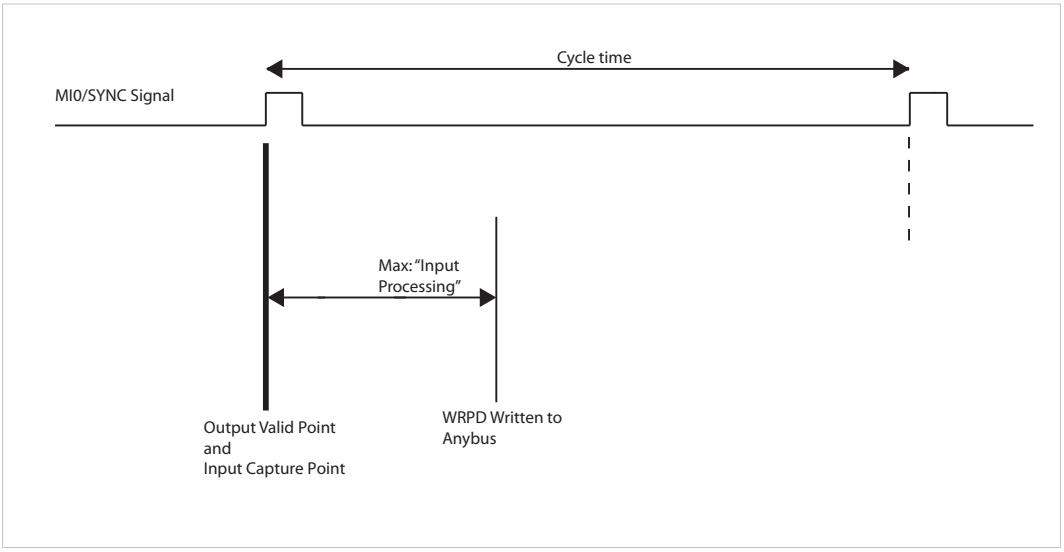

Fig. 8

以下の手順は、非常に単純な同期アプリケーションを作成する方法を示します。

1. SYNCの立ち上がりエッジによってトリガーされる割り込みルーチンをセットアップします。RDPDI割り込みを無効にします。
2. SYNC割り込みルーチンで以下のことを行います。
   - 入力データをサンプリングしてAnybus CompactCom 40に書き込みます。
   - Anybus CompactCom 40から出力データを読み取り、それを直ちに使い始めます。
3. このアプリケーションでは、SYNCオブジェクトのアトリビュート4 Output processing をゼロ（0）に設定します。 測定は必要ありません。
4. アトリビュート#5 Input processing を決定する必要があります。 おそらく固定値にハードコーディングできますが、これはアプリケーションに固有であり、SYNC割り込みルーチンの複雑さとアプリケーションプロセッサーの性能に依存します。
5. アトリビュート#2 Output valid とアトリビュート#3 Input capture については、値ゼロ（0）のみがアプリケーションで受け入れられます。

この単純なステップバイステップの方法は、プロセスデータ処理を迅速かつシンプルに実行するすべてのアプリケーションで動作します。

3.8 多言語サポート

Anybus CompactCom 40は、必要に応じて複数の言語をサポートします。 これは主に、インスタンス名と列挙文字列に影響を与え、Anybusオブジェクトの現在の言語設定に基づいて設定されます。 なお、この設定はホストアプリケーションオブジェクトにも適用されます。そのため、ホストアプリケーションは、この設定に従って列挙文字列などの言語の変更が行える必要があります。

Anybus CompactCom 40は、必要に応じて、ネットワークから送られた言語変更要求をアプリケーションオブジェクトに転送します。 これにより、モジュールが言語設定を変更するのか、ネットワークから送られた元の要求を拒絶するのかが決まります。

サポートする言語：

- 英語（デフォルト）
- ドイツ語
- スペイン語
- イタリア語
- フランス語

下記も参照してください。

アプリケーションオブジェクト（*FFh*）, ページ *114*

3.9 ファームウェアダウンロード

特定のフィールドバスまたは産業用ネットワークで使用されるネットワーク通信ファームウェアをダウンロードおよびアップグレードするには、どのAnybus CompactCom 40をアップグレードするかに応じて方法が異なります。

3.9.1 重要：

ファームウェアのダウンロード後にAnybus CompactCom 40が再起動するとき、アプリケーションはアップグレードが完了するまで初期化の開始を待つ必要があります。 Anybus CompactCom 40はダウンロード中および/またはインストール中の電断に対する保護機能を持つため、再起動すると元の状態に戻ります。

- Firmware ManagerまたはFTPなどを使用したダウンロードが中断された場合、ファームウェアのダウンロード手続きを初めからやり直してください。
- ダウンロード完了後に新しいファームウェアをインストールするには、Anybus CompactCom 40をリセットします。 新しいファームウェアのインストールが電断などによって中断された場合は、Anybus CompactCom 40を再起動してください。 インストールプロセスが最初から自動的に開始され、特に操作を行わなくても新しいファームウェアがインストールされます。

詳細については、起動手順, ページ *57*を参照してください。

3.9.2 Firmware Manager IIの使用

このツールはwww.anybus.comから無料で提供されます。 どのAnybus CompactCom 40についても、このツールを使用して新しいファームウェアをダウンロードすることが可能です。

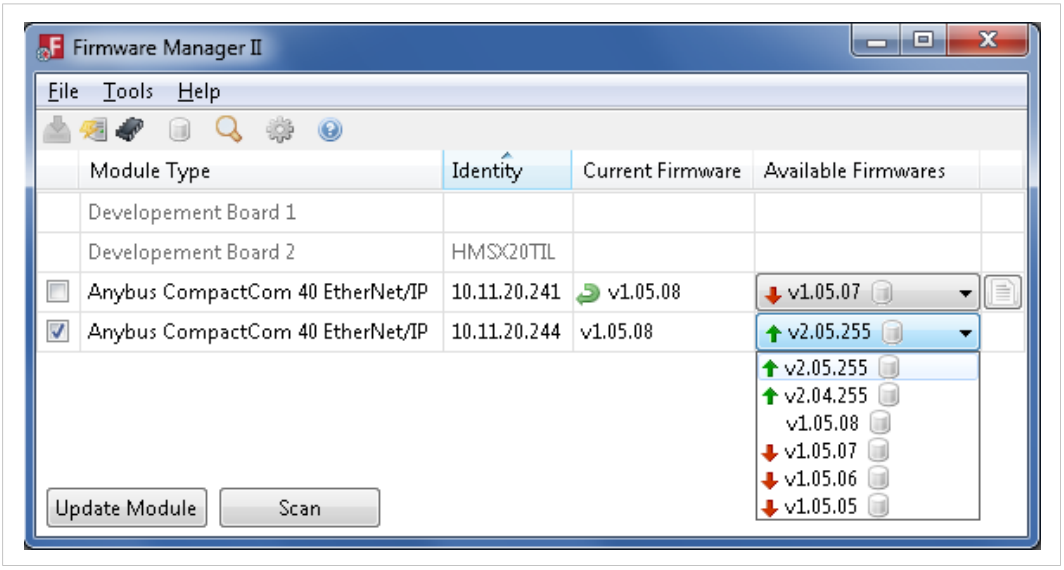

Fig. 9

このツールを使用して新しいファームウェアをモジュールにダウンロードするには、以下の手順に従います。

1. Firmware Manager IIソフトウェアをインストールしたコンピュータを対象のモジュールを含むネットワークに接続します。
2. Firmware Manager IIツールを起動します。
3. ネットワークをスキャンしてモジュールを検出します。
4. メニューにある Firmware Repository アイコンをクリックし、Firmware Repository ウィンドウを開きます。 ファームウェアフォルダをこのウィンドウにドラッグして、新しいファームウェアをリポジトリに追加します。 Firmware Repository ウィンドウを閉じます。
5. スキャンウィンドウの Available Networks タブで、当該のモジュールに適合するファームウェアを選択します。 **Change Network**ボタンをクリックします。 確認ウィンドウが表示されます。 **Yes**をクリックして、新しいファームウェアのダウンロードを開始します。 そして、ダウンロードが完全に終了したことを確認してください。
6. ダウンロードが完了したら、新しいファームウェアをインストールするために、モジュールを再起動する必要が有ります。 アプリケーションで許可されている場合、Firmware Manager IIツールの**Restart Module**ボタンを使用してモジュールを再起動できます。 アプリケーションでネットワークからの再起動が許可されていない場合は、モジュールを手動で再起動する必要があります。

詳細については、Firmware Manager IIソフトウェアのヘルプファイルを参照してください。

3.9.3 内部ファイルシステムの使用

ファイルシステムインターフェースオブジェクトを介して内部ファイルシステムにアクセスできます。 ホストアプリケーションからこのオブジェクトを介して、新しいファームウェアを内部ファイルシステムの/firmwareディレクトリに保存できます。 次回モジュールが起動したときにファームウェアがアップグレードされます。 ファームウェアがインストールされると、ファームウェアファイルは/firmwareディレクトリから削除されます。

> ファームウェアフォルダでは、ファイルの書き込み中に挿入モードは使用できません。 必ず書き込みモードだけ使用してください。

下記も参照してください。

- アプリケーション・ファイルシステムインターフェース・オブジェクト *(EAh), ページ 122*

3.9.4 FTPの使用

モジュールがFTPをサポートしている場合は、FTPを使用してファイルシステムにアクセスし、新しいファームウェアを/firmwareディレクトリに直接アップロードできます。 次回モジュールが起動したときにファームウェアがアップグレードされます。 ファームウェアがインストールされると、ファームウェアファイルは/firmwareディレクトリから削除されます。

下記も参照してください。

- アプリケーション・ファイルシステムインターフェース・オブジェクト *(EAh), ページ 122*

4 ホスト通信レイヤー

4.1 概要

メインの通信レイヤーは、8ビット/ 16ビットパラレルモードおよびSPIモードで使用されます。 このレイヤーは、プロセスデータのリード/ライト領域、メッセージデータのリード/ライト領域、および各種コントロールレジスタに分かれています。

以下、メモリマップおよび各種コントロールレジスタについて説明します。

4.1.1 通信の基本

ホストとAnybus CompactCom 40との通信は、簡単、高速、かつ柔軟です。

ホストは、プロセスデータをいつでもリード/ライトできます。 ホストは、バッファコントロールレジスタを使用するか、割り込みマスクレジスタにて該当する割り込みを有効にすることで、送信されてくるデータをチェックできます。

> **!** 予約レジスタにアクセスしようとすると、予期せぬ結果になります。 リードオンリーのレジスタに書き込みを行おうとすると、予期せぬ結果になります。 予約ビットにはゼロ（0）を書き込んでください。 予約ビットを読み込むと、不定値が返されます。

4.2 メモリマップ

以下に示すアドレスオフセットは、モジュールのベースアドレスからの相対アドレスです。すなわち、ホストアプリケーションのメモリ空間にてインターフェースが実装されている領域の先頭からの相対アドレスです。

メモリ領域はリセット中は使用できません。

グレーで表示されている領域とレジスタは、Anybus CompactCom 30シリーズとの後方互換性のために使用されます。

バイトアドレス	ワードアドレス	領域	バイト数	アクセス（ホストから見た場合）	メモ
0000h - 0FFFh	0000h - 07FFh	プロセスデータライト領域	4096	R/W	これは領域の全体サイズです。実際のサイズはネットワークによって異なります。 ネットワーク固有のプロセス領域のサイズについては、ネットワーク比較, ページ 138を参照してください。すべてのプロトコルに適用
1000h - 1FFFh	0800h - 0FFFh	プロセスデータリード領域	4096	R	
2000h - 25FFh	1000h - 12FFh	メッセージデータライト領域	1536	R/W	すべてのプロトコルに適用
2600h - 2FFFh	1300h - 17FFh	予約	2560	-	
3000h - 35FFh	1800h - 1AFFh	メッセージデータリード領域	1536	R	
3600h - 37FFh	1B00h - 1BFFh	予約	512	-	
3800h - 38FFh	1C00h - 1C7Fh	プロセスデータライト領域	256	R/W	
3900h - 39FFh	1C80h - 1CFFh	プロセスデータリード領域	256	R	
3A00h - 3AFFh	1D00h - 1D7Fh	予約	256	-	
3B00h - 3C06h	1D80h - 1E03h	メッセージデータライト領域	263	R/W	
3C07h - 3CFFh	1E04h - 1E7Fh	予約	249	-	
3D00h - 3E06h	1E80h - 1F03h	メッセージデータリード領域	263	R	
3E07h - 3FE7h	1F04h - 1FF3h	予約	481	-	
3FE8h - 3FEFh	1FF4h - 1FF7h	現在のネットワーク時間	8	R	
3FF0h - 3FF1h	1FF8h	モジュール機能レジスタ	2	R	
3FF2h - 3FF3h	1FF9h	LEDステータスレジスタ	2	R	
3FF4h - 3FF5h	1FFAh	アプリケーションステータスレジスタ	2	R/W	イベント駆動型プロトコルに適用。
3FF6h - 3FF7h	1FFBh	Anybus CompactComモジュールステータスレジスタ	2	R	
3FF8h - 3FF9h	1FFCh	バッファコントロールレジスタ	2	R/W	
3FFAh - 3FFBh	1FFDh	割り込みマスクレジスタ	2	R/W	
3FFCh - 3FFDh	1FFEh	割り込みステータスレジスタ	2	R/W	
3FFEh	1FFFh	コントロールレジスタ	1	R/W	半二重プロトコルに適用
3FFFh		ステータスレジスタ	1	R	

アプリケーションで「現在のネットワーク時間」を使用する場合は、まずネットワーク時間をサンプリングする必要があります。 これはこのレジスタへの書き込みによって実行されます。 レジスタがネットワークからの実際の値で更新された後、その値をアプリケーションから読み取ることが可能です。

*C*プログラマーの方は、共有メモリ全体を*volatile*であると指定することを忘れないようにしてください。

4.3 通信レジスタ

4.3.1 モジュール機能レジスタ

モジュール機能レジスタには、モジュールの種類を表す以下のいずれかの値が格納されます。 アプリケーションは、下位バイトのみを見てモジュールの種類を判断します。 上位バイトは将来のために予約されています。

値	モジュールの種類
0000h	Anybus CompactCom 30シリーズのアクティブモジュール。半二重プロトコルのみサポート 8ビットパラレルおよびシリアルモード
000Bh	Anybus CompactCom 40シリーズのアクティブモジュール。半二重およびイベント駆動型のプロトコルをサポート 8ビット/ 16ビットパラレル、シフトレジスタ、SPI、およびシリアルモード
000Ch	パラレルAXIバスを用いたAnybus IP
0101h	パッシブRS232
0202h	パッシブRS422
0303h	パッシブUSB
0404h	(予約)
0505h	パッシブBluetooth
0606h - 0909h	(予約)
0A0Ah	パッシブRS485
0C0Ch - 0F0Fh	(予約)

すべてのパッシブモジュールは*Anybus CompactCom 30*シリーズに属します。 *Anybus CompactCom 40*シリーズにはパッシブモジュールはありません。

4.3.2 LEDステータスレジスタ

このレジスタの最初の8ビットには、Anybusオブジェクトのインスタンスアトリビュートであ るLEDステータス (#13) の値で表されるLEDの状態が反映されます。 詳細は*Anybusオブジェクト (01h), ページ 62*を参照してください。 それ以降のビットは信号LED5A、LED5B、LED6A、LED6Bの状態を反映します。

ビットが1に設定されていると、対応するLEDの状態はオンになっています。

ビット	**LED信号**
0	LED1A
1	LED1B
2	LED2A
3	LED2B
4	LED3A
5	LED3B
6	LED4A
7	LED4B
8	LED5A
9	LED5B
10	LED6A
11	LED6B
12 ~ 15	(なし)

4.3.3 アプリケーションステータスレジスタ

アプリケーションステータスレジスタは、主に、SYNCアプリケーションで使用されます。 これは、対象のネットワークがプロセスデータの致命的なエラーをマスターに通知する機能をサポートするアプリケーションで使用されます。 この機能がサポートされている場合、Anybus CompactCom 40モジュールは、アプリケーションにより書き込まれた以下のステータスコードを受け付けて処理します。

このレジスタの使用は任意です。 プロセスデータの致命的なエラーをサポートしないネットワークの場合、モジュールはこのレジスタを無視します。

ステータスコード	エラー	説明
0000h	エラーなし	PROCESS_ACTIVE状態に遷移する準備ができている（デフォルト）。
0001h	同期していない	PROCESS_ACTIVE状態に遷移する準備ができていない。
0002h	SYNCコンフィグレーションエラー	SYNCオブジェクトのアトリビュート値に問題があるため、PROCESS_ACTIVE状態に遷移できない。
0003h	リードプロセスデータコンフィグレーションエラー	リードプロセスデータのマッピングに問題があるため、PROCESS_ACTIVE状態に遷移できない。
0004h	ライトプロセスデータコンフィグレーションエラー	ライトプロセスデータのマッピングに問題があるため、PROCESS_ACTIVE状態に遷移できない。
0005h	同期喪失	アプリケーションが同期ロックを失った Anybus CompactComがPROCESS_ACTIVE状態になっている場合は、下位の状態に遷移します。
0006h	過度のデータ消失	アプリケーションにより、ネットワークからのプロセスデータが大量に失われていることが検出された。 Anybus CompactComがPROCESS_ACTIVE状態になっている場合は、下位の状態に遷移します。
0007h	出力エラー	アプリケーションの誤動作。 Anybus CompactComがPROCESS_ACTIVE状態になっている場合は、下位の状態に遷移します。

Anybusステートマシンについては*Anybusステートマシン, ページ 43*で解説しています

4.3.4 Anybus CompactComモジュールステータスレジスタ

このレジスタには、Anybus CompactComモジュールの現在の状態、およびAnybus CompactComモジュールにより通知された監視ビットが格納されます。 Anybusステートマシンについては*Anybusステートマシン, ページ 43*で解説しています

ビット	名前	説明
0 ~ 2	S[0~2]	Anybus CompactComモジュールの現在の状態
3	監視ビット	1 =他のネットワークデバイスにより監視されている 0 =他のネットワークデバイスにより監視されていない 監視ビット（*SUP*）, ページ *34*を参照してください。
4 ~ 15	-	予約（0）

4.3.5 バッファコントロールレジスタ

このレジスタは、Anybus CompactComモジュールとのイベント駆動型通信をアプリケーションが制御するのに使用します。

このレジスタに書き込みを行うことで、該当するイベントをトリガーできます。 1と書き込むとそのビットがトリガーされます。0と書き込んだ場合、ビットは何も影響を受けません。

このレジスタを読み出すと、各種メモリ領域の現在のステータスが得られます。

ビット0～3の実装方法と使用方法に関する詳細は、通信の基本, ページ 35を参照してください。

ビット	名前	説明
0	WRPD （Write Process Data）	アプリケーションは、新しいデータを書き込んだとき、このビットに1を書き込む。
1	RDPD （Read Process Data）	モジュールがリードプロセスデータを更新したとき、このビットが1として読み込まれる。 アプリケーションはこのビットに1を書き込んで、最新のリードプロセスデータを要求する。 これによりビットがクリアされる。 注：「更新された」データは、必ずしも「変化した」データを意味するものではありません。
2	WRMSG （Write Message Data）	メッセージデータライト領域が占有されていると、このビットが1として読み込まれる。 このビットは、メッセージデータライト領域に新規メッセージを格納できるようになったとき、モジュールによりクリアされる。 アプリケーションがライトメッセージをモジュールに送信するとき、このビットに1を書き込む。 アプリケーションは、このビットが0のときにのみメッセージデータライト領域に書き込むことができる。 注：: ANBRビットも設定されているときにのみ、コマンドメッセージの書き込みが可能。
3	RDMSG （Read Message Data）	モジュールが新規リードメッセージをポストしたとき、このビットが1として読み込まれる。 アプリケーションは、このビットに1を書き込んでメッセージを受領したことをモジュールに通知する。 これによりビットがクリアされる。 アプリケーションは、このビットが1のときにのみメッセージデータリード領域の読み込みができる。
4	ANBR （Anybus Ready）	モジュールによる新規コマンドメッセージの受信準備が整うと、このビットが1に設定される。 アプリケーションは、このビットが1のときにのみコマンドメッセージを送信できる。 このビットは、WRMSGが 1のときにのみ1から0に変更される。 0から1へはいつでも変更可能。
5	APPR （Application Ready）	アプリケーションは、新規コマンドメッセージの受信準備ができたとき、このビットに1を書き込む。 モジュールは、このビットが1のときにのみコマンドメッセージを送信する。 このビットを0に設定するにはAPPRCLRを使用する。
6	APPRCLR （Application Ready Clear）	アプリケーションは、このビットに1を書き込むことで、APPRビットをクリアできる。 この操作はRDMSGが1のときにのみ可能。
7～15	-	予約

4.3.6 割り込みマスクレジスタ

このレジスタにより、アプリケーションは以下の表に従って割り込みを個別に有効/無効にすることができるようになります。

ビット	名前	説明
0	RDPDIEN	このビットを1に設定すると、バッファコントロールレジスタのRDPDビットが0から1に変化したときに発生する割り込みが有効になる。
1	RDMSGIEN	このビットを1に設定すると、バッファコントロールレジスタのRDMSGビットが0から1に変化したときに発生する割り込みが有効になる。
2	WRMSGIEN	このビットを1に設定すると、バッファコントロールレジスタのWRMSGビットが1から0に変化したときに発生する割り込みが有効になる。
3	ANBRIEN	このビットを1に設定すると、バッファコントロールレジスタのANBRビットが0から1に変化したときに発生する割り込みが有効になる。
4	STATUSIEN	このビットを1に設定すると、Anybus CompactComモジュールのステータスレジスタの値が変化したときに発生する割り込みが有効になる。
5	-	予約
6	SYNCIEN	このビットを1に設定すると、SYNCイベントの割り込みが有効になる。
7～15	-	予約

4.3.7 割り込みステータスレジスタ

このレジスタにより未処理状態の割り込みを表します。

ビット	名前	説明
0	RDPDI	バッファコントロールレジスタのRDPDが0から1に変化したとき、このビットが1に設定される。 このビットを0に設定するには、アプリケーションにて、このビットに1を書き込む。
1	RDMSGI	バッファコントロールレジスタのRDMSGが0から1に変化したとき、このビットが1に設定される。 このビットを0に設定するには、アプリケーションにて、このビットに1を書き込む。
2	WRMSGI	バッファコントロールレジスタのWRMSGが1から0に変化したとき、このビットが1に設定される。 このビットを0に設定するには、アプリケーションにて、このビットに1を書き込む。
3	ANBRI	バッファコントロールレジスタのANBRが0から1に変化したとき、このビットが1に設定される。 このビットを0に設定するには、アプリケーションにて、このビットに1を書き込む。
4	STATUSI	Anybus CompactComモジュールのステータスレジスタの値が変化すると、このビットが1に設定される。 このビットを0に設定するには、アプリケーションにて、このビットに1を書き込む。
5	PWRI	起動またはハードウェアリセット後、モジュールの通信準備が整うと、このビットが1に設定される。 このビットを0に設定するには、アプリケーションにて、このビットに1を書き込む。
6	SYNCI	SYNCイベントが発生するたびに、このビットが1に設定される。 このビットを0に設定するには、アプリケーションにて、このビットに1を書き込む。
7 ~ 15	-	予約

4.3.8 コントロールレジスタ（リード/ライト）

半二重（ピンポン）プロトコルでのみ使用。

このレジスタは、Anybus CompactComに対する通信を制御します。

b7（MSB）	**b6**	**b5**	**b4**	**b3**	**b2**	**b1**	**b0（LSB）**
CTRL_T	CTRL_M	CTRL_R	CTRL_AUX	-	-	-	-

ビット	説明
CTRL_T	ホストアプリケーションは、新規テレグラムを送信するときにこのビットをトグルすること。 モジュールに対してアプリケーションから最初に送信するテレグラムでは、CTRL_Tを1に設定する必要がある。
CTRL_M	セットすると、現在のテレグラムのメッセージサブフィールドが有効になる。
CTRL_R	セットすると、ホストアプリケーションが新規コマンドを受信できるようになる。
CTRL_AUX	（無視）
-	(予約、ゼロに設定)

4.3.9 ステータスレジスタ（リードオンリー）

半二重（ピンポン）プロトコルでのみ使用。

このレジスタには、Anybus CompactComの現在の状態が格納されます。

b7（MSB）	**b6**	**b5**	**b4**	**b3**	**b2**	**b1**	**b0（LSB）**
STAT_T	STAT_M	STAT_R	STAT_AUX	SUP	S2	S1	S0

ビット	説明			
STAT_T	このビットは、モジュールが新規テレグラムを発行したとき、ホストアプリケーションから最後に受信したテレグラムのCTRL_Tと同じ値に設定される。			
STAT_M	セットすると、現在のテレグラムのメッセージサブフィールドが有効になる。			
STAT_R	セットすると、Anybusモジュールが受信コマンドを処理できるようになる。			
STAT_AUX	補助ビット（*STAT_AUX*、 *CTRL_AUX*）, ページ *34*を参照してください。			
SUP	値 意味 0 : モジュールは監視されていない。 1: モジュールは他のネットワークデバイスによって監視されている。 監視ビット（*SUP*）, ページ *34*を参照してください。			
S[0~ 2]	これらのビットは、モジュールの現在の状態を表す（*Anybus*ステートマシン, ページ *43*を参照）。			
	S2	S1	S0	Anybusの状態
	0	0	0	SETUP
	0	0	1	NW_INIT
	0	1	0	WAIT_PROCESS
	0	1	1	IDLE
	1	0	0	PROCESS_ACTIVE
	1	0	1	ERROR
	1	1	0	(予約)
	1	1	1	EXCEPTION

ステータスレジスタは、起動時にクリアされているとみなします。 そのため、ホストアプリケーションから最初に発行されるテレグラムでは、STAT_Rが事実上0にクリアされているため、このテレグラムには有効なメッセージサブフィールドを含めてはなりません。

4.3.10 監視ビット（SUP）

Anybusのステートマシンがサイクリックなデータ交換に関する状態を反映するのに対し、SUPビットは、アサイクリックな通信を含むネットワーク通信の全体的な状態を表します。 例えばCIPでは、このビットはマスターから本モジュールへのコネクションが確立されていることを表します。 このコネクションは、I/Oコネクション、またはアサイクリック（明示的）なコネクションの場合があります。 後者の場合、通信はある一定期間「無通信」になり、その間、ステートマシンはネットワークがアイドル状態であることを表します。 一方、SUPビットは、モジュールに対するコネクションが継続されていることを表します。

この機能が実際にどのように処理されるかや、提供されるセキュリティレベル、およびそれを正しく作動させる方法は、ネットワークごとに異なり、ネットワークのコンフィグレーションや他のネットワークデバイスなどの外部環境に依存することがしばしばあります。 そのため、SUPビットの使用が適切かどうかは、アプリケーションやネットワークごとに決める必要があります。

4.3.11 補助ビット（STAT_AUX、 CTRL_AUX）

Anybus CompactCom 40モジュールは、コントロールレジスタのCTRL_AUXビットを無視します。

本モジュールは、最後のテレグラム受信後、ネットワークから新規プロセスデータを受信すると、ステータスレジスタのSTAT_AUXビットをセットします。

5 パラレルホスト通信

5.1 フロー制御

以下の説明は、イベント駆動モード (全二重モード) にのみ当てはまります。 半二重モードに関する情報は、 シリアルホスト通信 (*UART*) , ページ *42*を参照してください。

データは、その場所と順序にかかわらず、ホストまたはAnybus CompactComモジュールのいずれからもリード/ライトできます。 通信は、バッファコントロールレジスタをリード/ライトするか、割り込みマスクレジスタにて該当するイベントの割り込みを有効にすることで、完全に制御することが可能です。 割り込みを有効にすると、モジュールが新規データを提供するたびに割り込みが生成されます。

バッファコントロールレジスタ, ページ *31*および 割り込みマスクレジスタ, ページ *31*を参照してください。

5.1.1 通信の基本

パラレスホストインターフェースを使用する場合、共有メモリ領域を介してデータが交換されます。 詳細については、メモリマップ, ページ *28*を参照。

データ送信

プロセスデータを書き込むには以下のようにします。

1. ライトプロセスデータのメモリ領域にデータを書き込みます。 ADIによりプロセスデータ用にマッピングされている領域は、新規データで更新する必要があります。
2. バッファコントロールレジスタのビット0 (WRPD) に1を書き込み、プロセスデータ書き込み処理を完了します。

メッセージデータを書き込むには以下のようにします。

1. バッファコントロールレジスタのビット2 (WRMSG) を読み込みます。
   - このビットが0の場合、該当する領域に新規メッセージデータを書き込むことが可能です。
   - このビットが1の場合、該当する領域は占有されており、新規メッセージデータの送信にはまだ使用できません。
2. ライトメッセージデータのメモリ領域にデータを書き込みます。
3. バッファコントロールレジスタのビット2 (WRMSG) に1を書き込み、ライト処理を確定します。

データ受信

最新のリードプロセスデータを受信するには以下のようにします。

1. プロセスデータにアクセスするには、バッファコントロールレジスタのビット1 (RDPD) に1を書き込みます。
2. リードプロセスデータ領域から最新のリードプロセスデータを読み込みます。

最新のリードメッセージデータを受信するには以下のようにします。

1. バッファコントロールレジスタのビット3 (RDMSG) を読み込みます。
   - このビットが0の場合、新規メッセージはポストされていません。
   - このビットが1の場合、リードメッセージデータ領域に新規メッセージが存在します。
2. リードメッセージデータ領域から最新のメッセージデータを読み込みます。
3. バッファコントロールレジスタのビット3 (RDMSG) に1を書き込みます。

5.2 Anybusのイベント駆動型ウォッチドッグ

ホストは、メッセージの応答時間を定期的に測定することで、Anybus CompactComモジュールが正しく動作しているかどうかを判断できます。 この時間が指定した値を超えた場合、モジュールは正しく動作していないとみなせます。 このときホストは、アプリケーション固有のセーフ状態になるか、モジュールをリセットするか、またはそれらと同様の動作を行うことができます。

必要に応じてモジュールを再起動できるように、少なくとも基本的なウォッチドッグメカニズムを実装することを強く推奨します。

5.3 アプリケーションのイベント駆動型ウォッチドッグ

アプリケーションから要求される場合、Anybus CompactComモジュールにてアプリケーションに対するウォッチドッグタイムアウト監視を有効にできます。 このタイムアウトを有効にすると、本モジュールは、ライトプロセスデータバッファの更新間隔がアプリケーションで選択されたウォッチドッグタイムアウト時間を超えたときに、アプリケーションが正しく動作していないとみなします。

アプリケーションのウォッチドッグタイムアウトは、Anybusオブジェクトのインスタンスアトリビュート#4（アプリケーションのウォッチドッグタイムアウト）で指定します。 *Anybusオブジェクト (01h), ページ 62*を参照してください。

6 SPIホスト通信

6.1 概要

SPI（Serial Peripheral Interface）は、シリアルの全二重プロトコルです。 これはマスター/スレーブモードで、ホストがマスターとして動作し、Anybus CompactComモジュールがスレーブとして動作します。

SPIフレームの各バイトは最上位ビットから転送されますが、バイトの順序はリトルエンディアンになっています。 最下位バイトから転送されます。 エラーは、32ビットのCRCを用いて検出します。

6.2 SPIのフレームフォーマット

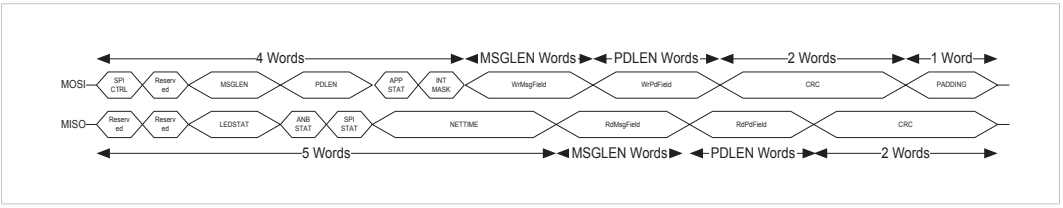

Fig. 10

6.2.1 MOSI（Master Output、Slave Input）フレームのデータ定義

SPI MOSIのフレームフォーマット		
バイト	名前	説明
0	SPI CTRL	SPIコントロールバイト。以下のSPIコントロールバイトの表を参照。
1	(予約)	
2 ~ 3	MSGLEN	WrMsgFieldおよびRdMsgFieldフィールドのサイズ、ワード単位。
4 ~ 5	PDLEN	WrPdFieldおよびRdPdFieldフィールドのサイズ、ワード単位。
6	APP STAT	アプリケーションステータス。アプリケーションステータスレジスタ, ページ 30を参照。
7	INT MASK	割り込みマスク。割り込みマスクレジスタ, ページ 31を参照。
MSGLEN words	WrMsgField	メッセージフィールド
PDLEN words	WrPdField	ライトプロセスデータフィールド
2ワード	CRC	-
1ワード	PADDING	ダミーデータ

SPIコントロールバイト		
ビット	名前	説明
0	WRPD VALID	このビットがセットされているとき、Anybus CompactCom 40はライトプロセスデータフィールドの内容に従って動作する。 このビットがセットされていないとき、モジュールはライトプロセスデータフィールドの内容を無視する。
1 ~ 2	CMDCNT	これらの2つのビットは、アプリケーションが受信可能なコマンド数を表す。 00 =アプリケーションはコマンド受信準備ができていない 01 =アプリケーションは少なくとも1つのコマンドを受信する準備ができている 10 =アプリケーションは少なくとも2つのコマンドを受信する準備ができている 11 =アプリケーションは少なくとも3つのコマンドを受信する準備ができている
3	M	セットされているとき、メッセージフィールドにメッセージが格納されている。
4	LAST FRAG	セットされているとき、メッセージの最後のフラグメントがメッセージフィールドに格納されている。
5 ~ 6	-	予約、0に設定。
7	TOGGLE	初回送信時、このビットを1に設定すること。 SPI転送を行うたびにこのビットを反転すること。 注： CRCエラーが検出された場合は、再送を表すために、このビットを反転しないでください。

6.2.2 MISO（Master Input、Slave Output）フレームのデータ定義

SPI MOSIのフレームフォーマット		
バイト	名前	説明
0 ~ 1	(予約)	
2 ~ 3	LEDSTAT	LEDステータス。*LEDステータスレジスタ, ページ 29*を参照。
4	ANB STAT	Anybus CompactComモジュールステータス。*Anybus CompactComモジュールステータスレジスタ, ページ 30*を参照。
5	SPI STAT	SPIステータス。下記のSPIステータスバイトの表を参照。
6 ~ 9	NETTIME	この4バイトには、ネットワーク時間の下位32ビットが格納される。
MSGLEN words	RdMsgField	メッセージフィールド
PDLEN words	RdPdField	リードプロセスデータフィールド
2ワード	CRC	-

SPIステータスバイト		
ビット	名前	説明
0	WRMSG FULL	セットされているとき、ライトメッセージバッファが一杯になっている。 MOSIフレームにメッセージが存在した場合、そのメッセージはAnybus CompactCom 40により無視されているため、再度送信する必要がある。 重要： この場合、トグルビットは反転させる必要があります。
1 ~ 2	CMDCNT	これらの2つのビットは、モジュールが受信可能なコマンド数を表す。 00 =モジュールはコマンド受信準備ができていない 01 =モジュールは少なくとも1つのコマンドを受信する準備ができている 10 =モジュールは少なくとも2つのコマンドを受信する準備ができている 11 =モジュールは少なくとも3つのコマンドを受信する準備ができている WRMSG FULLがセットされている場合、モジュールは最後の「ライトメッセージ」をまだ構文解析していない。 そのため、CMDCNTには最後のコマンドが反映されていない可能性がある。 待ち時間なしで複数のコマンドを連続で送ろうとするアプリケーションは、CMDCNTを評価するときにこの点を考慮する必要がある。
3	M	セットされているとき、メッセージフィールドにメッセージが格納されている。
4	LAST FRAG	セットされているとき、メッセージの最後のフラグメントがメッセージフィールドに格納されている。
5	NEW PD	セットされているとき、最後にSPI転送を行ってからネットワークにより更新されたデータがRDPDFIELDに格納されている。 なお、データの更新は必ずしもデータの変更を意味しない。 セットされていないとき、最後にSPI転送を行ったときと同じデータがRDPDFIELDに格納されている。 なお、SPI転送にてCRCエラーが発生した場合でも、このビットがクリアされる。
6	予約	-
7	予約	-

6.3 割り込み

割り込みマスクは、すべてのMOSIフレームで連続的に送信されます。 有効なMOSIフレームがAnybus CompactComによって受信されると、すべての割り込みが自動的に消去されます。

6.4 メッセージの分割

SPIプロトコルは、メッセージの分割をサポートします。

分割を無効にするには、ホストが送信するメッセージの最大サイズが収まる十分大きな値をMSGLENフィールドに設定します。 MビットおよびLAST FRAGビットは、各メッセージにおいてセットする必要があります。

分割を有効にするには、メッセージの最大サイズより小さな値をMSGLENフィールドに設定します。 Mビットは、メッセージまたはメッセージのフラグメントを含むすべてのSPIフレームに対してセットする必要があります。 LAST FRAGビットは、現在のフラグメントがメッセージの最後のフラグメントであることを表します。

6.5 SPIのエラー処理

エラーは、32ビットのCRCを用いて検出します。 MISOフレームとMOSIフレームとでは、CRCの位置がシフトしています。 Anybus CompactCom 40は、MOSIフレームのエラーを検出すると、ホストに対して無効なCRCを送信します。

ホストは、MISOフレームのCRCエラーを検出すると、その内容を無視して元のフレームを再送します。再送されるフレームでは、TOGGLEビット、Mビット、LAST FRAGビット、MSGLEN、およびMSGFIELDを、元のフレームと同じ値に維持する必要があります。 その他の各フィールドは新しい値を持つことができます。

以下の図に、通常のシナリオを示します。 ホストは、SPIフレームをサイクリックに送信します。このとき、送信を行うたびにSPIコントロールバイトのトグルビットを反転させます。

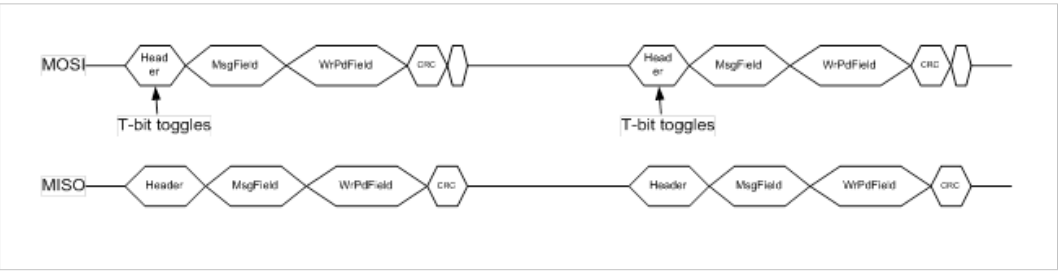

Fig. 11

MISOラインにて受信エラーが発生した場合、ホストは、これをMISO CRCを用いて検出し、再度送信を行います。 これが再送であることは、MOSIヘッダーのSPIコントロールバイトにあるトグルビットが反転していないことで表されます。

このシナリオを以下の図に示します。

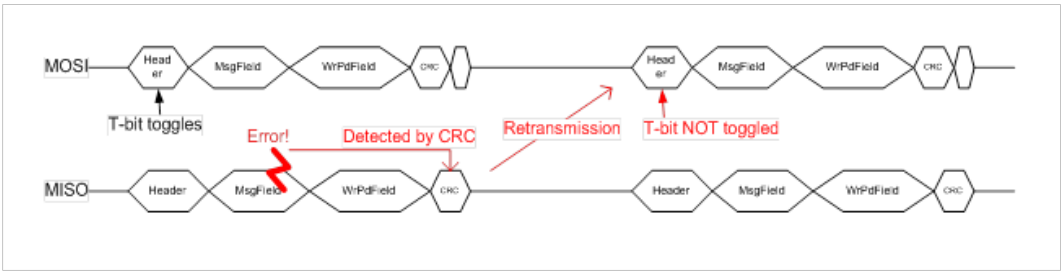

Fig. 12

MOSIラインにて受信エラーが発生した場合、Anybus CompactComは、これをMOSI CRCを用いて検出します。 Anybusは、MISO CRCが壊れた応答を返します。これにより、ホストからSPIフレームが再送されます。

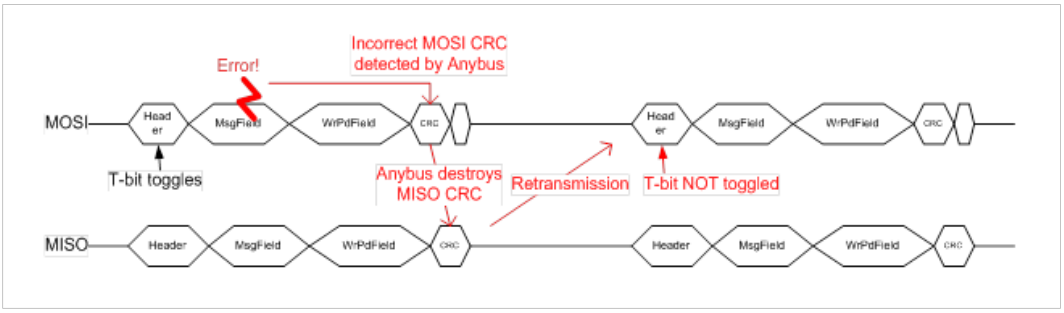

Fig. 13

6.6 アプリケーションのイベント駆動型ウォッチドッグ

アプリケーションにて要求される場合、Anybus CompactCom 40内でアプリケーションのウォッチドッグタイムアウトを有効にできます。 このタイムアウトを有効にすると、本モジュールは、ライトプロセスデータバッファの更新間隔がアプリケーションで選択されたウォッチドッグタイムアウト時間を超えたときに、アプリケーションが正しく動作していないとみなします。

アプリケーションのウォッチドッグタイムアウトは、Anybusオブジェクトのインスタンスアトリビュート#4（アプリケーションのウォッチドッグタイムアウト）で指定します。 *Anybusオブジェクト (01h), ページ 62*を参照してください。

7 シフトレジスタホスト通信

7.1 概要

Anybus CompactCom 40は、ホストプロセッサーを使用しないスタンドアロンでも使用できます。 プロセスデータは、ホスト上のシフトレジスタとの間でやり取りされます。 Anybus CompactCom 40はそれぞれの方向で最大32個のレジスタ（合計256ビットのデータ）をサポートします。

Anybus CompactCom 40のPROFIBUSバージョンは各方向とも24個までのレジスタをサポートし、合計で192ビットのデータをサポートします。

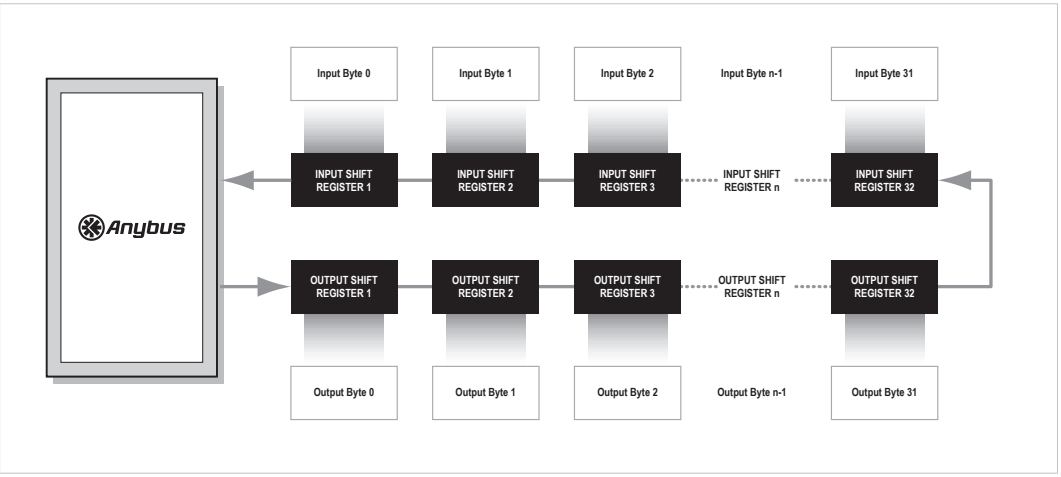

Fig. 14

Anybus CompactCom 40は、接続されている入出力シフトレジスタの数を自動的に検出します。 各シフトレジスタは、1個のUINT8型のADIで表されます。 入力ADIには"Input 0"、"Input 1"などの名前がつけられています。 出力ADIには"Output 0"、"Output 1"などの名前がつけられています。

Anybus CompactCom 40は、常にホストアプリケーションからネットワーク固有のアトリビュートを取得しようとします。 スタンドアロンモードではこれが不可能であるため、不揮発性メモリに格納された仮想アトリビュートリストが代わりに使用されます。*Anybusオブジェクト (01h), ページ 62*、セクション「仮想アトリビュート」を参照してください。 一部のアトリビュートは、コンフォーマンステストに合格するために実装が必須となっています。 コンフォーマンステスト情報, ページ *145*を参照してください。

7.2 リセット

スタンドアロンモードでは、ネットワークからのリセット要求を処理できるアプリケーションはありません。 リセットはAnybus CompactCom 40によって処理され、モジュールはネットワークからリセット要求が来ると自動的にリセットします。

8 シリアルホスト通信（UART）

8.1 概要

このモードは、Anybus CompactCom 30との後方互換性のために用意されているものです。Anybus CompactCom 40の通常の用途では使用しないでください。

シリアルホストインターフェースでは、テレグラムは共通の非同期シリアルインターフェースを介して送信されます。 ボーレートは、モジュールのホストインターフェースコネクタ上の信号で決定されます。詳細は*Anybus CompactCom 40 Hardware Design Guide*を参照してください。

シリアルホスト通信モードの詳細については、*Anybus CompactCom 30 Software Design Guide*を参照してください。

9 Anybusステートマシン

9.1 概要

Anybus CompactCom 40の基本を成しているのは、Anybusステートマシンです。 このステートマシンは、モジュールやネットワークの状態を常に反映しています。ステータスレジスタ（リードオンリー）, ページ 33を参照してください。

このステートマシンは、ムーアマシンとしてみなされます。すなわち、ホストアプリケーションは、すべての状態遷移を追跡する必要はありませんが、各状態において一定のタスクを実行することが期待されています。

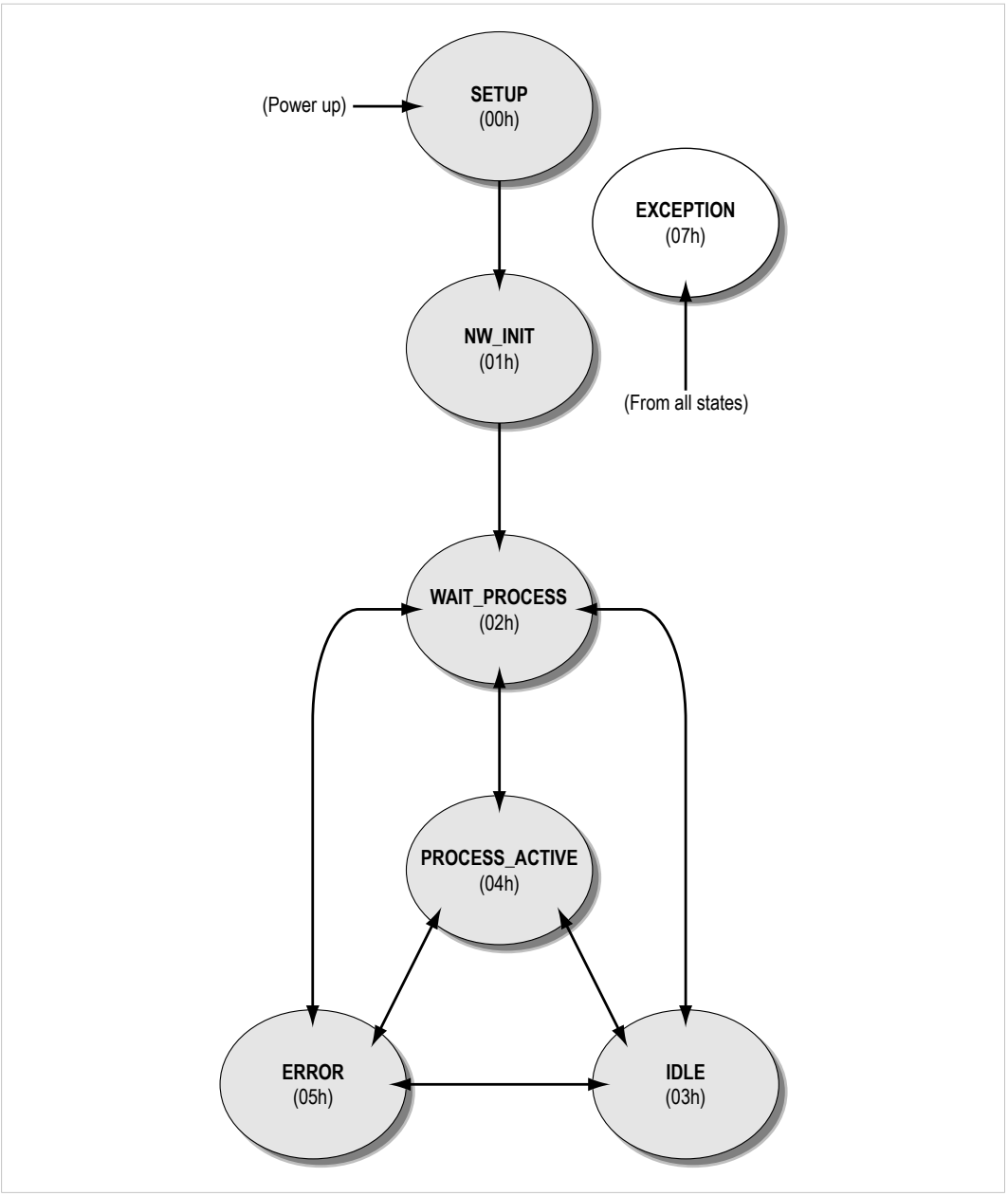

Fig. 15

9.2 状態に応じた動作

各状態において期待される動作を以下に示します。

状態	説明	期待される動作
SETUP	Anybus CompactComのセットアップが進行中。 この状態にあるとき、モジュールはアプリケーションにコマンドを送信しない。	*Anybus*の設定（*SETUP*状態）, ページ *59*を参照してください。
NW_INIT	Anybus CompactComは、ネットワーク関連の初期化タスクを実行中。 テレグラムにはプロセスデータが含まれる（そのようなデータがマッピングされている場合）が、プロセスデータチャネルはまだアクティブではない。	ネットワークの初期化（*NW_INIT*状態）, ページ *60*を参照してください。
WAIT_PROCESS	ネットワークのプロセスデータチャネルが一時的に非アクティブになっている。	ホストアプリケーションは、リードプロセスデータが無効であるとみなすこと。
IDLE	ネットワークインターフェースがアイドル状態。この状態の正確な意味はネットワークによって異なります。リードプロセスデータが、更新されるか、静的であるか（変化しない）は、ネットワークの種類によって異なります。	ホストアプリケーションは、リードプロセスデータに従って動作するか、アイドル状態に遷移することが可能です。
PROCESS_ACTIVE	ネットワークのプロセスデータチャネルがアクティブかつエラーがない状態。	通常のデータ処理を行う。
ERROR	1つ以上の重大なネットワークエラーが存在する。	リードプロセスデータが無効であるとみなすこと。 オプションとして、ホストアプリケーションはネットワーク固有の動作を行うことが可能です。 この場合でも、ライトプロセスデータはマスターに転送されるため、アプリケーションはこのデータを最新の状態に保つ必要があります。
EXCEPTION	ホストアプリケーションに関するエラーが発生しているため、モジュールはすべてのネットワークへの参加を停止した。 この状態は復旧不可能。すなわち、ネットワークのデータを交換できるようにするには、モジュールを再起動する必要がある。	可能であればエラーを訂正すること（Anybusオブジェクトから読み取り可能なエラーの詳細については、*Anybus*オブジェクト *(01h)*, ページ *62*を参照）。 エラーを訂正したら、Anybusモジュールをリセットする。

> **!** ライトプロセスデータはAnybus CompactComによりバッファリングされ、状態遷移後にネットワークに送信されるため、ホストアプリケーションは、NW_INIT（初期データ）、WAIT_PROCESS、IDLE、ERROR、およびPROCESS_ACTIVEにあるとき、ライトプロセスデータを最新の状態に保つこと。

下記も参照してください。

- ネットワークコンフィグレーションオブジェクト（*04h*）, ページ *79*

10 オブジェクトメッセージング

10.1 概要

10.1.1 基本原理

オブジェクトメッセージングでは、コマンドと応答の2つのメッセージを使用します。 メッセージのレベルでは、ホストアプリケーションとAnybus CompactComモジュールとの間でマスター/スレーブの関係はありません。どちらもコマンドを発行できるとともに、応答を返すことが求められます。 コマンドと応答は、Anybusオブジェクトモデルのインスタンスと常に関連しています。 これは、オブジェクト自身（インスタンス#0で指定）でも構いませんし、オブジェクト内のインスタンスでも構いません。

コマンドはいつでも発行可能です（ただし、受信側で新規コマンドの受け入れ準備ができている場合）。一方、応答は、以前に受信したコマンドに対する応答としてのみ送信可能です。 予期せぬ応答や不正な形式の応答は、常に破棄されます。

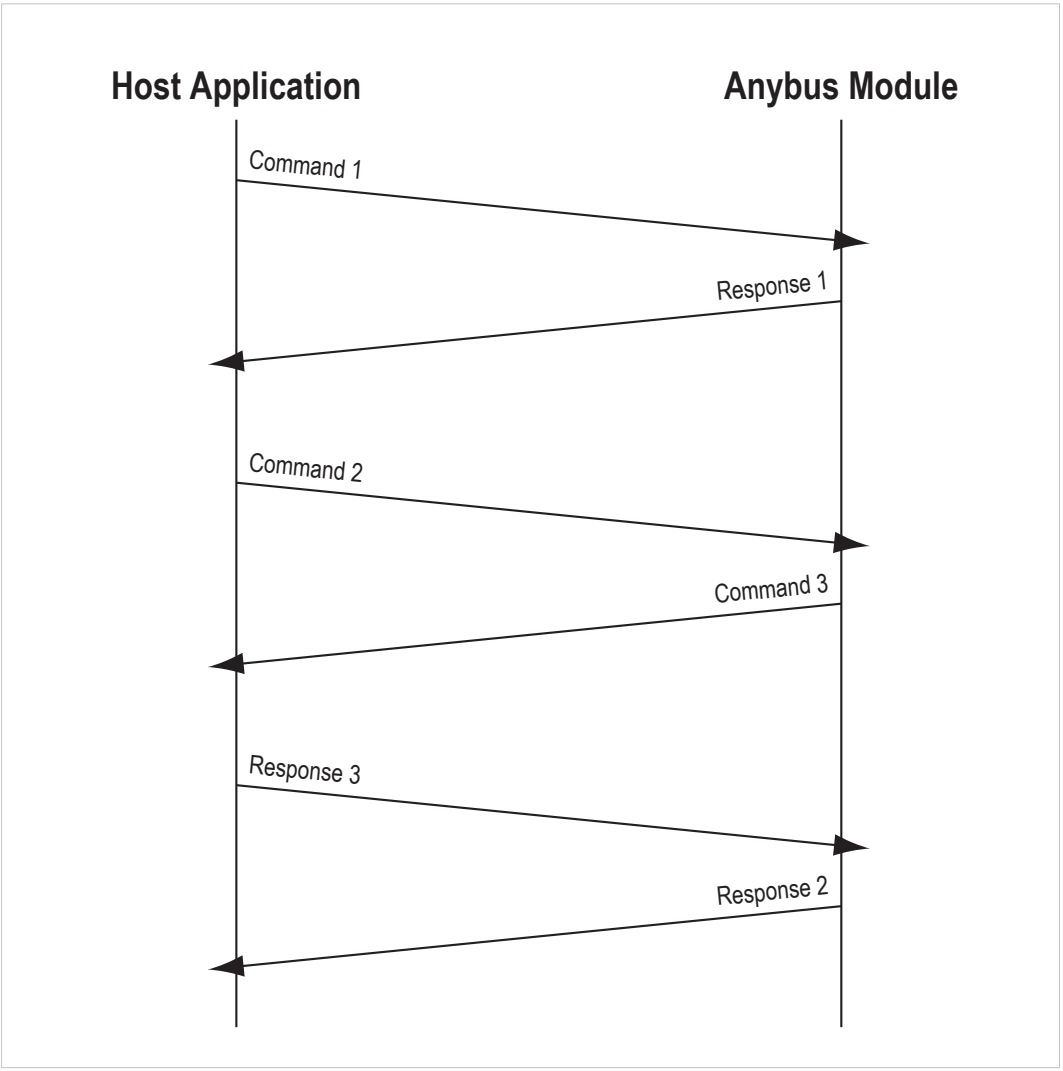

Fig. 16

コマンドと応答は非同期で処理されます。すなわち、前回のコマンドに対する応答が返される前に、新規コマンドが発行される場合があります。 また、コマンドが必ずしも到着順に処理されるとは限らず、応答が任意の順序で返されることをも意味します（図を参照）。 必要であれば、動作や結果が次のコマンドに影響を与えるコマンドに対し、ホストアプリケーション側でそのコマンドの応答を待つようにしてください。

10.1.2 ソースID

どの応答がどのコマンドに属するのかを追跡するため、各メッセージにはソースIDが付けられています。ホストアプリケーションはコマンド発行時に任意のソースIDを選択できますが、Anybusモジュールにより発行されたコマンドに応答する場合は、応答に含めるソースIDは元のコマンドからコピーしたものを使用する必要があります。

10.1.3 エラー処理

何らかの理由でコマンドが処理されなかった場合でも、受信側は応答を返す必要があります。 このような場合、適切なエラーコードをメッセージデータフィールドに設定するとともに、応答メッセージのヘッダーにエラーフラグをセットする必要があります。

そして、コマンドの発行元は、その応答を検査して、コマンドに対する成功応答なのかエラーメッセージなのかを確認する必要があります。

下記も参照してください。

- エラーコード, ページ 52

10.2 メッセージレイアウト

オブジェクトメッセージは、12バイトのヘッダーと、その後ろに続くメッセージに関連するデータで構成されています。

	内容								
オフセット	**b7**	**b6**	**b5**	**b4**	**b3**	**b2**	**b1**	**b0**	説明
0 ~ 1	メッセージデータサイズ								バイト単位で表した、MsgData[]フィールドのサイズ (最大1524バイト) 。
2 ~ 3	(予約)								-
4	ソースID								ソースID, ページ 46を参照。
5	オブジェクト								Anybusオブジェクトモデル内のソース/デスティネーションを指定する。
6	インスタンス (LSB)								
7	インスタンス (MSB)								
8	E								値と意味 0: メッセージはコマンドまたは正常応答 1: メッセージはエラー応答
		C							値と意味 0: メッセージは応答 1: メッセージはコマンド
			コマンドコード						コマンドコード, ページ 51を参照。
9	(予約)								-
10	CmdExt[0]								コマンド固有の拡張情報。 コマンド仕様, ページ 51を参照。 これらのフィールドは、エラー応答の場合、そのままにしておく必要があります。
11	CmdExt[1]								
12~n	MsgData[0-n]								メッセージデータフィールド

Anybus CompactCom 40 がAnybus CompactCom 30アプリケーションで使用されている場合は、8バイトのヘッダを使用する必要があります。 詳しくは、 *Anybus CompactCom 30 Software Design Guide*を参照してください。

10.3 メッセージの分割

Anybus CompactCom 40で可能な最大メッセージサイズは通常は1524 バイトです。 一部のアプリケーションでは最大メッセージサイズが255 バイトです。例えばアプリケーションには変更を加えず、Anybus CompactCom 40でAnybus CompactCom 30を置換する場合が挙げられます。 一部のオブジェクトサービスは、255バイトを超えるメッセージをサポートする必要があります。 これに対応するため、Anybus CompactCom 40は分割プロトコルをサポートしています。 シリアルテレグラムで使用される分割プロトコル（フラグメンテーションプロトコル）との混同を避けるため、このプロトコルはセグメンテーションプロトコルと呼ばれています。

10.3.1 コマンド分割手順

メッセージが分割された場合、コマンドの発行元は、同じコマンドヘッダーを複数回送信します。 各メッセージにおいて、データフィールドは次のデータセグメントにて送信されます。

コマンド詳細

	コマンドセグメントビット	説明
CmdExt[1]	ビット0: ビット1: ビット2: ビット3～7：	FS（先頭セグメント） LS（最終セグメント） AB（中断） 予約（0）

応答詳細

	応答セグメントビット	説明
CmdExt[1]	ビット0～7:	予約（0）

手順

分割されたコマンドを送信するには、以下のようにします。

- 最初のエレメントでは、FSビットをセットします。
- その後に続くエレメントでは、FSビットとLSビットをクリア（0）します。
- 最後のエレメントでは、LSビットをセットします。（単一フレームコマンド（<= 255または1524バイト、メッセージチャネルによって異なる）の場合は、FSビットとLSビットの両方をセットします。）

コマンドの受信側は、各セグメントに対して応答（ACK/NACK）を返し、セグメントを受け付けたかどうかを通知する必要があります。 NACKの場合、そのセグメントは破棄されます。 ただし、セグメンテーションは中断されないため、既に受け付けられたセグメントはセグメントバッファに残ります。

最後のセグメントに対する応答（ACK/NACK）には、実際の処理結果が格納されます。

コマンドの発行元は、ABビットがセットされたメッセージを発行することで、処理をいつでも中断できます。 これにより、セグメンテーションバッファがフラッシュされます。

コマンドが前回と同じかどうか判断するには、以下の項目をチェックします。

- 宛先オブジェクト
- インスタンス番号
- コマンド番号
- コマンドエクステンション0（CmdExt[0]）

10.3.2 応答分割手順

応答メッセージが分割された場合、コマンドの発行元は、同じコマンドを複数回送信して次のセグメントを要求します。 各メッセージにおいて、データフィールドは次のデータセグメントにて送信されます。

コマンド詳細

	コマンドセグメントビット	説明
CmdExt[1]	ビット0: ビット1: ビット2: ビット3～7：	予約（0） 予約（0） AB（中断） 予約（0）

応答詳細

	応答セグメントビット	説明
CmdExt[1]	ビット0: ビット1: ビット2～7:	FS（先頭セグメント） LS（最終セグメント） 予約（0）

手順

分割された応答を送信するには、以下のようにします。

- 最初のエレメントでは、FSビットをセットします。
- その後に続くエレメントでは、FSビットとLSビットをクリア（0）します。
- 最後のエレメントでは、LSビットをセットします。（単一フレームコマンド（<= 255または1524バイト、メッセージチャネルによって異なる）の場合は、FSビットとLSビットの両方をセットします。）

応答のLSビットがセットされていない場合、コマンドの発行元は、同じコマンドを再度送信して次のセグメントを要求します。

コマンドの発行元は、ABビットがセットされた要求/応答を発行することで、処理をいつでも中断できます。 これにより、セグメンテーションバッファがフラッシュされます。

コマンドが前回と同じかどうか判断するには、以下の項目をチェックします。

- 宛先オブジェクト
- インスタンス番号
- コマンド番号
- コマンドエクステンション0（CmdExt[0]）

10.4 データ形式

10.4.1 利用可能なデータ型

Anybus CompactCom 40は、標準で以下のデータ型を使用します。 ネットワーク固有のデータ型は、別途、ネットワークインターフェースのAppendixに記述されています（該当する場合）。

#	種類	ビット数	説明	範囲	すべてのネットワークで利用可能	プロセスデータで有効
0	BOOL	8	ブール型	0 = False, !0 = True	可	可
1	SINT8	8	符号つき8ビット整数	-128~ +127	可	可
2	SINT16	16	符号つき16ビット整数	-32768 ~ +32767	可	可
3	SINT32	32	符号つき32ビット整数	-2^{31}~ +(2^{31}-1)	可	可
4	UINT8	8	符号なし8ビット整数	0~ +255	可	可
5	UINT16	16	符号なし16ビット整数	0~ +65535	可	可
6	UINT32	32	符号なし32ビット整数	0~ +(2^{32}-1)	可	可
7	CHAR	8		0~ +255	可	不可
8	ENUM	8		0~ +255	可	可
9	BITS8	8	8ビットのビットフィールド	00000000~ 11111111	可	可
10	BITS16	16	16ビットのビットフィールド	0000000000000000~ 1111111111111111	可	可
11	BITS32	32	32ビットのビットフィールド	00000000 00000000 00000000 00000000~ 11111111 11111111 11111111 11111111	可	可
12	OCTET	8	不定の8ビットデータ	0~ +255	可	不可
13 ~ 15		(予約)				
16	SINT64	64	符号つき64ビット整数	-2^{63}~ +(2^{63}-1)	不可	可
17	UINT64	64	符号なし64ビット整数	0~ +(2^{64}-1)	不可	可
18	FLOAT	32	浮動小数点数（IEC 60559）	±1.17549435E-38~ ±3.40282347E+38	不可	可
32 ~ 48	PADx	0 ~ 16	パディングで使用するサイズ0 ~ 16のビットフィールド	N/A	可	可
65 ~ 71	BITx	1 ~ 7	サイズ1 ~ 7のビットフィールド	[0 ~ 1] ~ [0000000 ~ 1111111]	可	可

- CHAR型配列は、ネットワーク固有の文字列型に変換されます。
- コマンド「Set_Indexed_Attribute」および「Get_Indexed_Attribute」はCHAR型データには使用できません。
- ENUM型のデータは列挙値であり、ゼロから始まる連続する値の範囲に限られます。
- BITS8型、BITS16型、BITS32型、OCTET型、PADx型、BITx型のデータは、Anybus CompactCom 40でのみサポートされています。

10.4.2 ビットフィールド

ビットフィールド型は、ビットまたはビットの集まりが、それぞれ個々の意味を持つパラメータで使用します。 control/status wordやデジタルI/Oがその典型的な例です。

ビットフィールドのパラメータは、各ネットワークに固有の、デジタルI/Oやcontrol/status wordに適したデータ型に変換されます。

BITSx

BITSxデータ型（BITS8、BITS16、BITS32）は、偶数のデータバイトサイズを持ち、バイト境界に合わせる必要があります。 これらは、バイトオーダーに関して他のマルチバイトデータ型と同じように扱われます。

BITx

BITxデータ型（BIT1～BIT7）は、一組のビット列で構成され、バイト境界をまたいで配置できます。 BITS1～BITS7の型コード（65～71）は、指定子としての上位5ビット（常に01000）と、ビットカウンターとしての下位3ビットで構成され、ビットカウンターフィールドは1～7の値をとることが可能です。

10.4.3 CHAR型配列（文字列）の扱い

読み取り可能な文字列は、ADIを用いた2つの異なる方法で表すことが可能です。 1つはCHAR配列として表す方法、もう1つは文字列変数として表す方法です。 推奨される方法は、アプリケーションデータオブジェクト（FEh）のアトリビュートである「サブエレメント数」を使用して、ADI内の読み取り可能な文字列を変数（複数のサブエレメントを持つ1つのエレメントで構成される文字列変数）として表す方法です。 このセクションの内容は、主に、ADIでCHAR型配列を使用する場合に当てはまります。 以下、これらの文字列型を文字列と呼びます。

CHAR型配列は、ネットワーク固有の文字列型に変換されます（可能な場合）。 文字列の最大長、および文字列を格納するためのバッファ空間は、データ型と要素数で決定されます。

CHAR型配列の要素はすべて意味があります。Anybusモジュールは、文字列を読み取るときに終端文字列を期待しません。また、文字列を書き込むときに終端文字列を生成しません。 文字列の実際の長さは、Get_AttributeコマンドおよびSet_Attributeコマンドで与えられるペイロードサイズで決まります。

Set_AttributeおよびGet_Attributeを用いてADI構造の文字列にアクセスする際は、最大長になるようにNULバイトでパディングする必要があります。 データフィールドのサイズが0のGetまたはSet文字列は有効で空の文字列を示します。

通常、要素数、データ型、およびメッセージのペイロードサイズは、できるだけ整合性を保つようにしてください。 Anybus CompactCom 40がペイロードの長さと実際のバッファ空間との整合性を必ずしもチェックするとは限りません。

下記も参照してください。

- アプリケーションデータオブジェクト（*FEh*）, ページ *105*

10.4.4 OCTET

OCTET型は、バイトサイズの不定データに対して使用されます。 ADIでは、OCTET型の要素はサブエレメントを持つことが可能です。

10.4.5 PADx

PADx型には、PAD0～PAD16の17種類の型があります。 PADx型の変数は、一組のビット列で構成され、バイト境界をまたぐことが可能です。 PADx変数の値は特に意味を持たず、各ネットワークの方式によって完全にスキップされます。

型コード（32～48）は、指定子としての上位3ビット（常に001）と、ビットカウンターとしての下位5ビットで構成され、ビットカウンターフィールドは0～16の値をとることが可能です。

10.5 コマンド仕様

10.5.1 概要

この章では、グローバルなコマンド、すなわち、アクセスするオブジェクトにかかわらず、同じコマンドコードを持つコマンドについて説明します。

オブジェクトの中には、オブジェクト固有のコマンドで処理される特殊な要件を持つものがあります。このような場合、グローバルなコマンドとは異なり、コマンドコードが同じであっても、状況によって（アクセスするオブジェクトによって）意味が異なる場合があります。 オブジェクト固有のコマンドは、オブジェクトごとに別途説明しています（該当するコマンドがある場合）。

下記も参照してください。

- *Anybus モジュールオブジェクト, ページ 61*
- *ホストアプリケーションオブジェクト, ページ 100*

一般的なコマンドの説明においては、コマンドの一般的な内容と構造が明確に定義されていますが、実際の効果は状況によって大きく異なる場合があることに注意してください。

例：

- アプリケーションがリセットを発行 →ネットワークコンフィグレーションオブジェクト=ネットワークの設定がリセットされる
- ネットワークからリセット →Anybusがリセットを発行 →アプリケーションオブジェクト= AnybusがEXCEPTIONに移行し、ハードウェアリセット待ちになる

> **!** 「予約」と記されているフィールドは、注意して扱ってください。 Anybus CompactComに送信されるメッセージの予約フィールドは、将来のAnybusリビジョンで使用するよう定義されている可能性がありますので、必ず0（ゼロ）に設定してください。 Anybus CompactComから受け取ったメッセージ中の予約フィールドは無視してください。

10.5.2 コマンドコード

以下のコマンドはグローバルコマンドです。すなわち、アクセスするオブジェクトにかかわらず同じコマンドコードが使用されます。 コマンドの詳細については、以下の章で説明します。

コマンドコード	コマンド名
00h	(予約)
01h	Get_Attribute
02h	Set_Attribute
03h	Create
04h	Delete
05h	Reset
06h	Get_Enum_String
07h	Get_Indexed_Attribute
08h	Set_Indexed_Attribute
09h ~ 0Fh	(予約)
10h ~ 30h	（オブジェクト固有のコマンド用に予約）
31h ~ 3Eh	(予約)
3Fh	（オブジェクト固有のコマンド用に予約）

10.5.3 エラーコード

何らかの理由によりコマンドが実行できなかった場合、応答におけるメッセージデータフィールド（MsgData[ ]）の最初のバイトが、コマンド発行元に問題の詳細を通知するのに使用されます。

また、メッセージデータセクションには、オブジェクト固有のエラー情報が追加される場合があります。

コード	エラー	意味
00h	(予約)	-
01h		
02h	Invalid message format	コマンドビットとエラービットがセットされている。
03h	Unsupported object	オブジェクトが登録されていない。
04h	Unsupported instance	ターゲットインスタンスが存在しない。
05h	Unsupported command	ターゲットオブジェクトは指定したコマンドをサポートしていない。
06h	Invalid CmdExt[0]	CmdExt[0]の値が無効、またはCmdExt[0]とCmdExt[1]の組み合わせが無効
07h	Invalid CmdExt[1]	CmdExt[1]の設定が無効 。
08h	Attribute not settable	要求されたアトリビュートを設定できない。
09h	Attribute not gettable	要求されたアトリビュートを取得できない。
0Ah	Too much data	メッセージデータフィールドのデータが多すぎる。
0Bh	Not enough data	メッセージデータフィールドに十分なデータがない。
0Ch	Out of range	指定された値が範囲外。このエラーコードは、11hまたは12hが使用できないときにのみ使用すること
0Dh	Invalid state	現在の状態ではこのコマンドはサポートされていない。
0Eh	Out of resources	リソースが不足しているため、ターゲットオブジェクトがコマンドを実行できない。
0Fh	Segmentation failure	不正なセグメンテーションプロトコル処理。
10h	Segmentation buffer overflow	受信データが多すぎる。
11h	Value too high	書き込んだ値が大きすぎる
12h	Value too low	書き込んだ値が小さすぎる。
13h	Attribute controlled from another channel	「リードプロセスデータ」にマップされたアトリビュートへの書き込みに対するNAKに使用。
14h	Message channel to small	ホストアプリケーションが使用中のメッセージリード / ライト領域は、応答データに見合う十分な大きさのデータフィールドを持つメッセージチャネルをサポートしていない。
15h	General error	他の既存のエラーコードと一致しないエラーが発生した。
16h	Protected access	現在、アクセスがホストアプリケーションによってプロテクトされているため、読み取りまたは書き込み操作を実行できなかった。
17h ~ FEh	(予約)	-
FFh	Object specific error	オブジェクトによりオブジェクト固有のエラーコードが返された。 メッセージデータフィールド（MsgData[0~n]）に追加の詳細情報が格納される場合がある。

10.5.4 Get_Attribute

詳細

コマンドコード: 01h

有効な対象： （状況により異なる）

説明

このコマンドは、アトリビュートの値を取得します。 エラー応答のアトリビュート番号は、そのままにしておく必要があります。

- コマンド詳細：

フィールド	内容
CMDExt[0]	アトリビュート番号
CMDExt[1]	(予約)

- 応答詳細：

フィールド	内容
MsgData[0～n]	アトリビュート値

10.5.5 Set_Attribute

詳細

コマンドコード: 02h

有効な対象： （状況により異なる）

説明

このコマンドは、アトリビュートに値を設定します。 エラー応答のアトリビュート番号は、そのままにしておく必要があります

- コマンド詳細：

フィールド	内容
CMDExt[0]	アトリビュート番号
CMDExt[1]	(予約)
MsgData[0～n]	アトリビュート値

- 応答詳細：

（データなし）

10.5.6 Create

詳細

コマンドコード: 03h

有効な対象： オブジェクトインスタンス (インスタンス #0)

説明

このコマンドは、オブジェクトに新規インスタンスを作成します。 成功した場合、新規に作成されたインスタンスの番号が応答のデータ部に格納されます。

- コマンド詳細：

  オブジェクト固有

  すべてのオブジェクトにこのコマンドに関する固有の詳細があるわけではありません。 オブジェクト固有の詳細がある場合は、当該オブジェクトの説明に記載されています。

- 応答詳細：

フィールド	内容
MsgData[0]	作成されたインスタンスの番号 (下位バイト)
MsgData[1]	作成されたインスタンスの番号 (上位バイト)

10.5.7 Delete

詳細

コマンドコード: 04h

有効な対象： オブジェクトインスタンス (インスタンス #0)

説明

このコマンドは、作成済みインスタンス (前記を参照) を削除します。 成功すると、指定したインスタンスにより占有されているすべてのリソースが解放されます。

- コマンド詳細：

フィールド	内容
CMDExt[0]	削除するインスタンスの番号 (下位バイト)
CMDExt[1]	削除するインスタンスの番号 (上位バイト)

- 応答詳細 (成功)：

  (データなし)

- 応答詳細 (エラー)：

フィールド	内容
Invalid CMDExt[0]	指定したインスタンスが存在しない。

10.5.8 Reset

詳細

コマンドコード: 05h

有効な対象： （状況により異なる）

説明

このコマンドは、オブジェクトに対してリセットコマンドを実行します。

- コマンド詳細：

フィールド	内容
CMDExt[0]	(予約)
CMDExt[1]	00h =パワーオンリセット（実際のパワーオンまたはシミュレートされたパワーオン） 01h =工場出荷時状態へのリセット 02h =パワーオンリセット + 工場出荷時状態へのリセット

- 応答詳細：

  （データなし）

10.5.9 Get_Enum_String

詳細

コマンドコード: 06h

有効な対象： （状況により異なる）

説明

このコマンドは、列挙型（ENUM）の領域のアトリビュートを取得します。 指定した列挙型の値に対応するリテラル文字列が返されます。

- コマンド詳細：

フィールド	内容
CMDExt[0]	アトリビュートの番号
CMDExt[1]	列挙型の値

- 応答詳細（成功）：

フィールド	内容
MsgData[0 ~ n]	列挙型の文字列。

- 応答詳細（エラー）：

フィールド	内容
Invalid CMDExt[0 ~ n]	列挙型の値が範囲外。

10.5.10 Get_Indexed_Attribute

詳細

コマンドコード: 07h

有効な対象： （状況により異なる）

説明

このコマンドは、複数の要素を含むアトリビュート（配列）に含まれる1つの要素の値を取得します。 なお、このコマンドを使用してCHAR型のアトリビュートにアクセスすることはできません。

- コマンド詳細：

フィールド	内容
CMDExt[0]	アトリビュートの番号
CMDExt[1]	インデックス（最初の要素のインデックスは0）

- 応答詳細（成功）：

フィールド	内容
MsgData[0 ~ n]	値

- 応答詳細（エラー）：

フィールド	内容
Invalid CMDExt[0 ~ n]	インデックスが範囲外

10.5.11 Set_Indexed_Attribute

詳細

コマンドコード: 08h

有効な対象： （状況により異なる）

説明

このコマンドは、複数の要素を含むアトリビュート（配列）に含まれる1つの要素に値を設定します。 なお、このコマンドを使用してCHAR型のアトリビュートにアクセスすることはできません。

- コマンド詳細：

フィールド	内容
CMDExt[0]	アトリビュートの番号
CMDExt[1]	インデックス（最初の要素のインデックスは0）
MsgData[0 ~ n]	設定する値

- 応答詳細（成功）：

  （データなし）

- 応答詳細（エラー）：

フィールド	内容
Invalid CMDExt[1]	インデックスが範囲外

11 初期化と起動

11.1 概要

ネットワークに参加する前に、以下の手順を行う必要があります。

1. 起動手順

   起動手順の目的は、両方の側（ホストアプリケーションとAnybus CompactComモジュール）が通信できるよう準備することです。 通常、Anybus CompactComモジュールは1.5秒以内に通信可能になります。その後、モジュールはSETUP状態になります。 詳細については、起動手順, ページ 57を参照してください。

2. Anybus CompactComの設定

   この手順では、モジュールがどのように動作するかを決定します。 完了するとモジュールはNW_INIT状態になります。

   詳細については、*Anybusの設定（SETUP状態）, ページ 59*を参照してください。

3. ネットワークの初期化

   このステージでは、モジュールは、ホストアプリケーションからの情報の読み取りと評価を試みます。 完了するとモジュールはWAIT_PROCESS状態になります。

   詳細については、ネットワークの初期化（*NW_INIT状態*）, ページ *60*を参照してください。

> ! ファームウェアのダウンロード後にモジュールが再起動されるとき、アプリケーションはアップグレードの完了を待ってから他の処理を開始する必要があります。以下を参照してください。

11.2 起動手順

起動手順はどの種類のホストインターフェースを使用するかによって若干異なりますが、通常は1.5秒以内に完了します。

1. 電源を入れます。
2. モジュールへのリセットを解放します。
3. Anybus CompactCom 40が応答するのを待ちます。 期待される応答はインターフェースによって異なります。

インターフェース	期待される応答
パラレルホスト（8/16 ビット）	ホストアプリケーションは、Anybus CompactComの割り込み信号がアクティブになるまで待ってから通信を開始します。
SPI	ホストアプリケーションは、Anybus CompactComモジュールへのリセット信号を解放した後、Anybus CompactComの割り込み信号がアクティブになるまで待ってから（こうすることでモジュールの準備が完了したことが示されます）SPI通信を開始できます。 ホストアプリケーションのもう1つの選択肢は、Anybus CompactComのリセット信号を解放した後すぐにSPI通信を開始することです。 この場合は、最初のうち、ホストアプリケーションで受信したSPIテレグラムでCRCエラーが検出されることがあります。 これらのテレグラムは、SPIプロトコルの通常のエラー処理ルールに従って再送する必要があります。*SPIのエラー処理, ページ 39*を参照してください。
シフトレジスタ	Anybus CompactCom 40モジュールは、リセットが解放された後、自律的に初期化されます。
シリアルホスト	このインターフェースは、Anybus CompactCom 30シリーズのシリアルホストインターフェースと後方互換性があります。 詳細についてはAnybus CompactCom 30 Software Design Guideを参照してください。

11.2.1 ネットワークからアップグレードする際に推奨される起動手順

ネットワークからのファームウェアアップグレードを許可するには、アプリケーションオブジェクト（FFh）、インスタンス#1のアトリビュート#5を実装します。モジュールは、新しいファームウェア候補が候補領域で利用可能になると、ホストアプリケーションに通知します。 ホストアプリケーションは、この情報を不揮発性メモリに格納する必要があります。 アプリケーションオブジェクト（*FFh*）, ページ *114*を参照してください。

ネットワークからのファームウェアアップグレードが許可されている場合は、以下の起動手順が推奨されます。

> **!** 起動手順中はAnybus CompactComモジュールをリセットしないでください。

1. 電源を入れます。
2. モジュールへのリセットを解放します。
3. Anybus CompactCom 40が応答するのを待ちます。 期待される応答はインターフェースによって異なります。

インターフェース	期待される応答
パラレルホスト（8/16 ビット）	割り込み
SPI	CRCエラーのない最初のSPIテレグラム、またはアクティブな割り込み信号
シフトレジスタ	N/A
シリアルホスト	CRCエラーのない最初のシリアルテレグラム

4. 新しいファームウェア候補が利用可能な場合、モジュールはファームウェアのロードを開始します。これには、最大1分を要します。利用できる候補ファームウェアがない場合は、ブート時間は常に1.5 秒以内です。 ファームウェアアップデートの場合は、モジュールをリセットしないでください。可能であれば、エンドユーザーに対して次のようなメッセージを表示してください。「Waiting for Anybus module」....
5. 応答が検出されたら： Anybus CompactCom 40モジュールの初期化を開始します。 それまでに表示されたメッセージをすべて削除します。

モジュールが上記のように反応しない場合は、正しく起動されていません。 www.anybus.com/supportにてHMS Industrial Networks ABまでお問い合わせください。

> **!** ファームウェアのダウンロード後にAnybus CompactCom 40がリセットされる際、アプリケーションはアップグレードが完了するまで初期化の開始を待つ必要があります。 Anybus CompactComはダウンロード中および/またはインストール中に発生する問題に対して保護されているため、再起動すると元の状態に戻ります。
>
> ダウンロード後に新しいファームウェアをインストールするには、Anybus CompactCom 40をリセットします。新しいファームウェアのインストールが電源喪失などで中断された場合は、Anybus CompactCom 40を再起動してください。インストールプロセスが最初から自動的に開始され、特に操作を行わなくても新しいファームウェアがインストールされます。

11.3 Anybusの設定（SETUP状態）

このステージは、大きく4つの手順から成ります。

1. Anybusモジュールに関する情報を収集（任意）

   ホストアプリケーションは、ネットワークの種類およびモジュールの設定時に必要となる可能性のあるその他のプロパティを、Anybusオブジェクト（01h）から取得します。 この情報は、モジュールタイプの値などに基づき各種実装を選択するのに使用される場合もあります。

2. ネットワークコンフィグレーション（任意）

   このステージでは、ホストアプリケーションは、物理スイッチの設定値（すなわち、ノードアドレス、ボーレートなど）などを用いて、ネットワークコンフィグレーションオブジェクトの全インスタンスを更新する必要があります。 キーパッドやディスプレイなどの"ソフト"入力デバイスに由来する設定は、この時点では更新する必要はありません。

3. プロセスデータマッピング（任意）

   ホストアプリケーションは、オプションとして、ADIをプロセスデータにマッピングできます。

   このステップは任意ですが、ネットワークシステムやAnybusの実装によっては必須な場合があります。

   一部のAnybus実装では、動作中にプロセスデータマップの変更を行う場合があります。 詳細については、アプリケーションデータオブジェクト（*FEh*）, ページ *105*を参照してください。

4. 設定の完了

   Anybusオブジェクト（01h）のSetup Completeアトリビュートを TRUEに設定することで一連の設定の完了を行います。

   成功すると、モジュールはNW_INIT状態（下記）に移行します。失敗した場合はEXCEPTION状態に移行します。 後者の場合、Anybusオブジェクト（01h）のExceptionアトリビュートから詳細情報を読むことが可能です。

下記も参照してください。

- ネットワークのデータ交換, ページ *14*
- *Anybus*ステートマシン, ページ *43*
- *Anybus*オブジェクト *(01h)*, ページ *62*
- ネットワークコンフィグレーションオブジェクト（*04h*）, ページ *79*

11.4 ネットワークの初期化（NW_INIT状態）

このステージでは、Anybusモジュールは、ホストアプリケーションからの情報の読み取りと評価を試みます。 ホストアプリケーションは、ホストアプリケーションオブジェクトに送られてくる要求に対して応答を返す必要があります。 要求されたオブジェクトまたはアトリビュートがホストアプリケーションに実装されていない場合は、単にエラーメッセージを返します。 このときモジュールは、要求されたアトリビュートに対するモジュール固有のデフォルト値を使うか、設定済みの仮想アトリビュートを使用します。

ホストアプリケーションは、物理スイッチを使用しないものも含め、ネットワークコンフィグレーションオブジェクトのすべてのインスタンスを自由に更新できます。

この状態において重大なエラー（モジュールの処理の進行を妨げるエラーなど）が発生した場合、モジュールはEXCEPTION状態に移行します。 詳細情報は、Anybusオブジェクト（01h）のExceptionアトリビュートから読むことが可能です。

完了すると、モジュールはWAIT_PROCESS状態になります。

> **!** この状態への遷移は重要です。特に、シリアルホストインターフェースを使用する場合、送信されるテレグラムにはプロセスデータが含まれることがあるため（設定により異なる）、この状態への遷移は非常に重要です。 ライトプロセスデータはモジュールによりバッファリングされ、次の状態遷移でネットワークに送信される可能性があるため、この状態においてライトプロセスデータを最新の状態に保つことが重要です。

下記も参照してください。

- *Anybusステートマシン, ページ 43*
- *ネットワークコンフィグレーションオブジェクト（04h）, ページ 79*

12 Anybus モジュールオブジェクト

12.1 概要

本章で説明するオブジェクトは、すべてのAnybus CompactComに標準で実装されています。 本章では、オブジェクトをいつ、どのように使用するかという観点から、オブジェクトの機能を分類しています。

下記も参照してください。

- *メッセージの分割, ページ 47*
- *エラーコード, ページ 52*
- *機能の分類, ページ 137*

各オブジェクトの詳細情報については、下記を参照してください。

- *Anybusオブジェクト (01h), ページ 62*
- *診断オブジェクト (02h), ページ 68*
- *ネットワーク オブジェクト (03h), ページ 73*
- *ネットワークコンフィグレーションオブジェクト (04h) , ページ 79*
- *Anybus ファイルシステムインターフェース・オブジェクト (0Ah), ページ 81*
- *ファンクショナルセーフティモジュール・オブジェクト (11h), ページ 95*

12.2 オブジェクトリビジョン

オブジェクトリビジョンアトリビュートの目的は、Anybusモジュールにおけるオブジェクトのリビジョンがホストアプリケーションに実装されているソフトウェアと互換性があるかを判断し、オブジェクトのリビジョンに応じて実装を選択できるようにすることです。

原則として、オブジェクトのリビジョンは、ホストアプリケーションソフトウェアの実装において互換性の問題が発生する可能性のある変更がオブジェクトに行われたときに更新されます。 アトリビュートやコマンドの追加などの小規模な変更の場合は、リビジョン変更は通常発生しません。

12.3 Anybusオブジェクト (01h)

カテゴリ

基本

オブジェクトの説明

このオブジェクトには、Anybus CompactComモジュール自身に関するデータが集められています。 対象となるデータは、モジュールが対応している産業用ネットワークの情報ではなく、モジュール固有の情報を表しています。 このデータはアプリケーションで使用できます。 その値は産業用ネットワークによって異なる場合があります。その場合、各Appendixにてそれらのデータが記述されています。

このオブジェクトにおけるアトリビュートのほとんどは、Get_Attributeコマンドを用いてデータの取得が可能な「get」アクセスタイプを持っています。 唯一設定が必須となっているアトリビュートは、設定が完了したことをアプリケーションからモジュールに通知するのに使用される、「Setup complete」（インスタンス#1、アトリビュート#5）です。コンフィグレーションが受け付けられなかった場合、モジュールはEXCEPTION状態に移行します。このとき、インスタンス#1、アトリビュート#6（「Exception Code」）から情報を読み取ることが可能です。

サポートコマンド

オブジェクト：	Get_Attribute（01h）
インスタンス：	Get_Attribute（01h）
	Set_Attribute (02h)
	Get_Enum_String (06h)

オブジェクトアトリビュート（インスタンス**#0**）

#	名前	アクセス	データ型	値
1	Name	Get	CHAR配列	Anybus
2	Revision	Get	UINT8	04h
3	Number of instances	Get	UINT16	0001h
4	Highest instance no.	Get	UINT16	0001h

インスタンスアトリビュート（インスタンス**#1**）

#	名前	アクセス	データ型	説明	
1	Module type	Get	UINT16	値	意味
				0401h:	Standard Anybus CompactCom 30
				0402h:	Anybus CompactCom Drive Profile 30
				0403h:	Standard Anybus CompactCom 40
				0404h:	Anybus IP
				（その他）	（将来の製品のために予約）
2	Firmware version	Get	構造体： UINT8 メジャー UINT8 マイナー UINT8 ビルド	ファームウェアのバージョン。 なお、通常は、この値を個々の機能が利用可能かどうかの判断に使用しないでください。 その判断を行う場合は、各オブジェクトのRevisionアトリビュートを使用してください	
3	Serial number	Get	UINT32	一意のシリアル番号	

#	名前	アクセス	データ型	説明
4	Application watchdog timeout	Get/Set	UINT16	アプリケーションウォッチドッグのコンフィグレーション 値 意味 0: 無効 (デフォルト) （その他）: タイムアウト値（ms） 有効にすると、アプリケーションの状態にかかわらず、ウォッチドッグのタイムアウト時間が直ちにアクティブになります。 タイムアウト時間が再起動されるたびに内部タイマーがリロードされます。そのため、稼働中にこのアトリビュートの値の変更が行えます。 タイムアウト値を超えると、Anybus CompactCom 40はEXCEPTION状態になって例外コード01hを示します。
5	Setup complete	Get/Set	BOOL	Anybusの設定が完了したら、このアトリビュートをTRUEに設定します。 コンフィグレーションが受け付けられると、AnybusモジュールはNW_INIT状態に移行します。 そうでない場合、すなわち、コンフィグレーションにて重大なエラーが検出された場合は、モジュールはEXCEPTION状態に移行します。この場合、Exception Codeアトリビュート（下記）から詳細情報を読み取ることが可能です。 下記も参照してください。 *Anybusの設定（SETUP状態）, ページ 59*
6	Exception code	Get	ENUM	以下に示すException Code（例外コード）を参照してください。
7	FATAL event	Get/Set	構造体：（HMS独自）	最後に発生したFATAL event（存在する場合）がこのインスタンスに記録されます。 HMSサポートによる確認時に使用されます。 （このアトリビュートの内容は、アプリケーション開発中におけるHMSサポートへの情報としてのみ使用されます）
8	Error Counters	Get	構造体： UINT16 DC UINT16 DR UINT16 SE UINT16 FE	エラーカウンター（FFFFhで停止）。 （このアトリビュートの内容は、アプリケーション開発中にのみ使用されます。） DC: 破棄されたコマンド（R = 0の状態で受信） DR: 破棄された（予期せぬ）応答 SE: シリアル受信エラー FE: フラグメンテーションエラー
9	Language	Get/Set	ENUM	現在の言語 値 文字列 00h: “English”（デフォルト） 01h: “Deutsch” 02h: “Español” 03h: “Italiano” 04h: “Français” 下記も参照してください。 *アプリケーションオブジェクト（FFh）, ページ 114*（コマンドChange_Language_requestの詳細を含む）。
10	Provider ID	Get	UINT16	製造時にHMSにてFLASHに書き込まれ、変更できません（詳細はHMSに問い合わせてください）。 値 意味 0001h: HMSネットワーク FFFFh: (予約) その他： プロバイダー固有
11	Provider specific info	Get/Set	UINT16	このアトリビュートに格納される情報は、プロバイダー固有です。すなわち、定義済みの意味を持たず、Anybusが評価することも使用することもありません。 このアトリビュートに書き込まれた値は、不揮発性メモリに格納されます。デフォルト値は0000hです。

#	名前	アクセス	データ型	説明
12	LED colors	Get	構造体： UINT8 LED1A UINT8 LED1B UINT8 LED2A UINT8 LED2B	このアトリビュートは、ネットワークステータスLEDの色を決定します。 詳細については、Anybus CompactCom M40 Hardware Design Guideを参照してください。 値 意味 00h: なし（未使用） 01h: 緑色 02h: 赤色 03h: 黄色 04h: オレンジ色 05h: 青色 06h: 白色
13	LED status	Get	UINT8	下記のように、ネットワークステータスLEDの現在の状態を保持するビットフィールド。 ビット： 内容 b0: LED1Aの状態（0 = OFF、1 = ON） b1: LED1Bの状態（0 = OFF、1 = ON） b2: LED2Aの状態（0 = OFF、1 = ON） b3: LED2Bの状態（0 = OFF、1 = ON） b4: LED3Aの状態（0 = OFF、1 = ON） b5: LED3Bの状態（0 = OFF、1 = ON） b6: LED4Aの状態（0 = OFF、1 = ON） b7: LED4Bの状態（0 = OFF、1 = ON）
14	Switch status	Get	構造体： UINT8 SW1 UINT8 SW2	ノードアドレスとボーレートを表すDIPスイッチの値。 シリアル、SPI、シフトレジスタモードでサポートされています。その他のモードでは、アトリビュートにランダムなデータが含まれます。 このアトリビュートはAnybus CompactCom 40でのみサポートされています。
15	（Anybus CompactCom 40では未使用）	-	-	-
16	GPIO configuration	Get/Set	UINT16	ホストインターフェースのGPIOピンの構成。 Setアクセスは SETUP状態においてのみ有効。 コード 説明 0000h、 GIP[0~1]は一般的な入力ピンとして使用されます。 $\overline{\text{GOP}}$[0~1]は一般的な出力ピンとして使用されます。 Anybus CompactCom 40の場合、このモードは拡張LED機能（0001h）モードと同一です。 0001h: 拡張LED機能（デフォルト）： GIP[0~1]がネットワーク固有仕様として使用され、アクティブローLED出力が左側、または単一ポートに関連付けられています。 $\overline{\text{GOP}}$[0~1]がネットワーク固有として使用され、アクティブローLED出力が右側ポートに関連付けられています。 0002h: RMII: GIP[0~1]、/GOP[0~1]、LED1A、LED1B、LED2A、LED2Bを使用して、アプリケーションに対してRMIIインターフェースとして動作します。 RMIIをサポートするモジュールについてのみ有効です。 16ビットパラレルモードで実行している場合は、このコードは有効ではありません。 0003h Three-state: GIP[0~1]と$\overline{\text{GOP}}$[0~1]は、スリーステートに設定されています。

#	名前	アクセス	データ型	説明
17	Virtual attributes	Get/Set	UINT8の配列	このアトリビュートは、モジュールをスタンドアロンで使用する時などに、仮想ホストアプリケーションアトリビュートをモジュールに実装するために使用します。 このアトリビュートはAnybus CompactCom 40でのみサポートされています。 SetアクセスはSETUP状態においてのみ有効。 最大サイズ：1524バイト 不揮発性メモリに格納される。 下記の「仮想アトリビュート」 を参照ください。
18	Black list/White list	Get/Set	 構造体： UINT8 InfoBits UINT8 ListLen UINT16 Prot#1 UINT16 Prot#2 ... UINT16 Prot#n	このアトリビュートは、ブラックリスト/ホワイトリストを実装するのに使用されます。 下記の「 ブラックリスト/ホワイトリスト」 を参照ください。 このアトリビュートはAnybus CompactCom 40でのみサポートされています。 フォーマット： InfoBits： ビット0: 0 =ブラックリスト 1 =ホワイトリスト ビット1 ~ 7: 予約 ListLen： リストの長さ。#nエントリ（ Prot#n ）と同じ 0 =リストは無効 Prot# ： ネットワークタイプ識別子
19	Network time	Get	UINT64	現在のネットワーク時間。 ネットワーク時間のフォーマットはネットワーク 固有になっています。 0 =そのネットワークはネットワーク時間をサポートしていない このアトリビュートはAnybus CompactCom 40でのみサポートされています。
20	Firmware custom version	Get	UINT8	このアトリビュートはファームウェアのバージョンプレフィックスを保持し、ファームウェアの特別な派生であることを示します。
21	Anybus IP License	Get	UINT8	Anybus IPによって検出されたライセンスチップに関する情報。 値については以下を参照してください。 Anybus IPプラットフォームでのみサポートされています。

仮想アトリビュート

仮想アトリビュートリストは1524バイトの配列で、不揮発性メモリに格納されます。 アトリビュートは以下のフォーマットを使用して作成されます。

オブジェクト（ 8ビット ）
インスタンス（ 16ビット ）
アトリビュート（ 8ビット ）
長さ（ 16ビット ）
データ（ 長さ * 8ビット ）

仮想アトリビュートは、Anybusオブジェクトのアトリビュート#17を使用してアクセスします。

Anybus CompactCom 40がホストアプリケーションからネットワーク固有のアトリビュートを取得しようとして、アプリケーションがこれらのアトリビュートを供給できない場合、エラーコードが返されます。 このとき、モジュールは、欠落しているアトリビュートを仮想アトリビュートリストでチェックします。 エラー応答に含まれるアトリビュート番号はそのままにしておいてください。番号が変わると、要求されたアトリビュートをリスト内で見つけられません。

仮想アトリビュートリストを使用すると、ネットワーク固有のオブジェクトやアトリビュートがホストアプリケーションに実装されていなくても、それらをモジュールに提供できます。 これは例えば、アプリケーションを新しいネットワークに適応させる場合や、オリジナルのアプリケーションで使用できないネットワーク固有のアトリビュートをサポートしなければならない場合に役立ちます。 一部のアトリビュートは、コンフォーマンステストに合格するために必須となっています。 コンフォーマンステスト情報, ページ *145*を参照してください。

仮想アトリビュートリスト内の配列データが1つのメッセージに収まらない場合、Get_Attribute要求はエラーコード「Messaging channel too small」（ 14h ）を返します。

Anybus CompactCom 40がスタンドアロンモードで使用されているとホストアプリケーションを利用できないため、Anybus CompactCom 40が仮想アトリビュートリスト内のアトリビュートをチェックします。 仮想アトリビュートは、「Setup complete」アトリビュートの設定前にのみ設定できます。

> **!** スタンドアロン・シフトレジスタを使用して実装する際にモジュールのIDを変更する場合は、仮想アトリビュートを実装する必要があります。

ブラックリスト/ホワイトリスト

ブラックリスト/ホワイトリストを使用すると、ホストは、特定のネットワークタイプをブロックまたは許可できるようになります。

このリストのヘッダーにあるビット0は、それがブラックリストなのかホワイトリストなのかを決定します。 ホワイトリストとして構成すると、このリストにあるネットワークタイプのみが許可されます。 ブラックリストとして構成すると、このリストにあるすべてのネットワークタイプが拒否されます。

ホワイトリストを使用すると、事前定義に従って選択されたネットワークタイプのみを許可できます。

ブラックリストを使用すると、定義済みのネットワークタイプをブロックできます。

ブラックリスト/ホワイトリストは、Anybusオブジェクトのアトリビュート#18を使用してアクセスします。

Network type	ネットワーク
0005h	PROFIBUS DP-V1
0025h	DeviceNet
0087h	EtherCAT
0089h	PROFINET IRT
0090h	CC-Link
0093h	Modbus TCP
009Bh	EtherNet/IP
009Dh	PROFINET IRT Fiber Optic
009Eh	CC-Link IE Field
009Fh	Ethernet POWERLINK
00A3h	Common Ethernet

Anybus IPライセンス

コード	ライセンス	説明
0x00	None	セキュリティチップはまだアクセスされていません。
0x01	Time bomb	セキュリティチップが取り付けられていないか、セキュリティチップのアクセス・読み取り時に何かが正常に機能しませんでした。 Anybus IPは全機能が有効になっていますが、モジュールは限られた時間の間のみ機能します。
0x02	Standard	Anybus IPは機能限定で実行されています。
0x03	Extended	全機能が有効になっています。

下記も参照してください。

- *Anybusステートマシン, ページ 43*

例外コード

EXCEPTION状態にあるとき、このアトリビュートは詳細情報を提供します。

#	文字列	説明
00h	No exception	-
01h	Application timeout	指定したウォッチドッグタイムアウト時間内にホストアプリケーションが応答しなかった。
02h	Invalid device address	実際のネットワークにおいて、選択したデバイスアドレスが有効でない。
03h	Invalid communication settings	実際のネットワークにおいて、選択した通信設定が有効でない。
04h	Major unrecoverable app event	ホストアプリケーションにより、修復不可能な重大なイベントが診断オブジェクトに通知された。
05h	Wait for reset	モジュールがホストアプリケーションのリセット実行待ちになっている。
06h	Invalid process data config	プロセスデータのコンフィグレーションが無効である。
07h	Invalid application response	コマンドに対してホストアプリケーションが無効な応答を返した。
08h	Nonvolatile memory checksum error	チェックサムエラーの発生により、不揮発性メモリに格納されている1つ以上のパラメータがデフォルト値に戻された。
09h	ASM communication error	Anybus CompactComモジュール・Anybusセーフティモジュール間の通信が失われた。
0Ah	Insufficient application implementation	Anybusモジュールが処理を継続するのに必要な機能がアプリケーションに実装されていない。
0Bh	Missing serial number	モジュールにシリアル番号がない。 これは、シリアル番号が埋め込まれていない製品構成（Anybus IPなど）で、アプリケーションが番号を供給できない場合に発生する。
0Ch	Corrupt file system	ファイルシステムが破損しているため、ユーザーによるフォーマットが必要です。
（その他）	(予約)	-

下記も参照してください。

- *Anybusステートマシン, ページ 43*

オブジェクト固有のエラーコード

Setup completeアトリビュートの設定に対する応答として、以下に示すオブジェクト固有のエラーコードがモジュールにより返される場合があります。

#	エラー	説明
01h	Invalid process data configuration	プロセスデータのコンフィグレーションが無効である。
02h	Invalid device address	実際のネットワークにおいて、選択したデバイスアドレスが有効でない。
03h	Invalid communication settings	実際のネットワークにおいて、選択した通信設定が有効でない。

12.4 診断オブジェクト (02h)

カテゴリ

産業用ネットワークによって異なります。ネットワークガイドを参照してください。

オブジェクトの説明

このオブジェクトは、ネットワークに診断イベントを通知するための標準化された方法を提供します。 それが実際にどのように表されるかは、ネットワークによって異なります。ただし、修復不可能な重大なイベントが発生したときにモジュールがEXCEPTION状態に陥るのは、すべての実装において共通です。

モジュールが起動して初期化されたとき、モジュールには診断オブジェクトのインスタンスが存在しません。 アプリケーションにおいてヒューズ切れなどの診断イベントが発生すると、重大度やイベントの種類に関する情報を持つインスタンスがアプリケーションによって作成されます。 このインスタンスに格納される情報は、アプリケーションがそのインスタンスを削除するまで、アプリケーションにて利用可能です。 このインスタンスに格納されるイベントコードはモジュールにより処理され、イベントに関する正しいネットワーク固有の情報を使用しているネットワークに転送します。

サポートコマンド

オブジェクト：	Get Attribute (01h)
	Create (03h)
	Delete (04)
インスタンス：	Get Attribute (01h)

オブジェクトアトリビュート（インスタンス**#0**）

#	名前	アクセス	データ型	値
1	Name	Get	CHAR配列	“Diagnostic”
2	Revision	Get	UINT8	01h
3	Number of instances	Get	UINT16	（作成された診断イベントの数による）
4	Highest instance no.	Get	UINT16	（ネットワーク固有）
11	Max no. of instances	Get	UINT16	作成可能なインスタンスの最大数（ネットワーク固有） 最大インスタンス数の中でも常に1つのインスタンスは、モジュールを強制的にEXCEPTION状態にするために、重大度レベル「Major, unrecoverable」のイベント用に予約しておかなければなりません。
12	Supported functionality	Get	BITS32	ビット0: ラッチイベントがサポートされている場合1 ラッチイベントがサポートされていない場合0 ビット1～31: 予約（0とする）

インスタンスアトリビュート (インスタンス**#1 ~ N**)

#	名前	アクセス	種類	説明
1	Severity	Get	UINT8	このアトリビュートはビットフィールドとして扱います。 ビット0 (LSB) は、インスタンスにより拡張診断が使用されているかどうかを表します。 • ビット0 = 1: 拡張診断が使用されている • ビット0 = 0: 拡張診断が使用されていない ビット4 ~ 6は、重大度レベルの情報で使用されます。 下記を参照してください。
2	Event Code	Get	UINT8	下記を参照してください。
3	NW specific extension	Get	UINT8の配列	ネットワーク固有のイベント情報 (オプション) 。
4	Slot	Get	UINT16	診断イベントがモジュラーデバイスのどのスロットに関連付けられているかを表します。 デフォルト値： 0 詳細については134ページの「モジュラーデバイスオブジェクト (ECh) 」を参照してください
5	ADI	Get	UINT16	診断イベントがどのADIに関連付けられているかを表します。 デフォルト値： 0 (診断インスタンスがどのADIにも関連付けられていない場合)
6	Element	Get	UINT8	診断イベントがADIのどの要素に関連付けられているかを表します。 値255は、診断イベントがADI全体に関連付けられていることを表します。 デフォルト値： 255
7	Bit	Get	UINT8	診断イベントが要素のどのビットに関連付けられているかを表します。 値255は、診断イベントが要素全体に関連付けられていることを表します 。 デフォルト値： 255

Severity

このパラメータは、イベントの重大度を表します。 ビット4 ~ 6のみ重大度レベルの情報で使用されます。

重大度レベル

ビットの組み合わせ	**Severity**	コメント
000	軽微、修復可能	-
001	軽微、修復不可能	修復不可能なイベントが検出された。
010	重大、修復可能	-
011	重大、修復不可能	EXCEPTION状態への移行が発生する。
101	軽微、ラッチ中	
110	重大、ラッチ中	
(その他)	-	(将来のために予約)

修復可能なイベントは、エラーの原因が取り除かれたときに、アプリケーションにて削除してください。

修復不可能なイベントは削除できません。 このイベントは、Anybus CompactComがリセットされるか、電源がオフにされるまでアクティブのままになります。

ラッチされたイベントは、ネットワークのマスターにより明示的に了解されるまでアクティブのままになります。 ラッチされた診断イベントの了解がネットワークでサポートされていない場合、モジュールは、ラッチされた診断イベントの作成を拒否します。

ネットワークのマスターが1つまたは複数のラッチイベントを了解すると、モジュールは、「Reset Diagnostic」要求をアプリケーションオブジェクトに送信します。 この要求には、マスターが了解したい診断インスタンスのリストが含まれています。 アプリケーションオブジェクトは、モジュールが削除してもよい診断インスタンスのリストを含む応答を返します。 すると、モジュールは、許可されたインスタンスを削除し、該当する情報をネットワークのマスターに通知します。

イベントコード

#	意味	コメント
10h	一般エラー	-
20h	電流	-
21h	電流、デバイス入力側	-
22h	電流、デバイス内部	-
23h	電流、デバイス出力側	-
30h	電圧	-
31h	電源電圧	-
32h	デバイス内部電圧	-
33h	出力電圧	-
40h	温度	-
41h	周囲温度	-
42h	デバイス温度	-
50h	デバイスハードウェア	-
60h	デバイスソフトウェア	-
61h	内部ソフトウェア	-
62h	ユーザーソフトウェア	-
63h	データセット	-
70h	追加モジュール	-
80h	監視	-
81h	通信	-
82h	プロトコルエラー	-
90h	外部エラー	-
F0h	追加機能	-
FFh	ネットワーク固有	ネットワーク固有の定義。詳細は各ネットワークガイドを参照してください。

コマンドの詳細: Create

詳細

コマンドコード: 03h

有効な対象： オブジェクト

説明

新規インスタンスを作成します。ここでいうインスタンスとは、ホストアプリケーションの新規診断イベントのことです。

- コマンド詳細：

フィールド	内容	注
CMDExt[0]	ビット0: 拡張診断 ビット4～6: Severity その他のビット 予約。 ゼロに設定します。	
CMDExt[1]	イベントコード。前のページを参照。	
MsgData[0～1]	イベントと関連付けられたスロット番号。 不明またはサポートされていない場合は0に設定	これらのフィールドはビット0（拡張診断）がセットされている場合にのみ存在
MsgData[2～3]	イベントと関連付けられたADI。 不明またはサポートされていない場合は0に設定	
MsgData[4]	イベントと関連付けられた要素。 不明またはサポートされていない場合は255に設定	
MsgData[5]	イベントと関連付けられたビット要素。 不明またはサポートされていない場合は255に設定	
MsgData[6～7]	予約。 ゼロに設定	
MsgData[0/8～n]	ネットワーク固有の拡張機能（任意。定義はネットワーク固有）。	CmdExt[0]のビット0がセットされている場合はMsgData[8～n] CmdExt[0]のビット0がセットされていない場合はMsgData[0～n]

- 応答詳細（成功）：

フィールド	内容
MsgData[0～1]	コマンドの結果として作成されたインスタンスの番号

- 応答詳細（エラー）：

エラー	内容	コメント
Object Specific Error	MsgData[1] = 02h	エラーコード（イベントのラッチはサポートされていない）。 モジュールがイベントのラッチをサポートしていないためイベントを作成できなかった。
	MsgData[1] = FFh	エラーコード（ネットワーク固有のエラー）。 ネットワーク固有の理由によりイベントを作成できなかった。 イベントに関する情報は、応答のMsgData[2-n]に格納されている。

コマンドの詳細: Delete

詳細

コマンドコード:	04h
有効な対象：	オブジェクト

説明

既存のインスタンス、すなわち過去に作成された診断イベントを削除します。

修復不可能なイベントおよびラッチされたイベントを表すインスタンスは削除できません。

- コマンド詳細：

フィールド	内容
CMDExt[0]	削除するインスタンスの番号（下位バイト）
CMDExt[1]	削除するインスタンスの番号（上位バイト）

- 応答詳細（エラー）：

エラー	内容	コメント
Object Specific Error	MsgData[1] = 01h	エラーコード（削除できなかった） イベントそのものが修復不可能であるか、ラッチされたイベントであるか、またはネットワーク固有の理由により、イベントを削除できなかった。
	MsgData[1] = FFh	エラーコード（ネットワーク固有のエラー）。 ネットワーク固有の理由によりイベントを削除できなかった。 イベントに関する情報は、応答のMsgData[2-n]に格納されている。

下記も参照してください。

– エラーコード, ページ 52

12.5 ネットワーク オブジェクト (03h)

カテゴリ

基本

オブジェクトの説明

このオブジェクトは、ネットワークに関する一般的な情報（すなわち、ネットワークの種類、データ形式など）を持ちます。 また、ホストアプリケーション側からADIをプロセスデータとしてマッピングするのにも使用されます。

下記も参照してください。

- ファンクショナルセーフティオブジェクト *(E8h), ページ 103*
- アプリケーションオブジェクト（*FFh*）, *ページ 114*

サポートコマンド

オブジェクト：	Get_Attribute（01h）
インスタンス：	Get_Attribute（01h）
	Set_Attribute (02h)
	Get_Enum_String (06h)
	Map_ADI_Write_Area (10h)
	Map_ADI_Read_Area (11h)
	Map_ADI_Write_Ext_Area (12h)
	Map_ADI_Read_Ext_Area (13h)

オブジェクトアトリビュート（インスタンス#0）

#	名前	アクセス	データ型	値
1	Name	Get	CHAR配列	“Network”
2	Revision	Get	UINT8	02h
3	Number of instances	Get	UINT16	（モジュールの種類による）
4	Highest instance no.	Get	UINT16	（モジュールの種類による）

インスタンスアトリビュート（インスタンス**#1**）

#	名前	アクセス	カテゴリ	種類	説明
1	Network type	Get	拡張	UINT16	（各ネットワークガイドおよび/または下表を参照してください）
2	Network type string	Get	-	CHAR配列	
3	Data format	Get	基本	ENUM	ネットワークのデータ形式 値 文字列 00h: “LSB First” 01h: “MSB First”
4	Parameter data support	Get	拡張	BOOL	このアトリビュートは、ネットワークがアサイクリックなデータサービスをサポートするかどうかを表す。 どのADIをプロセスデータにマッピングするかを決定するのにも使用可能。 値 意味 True: ネットワークはアサイクリックデータアクセスをサポートする False: アサイクリックデータをサポートしない
5	Write Process Data size	Get	-	UINT16	現在のライトプロセスデータのサイズ（単位：バイト）。Map_ADI_Write_Area、Map_ADI_Write_Ext_Area、Remap_ADI_Write_Area、またはネットワーク固有のマッピングコマンドが成功するたびに更新されます。
6	Read Process Data size	Get	-	UINT16	現在のリードプロセスデータのサイズ（単位：バイト）。Map_ADI_Read_Area、Map_ADI_Read_Ext_Area、Remap_ADI_Read_Area、またはネットワーク固有のマッピングコマンドが成功するたびに更新されます。
7	Exception Information	Get	-	UINT8	モジュールがEXCEPTION状態になった場合、ネットワーク固有の追加情報がここに表示されます（各ネットワークガイドを参照）。

Network type	**Network Type String**
0005h	“PROFIBUS DP-V1”
0025h	“DeviceNet”
0087h	“EtherCAT”
0089h	“PROFINET IRT”
0090h	“CC-Link”
0093h	“Modbus TCP”
009Bh	“EtherNet/IP”
009Dh	“PROFINET IRT Fiber Optic”
009Eh	“CC-Link IE Field”
009Fh	“Ethernet POWERLINK”
00A3h	“Common Ethernet”

コマンドの詳細: Map_ADI_Write_Area

詳細

コマンドコード:	10h
有効な対象：	インスタンス

説明

このコマンドは、ADIをライトプロセスデータとしてマッピングします。 成功すると、マッピングされたADIのライトプロセスデータイメージ先頭からのオフセットが応答データに格納されます。

- ネットワークによっては、2回以上にわたるADIのマッピング（すなわち、リードプロセスデータまたはライトプロセスデータに複数回マッピングしたり、リードプロセスとライトプロセスの両方にマッピングしたりすること）は許容されない場合があるため、ADIは2回以上マッピングしない ことを強く推奨します。
- ADIの一部のみマッピングすることはできません。必ず、ADIのすべての要素をマッピングしてください。
- 1つの領域に対し、マッピング用のコマンドであるMap_ADI_Read/Write_AreaとMap_ADI_Read/Write_Ext_Areaを組み合わせて使用することはできません。
- このコマンドを使用して、断片的なビットサイズ（BIT1～BIT7）を持つBITSx型またはPADx型をマッピングすることはできません。
- このコマンドでは変数と配列のみマッピングできます。 構造体をマップすることはできません。
- 一部のAnybusの実装では、実行中にネットワークによるプロセスデータのリマップが可能です。 詳細については、アプリケーションデータオブジェクト（*FEh*）, ページ *105*を参照してください。

下記も参照してください。

- アプリケーションオブジェクト（*FFh*）, ページ *114*

> **!** エラーチェックはコマンドパラメータに対してのみ行います。 Anybusモジュールは、実際のADIアトリビュートを読み込み、これらのコマンドパラメータが正しいかどうかを検証することは行いません。

- コマンド詳細：

フィールド	内容
CmdExt[0]	ADIのインスタンス番号（下位バイト）
CmdExt[1]	ADIのインスタンス番号（上位バイト）
MsgData[0]	ADIのデータ型（データ形式, ページ *49*を参照）
MsgData[1]	ADIの要素数
MsgData[2]	ADIのオーダー番号（下位バイト）
MsgData[3]	ADIのオーダー番号（上位バイト）

マッピングコマンドにおけるオーダー番号は、アプリケーションデータオブジェクトのGet_Instance_Number_By_Orderコマンドの番号と同じです。

- 応答詳細（成功）：

フィールド	内容
MsgData[0]	マッピングされたADIの、ライトプロセスデータの先頭からのオフセット。

- 応答詳細（エラー）：

エラー	内容	
Invalid CmdExt[0]	ADI番号が無効。	
Invalid State	ADIのマッピングはSETUP状態でのみ可能	
Object Specific Error	オブジェクト固有のエラー。詳細はMsgData[1]を参照。	
	01h: Invalid data type	このデータ型はプロセスデータに対して有効でない。
	02h: Invalid number of elements	要素数が有効でない（ゼロ）。
	03h: Invalid total size	結果として得られるトータルデータサイズが許容最大値（ネットワークの種類により異なる）を超えたため、要求されたマッピングが拒否された。
	04h: Multiple mapping	指定したネットワークではADIの複数マッピングが許容されていないため、要求されたマッピングが拒否された。
	05h: Invalid Order Number	オーダー番号が無効（ゼロ）である。
	06h: Invalid map command sequence	コマンドを受信した順序が無効である。

エラーチェックはコマンドパラメータに対してのみ行います。 Anybusモジュールは、実際のADIアトリビュートを読み込み、これらのコマンドパラメータが正しいかどうかを検証することは行いません。

コマンドの詳細: Map_ADI_Read_Area

詳細

コマンドコード: 11h

有効な対象： インスタンス

説明

このコマンドは、ADIをリードプロセスデータにマッピングすること以外は、上述の通りMap_ADI_Write_Areaと同じです。

コマンドの詳細: Map_ADI_Write_Ext_Area

詳細

コマンドコード:	12h
有効な対象：	インスタンス

説明

このコマンドはAnybus CompactCom 40デバイスでのみサポートされています。

このコマンドはMap_ADI_Write_Areaと同じですが、256バイトを超えるデータをマッピングできます。 このコマンドは、断片的なバイトサイズを持つ型のマッピングをサポートします。また、ADIの特定の部分のみのマッピングに使用することが可能です。

このコマンドは、ADIをライトプロセスデータとしてマッピングします。 成功すると、マッピングされたADIのライトプロセスデータ領域先頭からのビット単位のオフセットが応答データに格納されます。

- ネットワークによっては、2回以上にわたるADIのマッピング（すなわち、リードプロセスデータまたはライトプロセスデータに複数回マッピングしたり、リードプロセスとライトプロセスの両方にマッピングしたりすること）は許容されない場合があります。
- 1つの領域に対し、マッピング用のコマンドであるMap_ADI_Read/Write_AreaとMap_ADI_Read/Write_Ext_Areaを組み合わせて使用することはできません（リード/ライト）。
- データ領域のオフセットは最初のマッピングアイテムにのみ与えられ、1つのエラーコードを使用してすべてのマッピングアイテムが拒絶される可能性があるため、初期開発の段階では、各マッピングコマンドにおいて1つのアイテムのみマッピングすることを推奨します。
- BIT1～BIT7およびPADxを除き、マッピングされるすべての要素はバイト境界に配置されている必要があります。
- 設定完了時にプロセスデータのサイズがバイト単位となっている必要があるため、最後にマッピングされたアイテムに限り、バイト境界まで暗黙のパディングが行われます。
- 明示的なパディングは、利用可能なADIのPADx型要素、または、255個のPAD1型要素を持つ配列であると仮定される仮想ADI 0を用いて行われます。 プロセスデータの明示的なパディングが、ADI 0の唯一の正しい使用方法です。ネットワーク上ではパディングビットが見えない場合があります。
- このコマンドは、コマンドに対してエラーが返された場合でも、Anybus CompactCom 40の状態を恒久的に変える可能性があります。 ネットワーク固有の制約により、n個のマッピングアイテムが受け付けられたとしても、マッピングアイテムn+1に対してエラーが発生する場合があります。この場合、nまでのマッピングは受け付けられますが、n+1以降のすべてのマッピングアイテムは拒絶されます。 受け付けられたマッピングの数は、応答のCmdExt[0]にて通知されます。
- 一部のAnybusの実装では、実行中にネットワークによるプロセスデータのリマップが可能です。 詳細については、アプリケーションデータオブジェクト（*FEh*）, ページ *105*を参照してください。

下記も参照してください。

アプリケーションオブジェクト（*FFh*）, ページ *114*

> **!** エラーチェックはコマンドパラメータに対してのみ行います。 Anybusモジュールは、実際のADIアトリビュートを読み込み、これらのコマンドパラメータが正しいかどうかを検証することは行いません。

- コマンド詳細：

フィールド	内容
CmdExt[0]	追加するマッピングアイテムの数（0～217）
CmdExt[1]	予約。 0に設定。
MsgData[0-1]	新規マッピングアイテム1： ADI番号
MsgData[2]	新規マッピングアイテム1： ADIの要素数
MsgData[3]	新規マッピングアイテム1： マッピングする最初の要素へのインデックス（0～254）
MsgData[4]	新規マッピングアイテム1： 後ろに続くマッピングする要素の数（1～255）
MsgData[5]	新規マッピングアイテム1： 型記述子の数（1～255）
MsgData[6~n]	新規マッピングアイテム1： マッピングされた各要素の型指定子の配列
...	マッピングアイテム2以降は、MsgData[0-n]（上記）を繰り返す。

- 応答詳細（成功）：

フィールド	内容
CmdExt[0]	受け付けられたマッピングアイテムの数（0～217）
MsgData[0]	マッピングされたADIの、ライトプロセスデータの先頭からのビットオフセット（最下位バイト）。
MsgData[1]	マッピングされたADIの、ライトプロセスデータの先頭からのビットオフセット 。
MsgData[2]	マッピングされたADIの、ライトプロセスデータの先頭からのビットオフセット 。
MsgData[3]	マッピングされたADIの、ライトプロセスデータの先頭からのビットオフセット（最上位バイト）。

- 応答詳細（エラー）：

エラー	内容	
Invalid CmdExt[0]	エラーが発生する前に受け付けられたマッピングアイテムの数	
Invalid State	ADIのマッピングはSETUP状態でのみ可能	
Object Specific Error	オブジェクト固有のエラー。詳細はMsgData[1]を参照。	
	01h: Invalid data type	このデータ型はプロセスデータに対して有効でない。
	02h: Invalid number of elements	要素数が有効でない（ゼロ。または要素数が多すぎる）。
	03h: Invalid total size	結果として得られるトータルデータサイズが許容最大値（ネットワークの種類により異なる）を超えたため、要求されたマッピングが拒否された。
	06h: Invalid map command sequence	コマンドを受信した順序が無効である。
	07h: Invalid mapping command	コマンドに矛盾があるため、コマンドを構文解析できない。
	08h: Bad alignment	プロセスデータのアライメント規則が守られていない。
	09h: Invalid use of ADI 0	ADI 0は、PAD1型の配列（要素数255）である。
	FFh: Network specific restriction	応答のData[2-n]に示されるネットワーク特有の理由によりマッピングが拒否された。 該当するネットワークガイドを参照すること。

エラーチェックはコマンドパラメータに対してのみ行います。 Anybusモジュールは、実際のADIアトリビュートを読み込み、これらのコマンドパラメータが正しいかどうかを検証することは行いません。

コマンドの詳細: Map_ADI_Read_Ext_Area

詳細

コマンドコード: 13h

有効な対象： インスタンス

説明

このコマンドはAnybus CompactCom 40デバイスでのみサポートされています。

このコマンドはMap_ADI_Read_Areaと同じですが、256バイトを超えるデータをマッピングできます。

ADIをリードプロセスデータにマッピングすること以外は、上述の通りMap_ADI_Write_Ext_Areaと同じです。

12.6 ネットワークコンフィグレーションオブジェクト（04h）

カテゴリ

ネットワーク固有

オブジェクトの説明

このオブジェクトには、エンドユーザーにより設定されるネットワーク特有のコンフィグレーションパラメータ（通常は、ボーレートやノードアドレスなどの設定）が格納されます。このオブジェクトにおけるインスタンスの実際の定義はネットワーク固有ですが、インスタンス1と2は固定です（設定可能な場合）。

設定可能な場合、これらのインスタンスに対して以下の仕様が適用されます。

インスタンス番号	データ型	パラメータ
1	8ビットまたは16ビットの任意のデータ型	現在選択されているネットワークデバイスアドレス（または同様の情報）
2	8ビットまたは16ビットの任意のデータ型	現在選択されているネットワーク通信ビットレート（または同様の情報）。

このオブジェクトのインスタンスの値は、元になる値が変更されるたびに更新する必要があります。 そのため、メカニカルスイッチなどをホストアプリケーションにて継続的に監視する必要があります。

- 「shared」アクセスと示されたインスタンス（Descriptorにより示される）は、**volatile**とみなす必要があります。 このインスタンスに「set」アクセスを行った場合、必ずしも値が変わるとは限りません。 値が影響を受けない場合、Anybusモジュールはエラーを返しません。
- 16ビットのインスタンスに対して8ビットのデータ設定要求が行われた場合、この設定要求は受け付けられて、上位8ビットには0が設定されます。
- 8ビットのインスタンスに対して16ビットのデータ設定要求が行われた場合、この設定要求は受け付けられて、上位8ビットは破棄されます。

入力デバイスの区別

Anybusモジュールは、「hardwired」入力デバイス（すなわち、物理的なメカニカルスイッチ）からのパラメータと、キーパッドやディスプレイなどのソフト入力デバイスで指定されたパラメータを区別します。 これにより、Anybusモジュールは、パラメータの実際の入力元に関するネットワーク固有の要求（例えば、ネットワークによっては、物理スイッチの値の変化をオンボードLEDで視覚的に確認する必要があります）を満たすことが可能です。

この区別は、ホストアプリケーションからの以下のアクションにより行われます（表を参照）。

状態	アクション（ホストアプリケーション）	**Anybus**の動作
SETUP	物理スイッチからのパラメータをポーリング/更新する（影響を受ける各パラメータに対し、最低1回Setコマンドを発行すること）。「soft」入力デバイスからのパラメータは更新しないこと（これらのパラメータについては、この時点ではSetコマンドを発行しないこと）。	Anybusモジュールは、影響を受けたパラメータを物理スイッチからのパラメータと認識する。 残りのパラメータは「soft」入力デバイスからのパラメータとみなされる。
（その他の状態）	必要に応じてすべてのパラメータ（物理スイッチと「soft」入力手段によるパラメータ）をポーリング/更新する。	Anybusモジュールは、SETUP状態において更新されたパラメータを記録しています。これにより、Anybusモジュールは、ネットワークから要求されたときにそれらを区別して扱うことが可能です。

サポートコマンド

オブジェクト：	Get_Attribute（01h）
	Reset（05h）（実際の動作はネットワークにより異なる）
インスタンス：	Get_Attribute（01h）
	Set_Attribute (02h)
	Get_Enum_String (06h)

オブジェクトアトリビュート（インスタンス#0）

#	名前	アクセス	データ型	値
1	Name	Get	CHAR配列	“Network Configuration”
2	Revision	Get	UINT8	01h
3	Number of instances	Get	UINT16	（ネットワークにより異なる）
4	Highest instance no.	Get	UINT16	（ネットワークにより異なる）

インスタンスアトリビュート（インスタンス #1 ~ n）

各インスタンスは、ネットワークコンフィグレーションパラメータを表します。 インスタンス内のアトリビュートは、パラメータの全体的な説明（名前、データ型など）を提供します。 インスタンス名と列挙型文字列は多言語に対応しています。 実際の文字列は当然ながらネットワークによって異なりますが、最大文字数は13文字に制限されています。

#	名前	アクセス	カテゴリ	種類	説明
1	Name	Get	アプリケーション固有	CHAR配列	パラメータ名（“Node address”など）
2	Data type	Get	アプリケーション固有	UINT8	データ型（データ形式, ページ 49を参照）
3	Number of elements	Get	アプリケーション固有	UINT8	指定したデータ型の要素数
4	Descriptor	Get	アプリケーション固有	UINT8	パラメータのアクセス権を指定するビットフィールド。 ビット　アクセス権 b0:　1: Getアクセス b1:　1: Setアクセス b2:　1: Sharedアクセス Sharedアクセスとタグ付けされたインスタンスは、**volatile**であるとみなす必要があります。このようなインスタンスにSetアクセスを行った場合、必ずしも値が変わるとは限りません。 値が影響を受けない場合、Anybusモジュールはエラーを返しません。
5	Value	アトリビュート#4によって決定	アプリケーション固有	アトリビュート #2によって決定	実際のパラメータ値。 不揮発性メモリに格納される。 Getアクセス： 実際に使用されている値が返される Setアクセス： 設定された（場合によっては実際の）値が書き込まれる
6	Configured value	Get	アプリケーション固有	アトリビュート #2によって決定	設定されたパラメータ値。 アトリビュートの設定値を返す。 これは、設定時にValueを直接使用しない場合に便利です（電源のオフ/オンが必要な場合など）。

インスタンス#1と#2が存在する場合、基本カテゴリに分類されます。 このオブジェクトにおける他のすべてのインスタンスは、各ネットワークガイドにて分類されています。

12.7 Anybus ファイルシステムインターフェース・オブジェクト (0Ah)

カテゴリ

拡張

オブジェクトの説明

このオブジェクトは内蔵ファイルシステムとのインターフェースになります。 各インスタンスはファイル ストリームへのハンドルを表し、ファイルシステム操作のためのサービスを含んでいます。 これにより、ホストアプリケーションは、モジュールの組み込みファイルシステムにアクセスできます。 インスタンスは、実行時に動的に作成および削除されます。

オブジェクトは、アプリケーション・ファイルシステムインターフェース・オブジェクト (EAh) と、構造的にほぼ同じです。アプリケーション・ファイルシステムインターフェース・オブジェクト *(EAh), ページ 122*を参照してください。

*Ethernet*モジュールには、ファームウェアのダウンロード/アップグレードや内部*Web*ページなどの別の目的でアプリケーションからアクセスできるファイルシステムが組み込まれています。 詳細については、それぞれのネットワークガイドを参照してください。 その他すべてのモジュールには、フォルダが*1*つだけ存在します。 このフォルダはファームウェアのダウンロードとアップグレードのためにのみ使用されます。

サポートコマンド

オブジェクト：	Get_Attribute (01h)
	Set_Attribute (02h)
	Create (03h)
	Delete (04h)
	FormatDisc (30h)
インスタンス：	Get_Attribute (01h)
	File Open (10h)
	File Close (11h)
	File Delete (12h)
	File Copy (13h)
	File Rename (14h)
	File Read (15h)
	File Write (16h)
	Directory Open (20h)
	Directory Close (21h)
	Directory Delete (22h)
	Directory Read (23h)
	Directory Create (24h)
	Directory Change (25h)

オブジェクトアトリビュート（インスタンス**#0**）

#	名前	アクセス	データ型	値/説明
1	Name	Get	CHAR配列	“Anybus File System Interface”
2	Revision	Get	UINT8	02h
3	Number of instances	Get	UINT16	-
4	Highest instance no.	Get	UINT16	-
11	Max no. of instances	Get	UINT16	4: IT機能をサポートするAnybus CompactCom 40モジュールについてのみ有効です。 1: 複数のオープンファイル/ディレクトリストリームをサポートしていない、Anybus CompactCom 40モジュールに対して有効です。
12	Disable virtual file system	Set	BOOL	仮想ファイルシステムが無効になっている場合は、内部Webページにアクセスできません。 0 =仮想ファイルシステムが有効（デフォルト） 1 =仮想ファイルシステムが無効
13	Total disc size	Get	UINT32	ディスクサイズ（単位：バイト）。
14	Free disc size	Get	UINT32	空きディスクサイズ（単位：バイト）。

インスタンスアトリビュート（インスタンス**#1 ~ 4**）

#	名前	アクセス	カテゴリ	種類	説明
1	Instance Type	Get	拡張	UINT8	値 意味 0： 予約 1: ファイルインスタンス 2: ディレクトリインスタンス
2	File size	Get	拡張	UINT32	ファイルサイズ（ディレクトリは0）
3	Path	Get	拡張	CHAR配列	インスタンスが処理を行うファイルのパス

ファイルシステムエラー

ファイルシステムインターフェースオブジェクトを呼び出すサービスでエラーが発生した場合、モジュールはFFh（オブジェクト固有のエラー）を返します。 ファイルシステムエラーに関する説明は、エラー応答のデータフィールドに返されます。

#	名前	説明
1	FILE_OPEN_FAILED	ファイルをオープンできない
2	FILE_CLOSE_FAILED	ファイルをクローズできない
3	FILE_DELETE_FAILED	ファイルを削除できない
4	DIRECTORY_OPEN_FAILED	ディレクトリをオープンできない
5	DIRECTORY_CLOSE_FAILED	ディレクトリをクローズできない
6	DIRECTORY_CREATE_FAILED	ディレクトリを作成できない
7	DIRECTORY_DELETE_FAILED	ディレクトリを削除できない
8	DIRECTORY_CHANGE_FAILED	ディレクトリを変更できない
9	FILE_COPY_OPEN_READ_FAILED	コピー元のファイルをオープンできない
10	FILE_COPY_OPEN_WRITE_FAILED	コピー先のファイルをオープンできない
11	FILE_COPY_WRITE_FAILED	コピー時にファイルを書き込めない
12	FILE_RENAME_FAILED	ファイルの名前を変更できない

コマンドの詳細: File Open

詳細

コマンドコード: 10h

有効な対象： インスタンス

説明

読み取り、書き込み、追加を行うファイルをオープンします。

- コマンド詳細：

フィールド	内容	コメント
CmdExt[0]	00h - 読み取りモード	リードオンリーアクセス用にファイルをオープンする。
	01h - 書き込みモード	ライトオンリーアクセス用にファイルをオープンする。 指定されたファイルが存在しない場合、そのファイルが作成される。 指定されたファイルが存在する場合、そのファイルに上書きされる。
	02h - 追加モード	ファイル終端に書き込むようにファイルをオープンする。 指定されたファイルが存在しない場合、そのファイルが作成される。 指定されたファイルが存在する場合、このファイルに書き込んだデータはファイル終端に追加される。
CmdExt[1]	（予約、0）	-
MsgData[0～n]	オープンするファイルのパス + ファイル名（現在のパスからの相対パス）	-

- 応答詳細：

  （データなし）

コマンドの詳細: File Close

詳細

コマンドコード: 11h

有効な対象： インスタンス

説明

オープンしているファイルをクローズします。

- コマンド詳細：

  （データなし）

- 応答詳細：

フィールド	内容	コメント
CmdExt[0]	予約（0）	-
CmdExt[1]	予約（0）	-
MsgData[0]	ファイルサイズ（下位バイト）	クローズしたファイルのサイズ
MsgData[1]	File size	
MsgData[2]	File size	
MsgData[3]	ファイルサイズ（上位バイト）	

コマンドの詳細: File Delete

詳細

コマンドコード: 12h

有効な対象： インスタンス

説明

指定したファイルを削除します。

- コマンド詳細：

フィールド	内容
CmdExt[0]	（予約、0）
CmdExt[1]	（予約、0）
MsgData[0～n]	削除するファイルのパス + ファイル名（現在のパスに対する相対パス）

- 応答詳細：

  （データなし）

コマンドの詳細: File Copy

詳細

コマンドコード: 13h

有効な対象： インスタンス

説明

ファイルのコピーを作成します。

- コマンド詳細：

フィールド	内容
CmdExt[0]	（予約、0）
CmdExt[1]	（予約、0）
MsgData[0]~	コピー元ファイルのパス + ファイル名（現在のパスからの相対パス）
MsgData[x]	NULL（00h）
MsgData[y]~	コピー先ファイルのパス + ファイル名（現在のパスからの相対パス）

- 応答詳細：

  （データなし）

コマンドの詳細: File Rename

詳細

コマンドコード: 14h

有効な対象： インスタンス

説明

ファイルの名前を変更またはファイルを移動します。

- コマンド詳細：

フィールド	内容
CmdExt[0]	（予約、0）
CmdExt[1]	（予約、0）
MsgData[0]~	変更前のパス + ファイル名（現在のパスからの相対パス）
MsgData[x]	NULL（00h）
MsgData[y]~	変更後のパス + ファイル名（現在のパスからの相対パス）

- 応答詳細：

  （データなし）

コマンドの詳細: File Read

詳細

コマンドコード: 15h

有効な対象： インスタンス

説明

読み取り用にオープンしているファイルからデータを読み取ります。

- コマンド詳細：

フィールド	内容
CmdExt[0]	読み取るバイト数（下位バイト）
CmdExt[1]	読み取るバイト数（上位バイト）

- 応答詳細：

フィールド	内容
CmdExt[0]	（予約、0）
CmdExt[1]	（予約、0）
MsgData[0]~	ファイルから読み取ったデータ

コマンドの詳細: File Write

詳細

コマンドコード: 16h

有効な対象： インスタンス

説明

書き込み用または追加用にオープンしているファイルにデータを書き込みます。

- コマンド詳細：

フィールド	内容
CmdExt[0]	(予約、0)
CmdExt[1]	(予約、0)
MsgData[0]~	ファイルに書き込むデータ

- 応答詳細：

フィールド	内容
CmdExt[0]	書き込んだバイト数 (下位バイト)
CmdExt[1]	書き込んだバイト数 (上位バイト)

コマンドの詳細: Directory Open

詳細

コマンドコード: 20h

有効な対象： インスタンス

説明

ディレクトリをオープンします。

- コマンド詳細：

フィールド	内容
CmdExt[0]	(予約、0)
CmdExt[1]	(予約、0)
MsgData[0]~	開くディレクトリのパス + ファイル名 (現在のパスからの相対パス)

- 応答詳細：

  (データなし)

コマンドの詳細: Directory Close

詳細

コマンドコード:	21h
有効な対象：	インスタンス

説明

ディレクトリをクローズします。

- コマンド詳細：

  （データなし）

- 応答詳細：

  （データなし）

コマンドの詳細: Directory Delete

詳細

コマンドコード:	22h
有効な対象：	インスタンス

説明

ファイルシステムにあるディレクトリを削除します。 削除するディレクトリは空でなければなりません。 空でないディレクトリを削除しようとすると、エラーが発生します。

- コマンド詳細：

フィールド	内容
CmdExt[0]	（予約、0）
CmdExt[1]	（予約、0）
MsgData[0]~	削除するディレクトリのパス + ファイル名（現在のパスからの相対パス）

- 応答詳細：

  （データなし）

コマンドの詳細: Directory Read

詳細

コマンドコード:	23h
有効な対象：	インスタンス

説明

このコマンドは、Directory Openコマンドにより読み取り用にオープンしているディレクトリからデータを読み取ります。

コマンド発行のたびに、次のディレクトリエントリ（ファイルまたはディレクトリ）が返されます。 ディレクトリ内にあるすべてのエントリの読み取りが完了すると、応答データのサイズがゼロ（0）に設定され、メッセージデータが返されなくなります。これにより、そのディレクトリにはこれ以上エントリが存在しないことが示されます。

- コマンド詳細：

  （データなし）

- 応答詳細：

フィールド	内容
CmdExt[0]	予約（0）
CmdExt[1]	予約（0）
MsgData[0]	オブジェクトのサイズ（下位バイト）
MsgData[1]	オブジェクトのサイズ
MsgData[2]	オブジェクトのサイズ
MsgData[3]	オブジェクトのサイズ（上位バイト）
MsgData[4]	オブジェクトフラグ
MsgData[5] ~	オブジェクト名（ファイルまたはディレクトリ）

- オブジェクトフラグ

フィールド	内容
01h	オブジェクトはディレクトリである
02h	オブジェクトはリードオンリーである
04h	オブジェクトは隠し属性である
08h	オブジェクトはシステムオブジェクトである

コマンドの詳細: Directory Create

詳細

コマンドコード: 24h

有効な対象： インスタンス

説明

ファイルシステムにディレクトリを作成します。

- コマンド詳細：

フィールド	内容
CmdExt[0]	（予約、0）
CmdExt[1]	（予約、0）
MsgData[0]~	作成するディレクトリのパス + ファイル名（現在のパスからの相対パス）

- 応答詳細：

  （データなし）

コマンドの詳細: Directory Change

詳細

コマンドコード: 25h

有効な対象： インスタンス

説明

インスタンスのディレクトリ/パスを変更します。

- コマンド詳細：

フィールド	内容
CmdExt[0]	（予約、0）
CmdExt[1]	（予約、0）
MsgData[0]~	変更後のディレクトリのパス + ファイル名（現在のパスからの相対パス）

- 応答詳細：

  （データなし）

コマンドの詳細: Format Disc

詳細

コマンドコード:	30h
有効な対象：	オブジェクト

説明

ファイルシステムにあるディスクをフォーマットします (ディスク上のデータはすべて消去されます) 。

- コマンド詳細：

フィールド	内容
CmdExt[0]	フォーマットするディスク。 ゼロ (0) に設定
CmdExt[1]	(予約、0)

- 応答詳細：

  (データなし)

例

このセクションでは、エンドユーザーがファイルシステムインターフェースオブジェクトを使用する一般的な場合の例を示します。

この例では、ルートフォルダに以下のファイルが存在する架空のフォルダ構造を使用します。

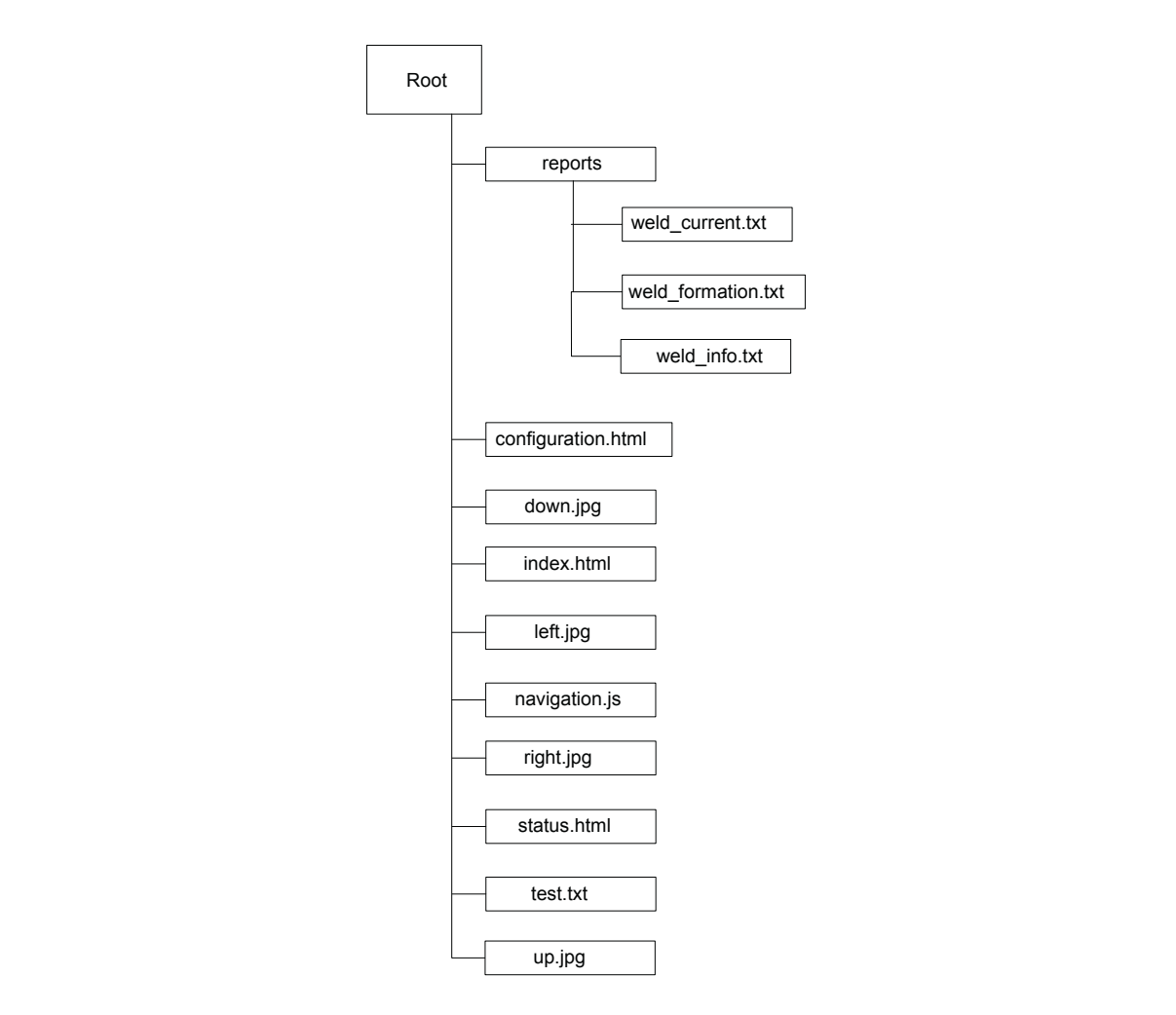

Fig. 17

ファイルの読み取り

次の例は、reportsフォルダにあるweld_info.txtを開き、このファイルからデータを読み取ります。

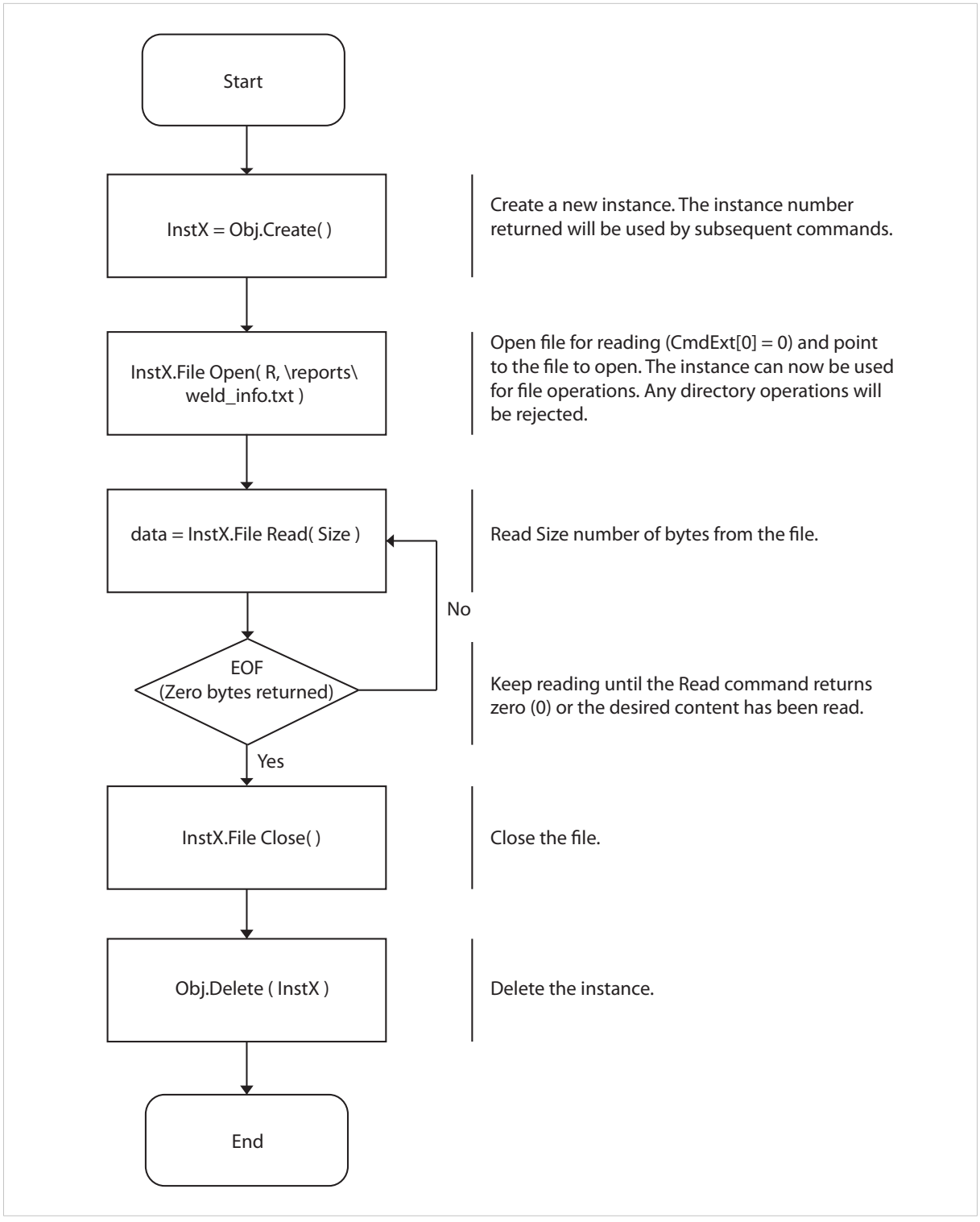

Fig. 18

ファイルの書き込み

次の例は、test.txtファイルを書き込みのために開きます。

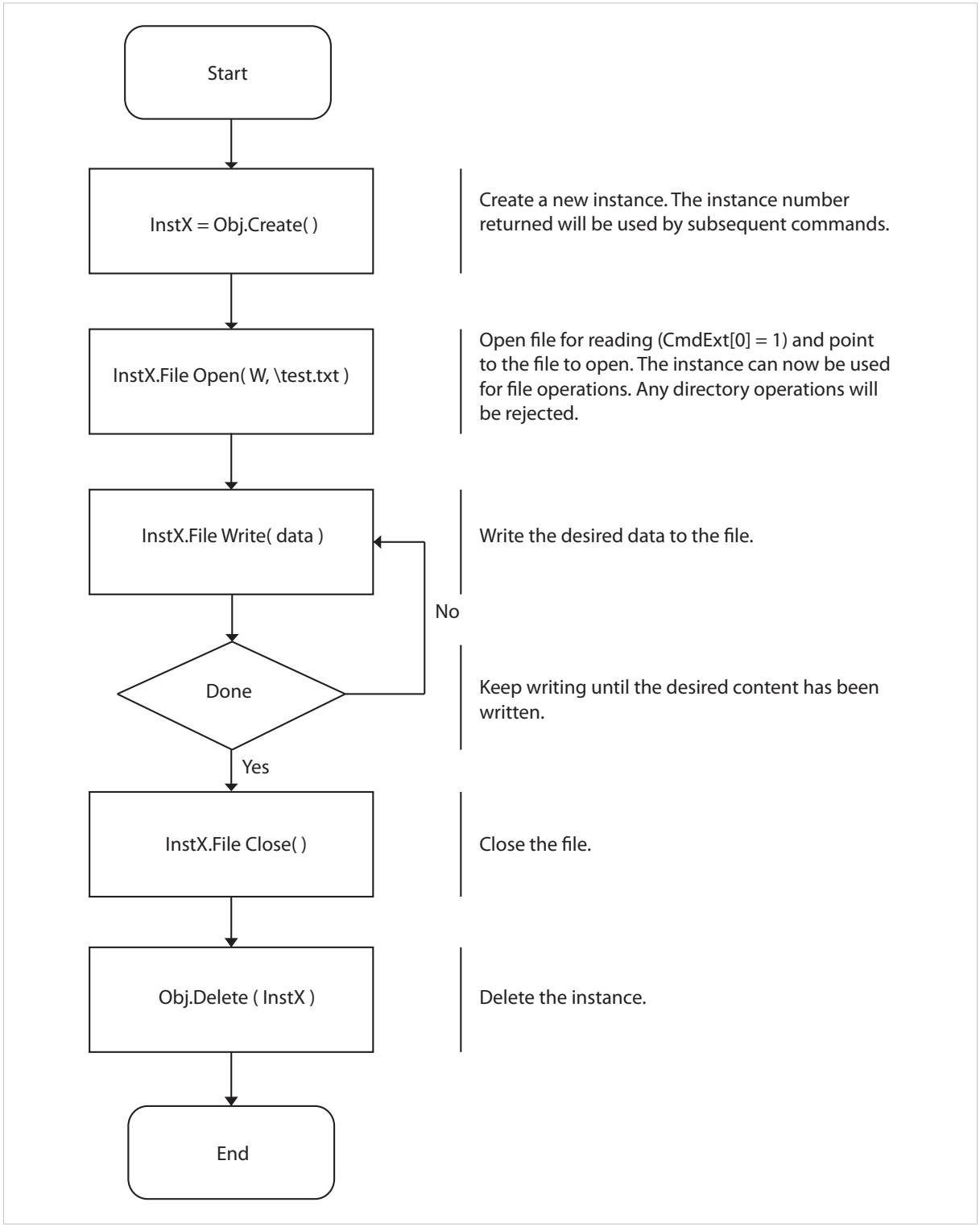

Fig. 19

ディレクトリ内容の一覧表示

次の例は、reportsディレクトリの内容を一覧表示します。

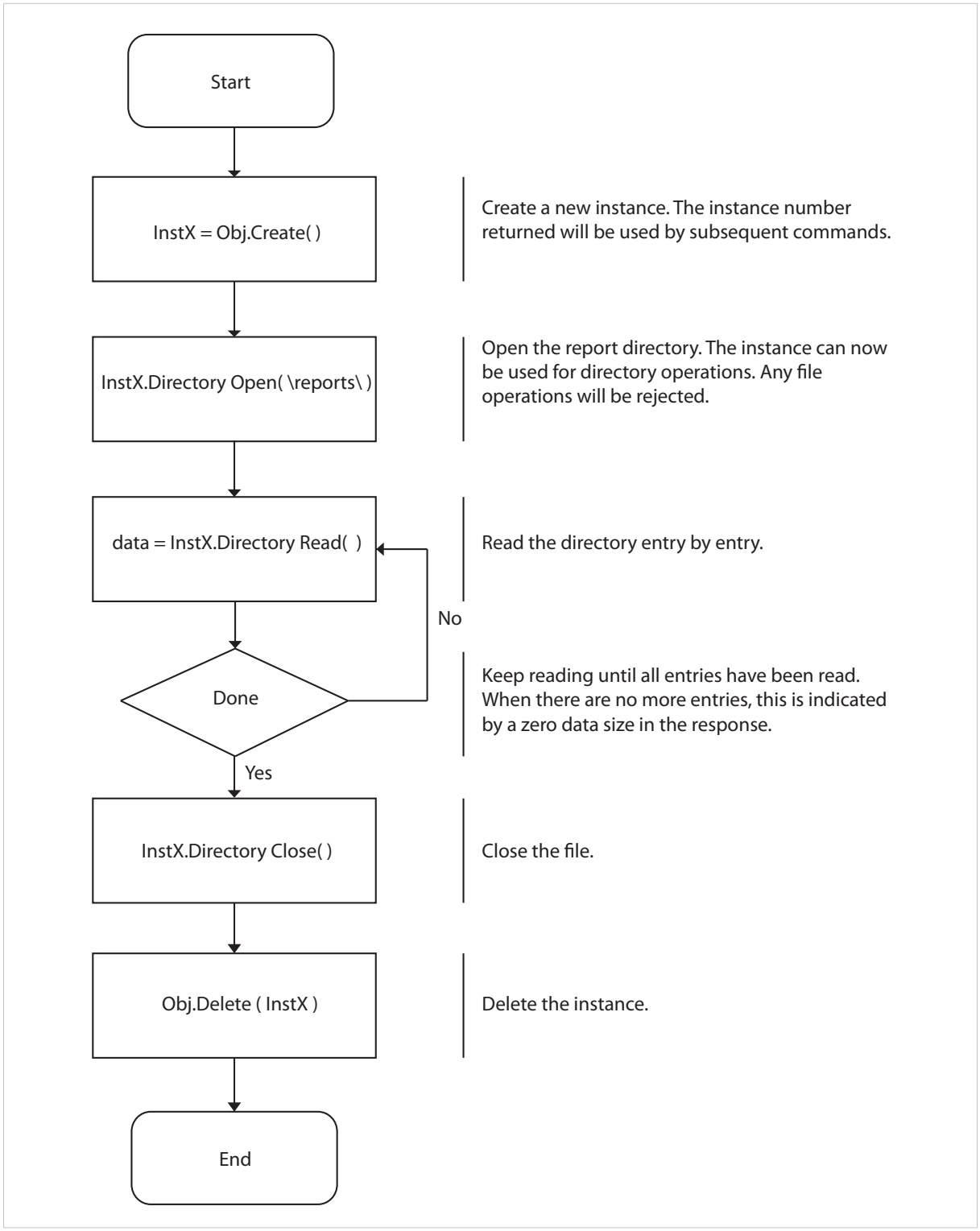

Fig. 20

12.8 ファンクショナルセーフティモジュール・オブジェクト (11h)

カテゴリ

拡張

オブジェクトの説明

このオブジェクトはAnybus CompactCom モジュールと接続されたセーフティモジュールによって提供される情報を格納します。 下記のアトリビュート値に関しては使用されるセーフティモジュールの説明書を参照ください。

サポートコマンド

オブジェクト：	Get_Attribute
	Error_Confirmation
	Set_IO_Config_String
	Get_Safety_Output_PDU
	Get_Safety_Input_PDU
インスタンス：	Get_Attribute

オブジェクトアトリビュート（インスタンス**#0**）

#	名前	アクセス	データ型	値
1	Name	Get	CHAR配列	“Functional Safety Module”
2	Revision	Get	UINT8	01h
3	Number of instances	Get	UINT16	0001h
4	Highest instance no.	Get	UINT16	0001h

インスタンスアトリビュート（インスタンス**#1**）

#	名前	アクセス	データ型	説明
1	State	Get	UINT8	セーフティモジュールの現在の状態 使用されるセーフティモジュールの説明書を参照ください。
2	Vendor ID	Get	UINT16	セーフティモジュールのベンダー識別子。 例：0001h（HMS Industrial Networks） 使用されるセーフティモジュールの説明書を参照ください。
3	IO Channel ID	Get	UINT16	セーフティモジュールが取り付けられているIOチャネルを表す。 使用されるセーフティモジュールの説明書を参照ください。
4	Firmware version	Get	構造体： UINT8 (メジャー) UINT8 (マイナー) UINT8 (ビルド)	セーフティモジュールのファームウェアバージョン。 フォーマット： バージョン「2.18.3」は以下のように表されます。 最初のバイト = 0x02、2番目のバイト = 0x12、3番目のバイト = 0x03。
5	Serial number	Get	UINT32	製造時にセーフティモジュールに割り当てられた、32ビットの番号。 使用されるセーフティモジュールの説明書を参照ください。
6	Output data	Get	UINT8の配列	セーフティモジュールの出力データ（ネットワークからのデータ）の現在値 注： このデータはAnybus CompactCom モジュールから提供されるので安全ではありません。
7	Input data	Get	UINT8の配列	セーフティモジュールの入力データ（ネットワークへのデータ）の現在値 注：このデータはAnybus CompactComモジュールから提供されるので安全ではありません。

#	名前	アクセス	データ型	説明
8	Error counters	Get	構造体： UINT16 (ABCC DR) UINT16 (ABCC SE) UINT16 (SM DR) UINT16 (SM SE)	エラー カウンター（各カンターはFFFFhで計数を停止）
				ABCC DR: Anybus CompactCom モジュールによって破棄されたセーフティモジュールからの（予期せぬ）応答。
				ABCC SE: Anybus CompactCom モジュールにより検出されたシリアル受信エラー
				SM DR: セーフティモジュールにより破棄されたAnybus CompactComモジュールからの（予期せぬ）応答。
				SM SE: セーフティモジュールによって検出されたシリアル受信エラー。
9	Event log	Get	UINT8の配列	最後に発生したセーフティモジュールのイベント情報（存在する場合）が、このアトリビュートに記録されます。 それより以前のイベント情報は新たなイベントが記録されるときに消去されます。 HMSサポートによる評価で使用されます。
10	Exception information	Get	UINT8	Anybusオブジェクト内の例外コードが「Safety communication error」（09h）に設定されている場合、詳細な例外情報がここに表されます（下表を参照）。
11	Bootloader version	Get	構造体： UINT8 メジャー UINT8 マイナー	セーフティモジュールのブートローダーバージョンです。 フォーマット： バージョン「2.12」は以下のように表されます。 最初のバイト = 0x02、2番目のバイト = 0x0C

Exception Information

例外コード09hがAnybusオブジェクトにセットされている場合、アプリケーション内のファンクショナルセーフティモジュールに関するエラーが発生しています。 例外情報はこの表に従ってインスタンスアトリビュート #10に表されます：

値	**Exception Information**
00h	情報なし
01h	ボーレートがサポートされていない
02h	開始メッセージが存在しない
03h	予期せぬメッセージ長
04h	応答に予期せぬコマンドが含まれている
05h	予期せぬエラーコード
06h	セーフティアプリケーションが見つからない
07h	セーフティアプリケーションのCRCが無効
08h	フラッシュアクセスなし
09h	ブートローダー通信中に間違ったセーフティプロセッサーから応答があった
0Ah	ブートローダーのタイムアウト
0Bh	ネットワーク固有のパラメータエラー
0Ch	無効なIOコンフィグレーション文字列
0Dh	セーフティマイクロプロセッサー間で応答が異なる（モジュールタイプが異なるなど））
0Eh	互換性のないモジュール（サポートしているネットワークなど）
0Fh	（CRCエラーなどにより）最大回数の再送が行われた
10h	ファームウェア ファイル エラー
11h	ファンクショナルセーフティ・ホストオブジェクトのアトリビュート #4にある周期時間の値は現在のボーレートでは使用できない
12h	起動テレグラムのSPDU入力サイズが無効
13h	起動テレグラムのSPDU出力サイズが無効
14h	不正形式の入力SPDU
15h	Anybusからセーフティモジュールへの初期化エラー

コマンドの詳細: Error_Confirmation

カテゴリ

拡張

詳細

コマンドコード	10h
有効な対象：	オブジェクト

説明

何らかの理由によりセーフティモジュールがセーフ状態になった場合、そのモジュールがセーフ状態から抜けるにはエラー確認を受信する必要があります。 このコマンドを使用すると、何らかの理由で同時にセーフ状態になっているセーフティモジュールのすべてのセーフティチャネルをリセットすることが可能です。 オペレータなどによってエラーがクリアされると、アプリケーションがこのコマンドをAnybus CompactComモジュールに発行します。 Anybus CompactComはこのコマンドをセーフティモジュールに転送します。

チャネルのセーフ状態は、安全PLCまたはセーフティモジュールによってもこのコマンドで確認できます。

- コマンド詳細

  (データなし)

- 応答詳細

  (データなし)

コマンドの詳細: Set_IO_Config_String

カテゴリ

拡張

詳細

コマンドコード	11h
有効な対象：	オブジェクト

説明

このコマンドは、セーフティ入出力のデフォルト コンフィグレーションを変更する必要があるとき、ホストアプリケーションから送られます。 この文字列は、コンフィグレーションをセーフティモジュールに提供するためにその他の手段 (例： PLCやその他のツール) が存在しないネットワークによって使用されます。 詳細はセーフティモジュールの仕様書を参照ください。 渡された文字列はHMSにより生成されており、このコマンドを使用してそのまま渡される必要があります。

この文字列についての情報は、文字列の送り先となるセーフティモジュールの仕様書を参照ください。

- コマンド詳細

フィールド	内容
CmdExt[0]	(未使用)
CmdExt[1]	
Data[0 ~ n]	データ(バイト文字列) このデータはIO コンフィグレーション文字列から成っており、そのデータフォーマットはセーフティ ネットワークに依存します。

- 応答詳細

  (データなし)

コマンドの詳細: Get_Safety_Output_PDU

カテゴリ

拡張

詳細

コマンドコード	12h
有効な対象：	オブジェクト

説明

このコマンドは、PLCによって送信された完全な安全PDU出力を取得するために、アプリケーションから発行できます。Anybus CompactCom 40は完全な安全PDUで応答します。アプリケーションはこれを解釈する必要があります。

- コマンド詳細

  (データなし)

- 応答詳細

フィールド	内容
CmdExt[0]	(未使用)
CmdExt[1]	
Data[0～n]	PLCからの安全PDU

コマンドの詳細: Get_Safety_Input_PDU

カテゴリ

拡張

詳細

コマンドコード	13h
有効な対象：	オブジェクト

説明

このコマンドは、セーフティモジュールによって送信された完全な安全PDU入力を取得するために、アプリケーションから発行できます。 Anybus CompactCom 40は完全な安全PDUで応答します。アプリケーションはこれを解釈する必要があります。

- コマンド詳細

  (データなし)

- 応答詳細

フィールド	内容
CmdExt[0]	(未使用)
CmdExt[1]	
Data[0～n]	セーフティモジュールからの安全PDU

オブジェクト固有のエラーコード

エラー コード	説明	コメント
01h	セーフティモジュールはメッセージを拒否しました。	セーフティモジュールによって送られたエラーコードはMsgData[2]とMsgData[3]の中にあります。
02h	セーフティモジュールからのメッセージ応答のフォーマットが不正です（例、不正な長さ）。	-

13 ホストアプリケーションオブジェクト

13.1 概要

このグループのオブジェクトは、ホストアプリケーションソフトウェア内で実装されることを想定しています。 Anybusモジュールは、これらのオブジェクトにコマンドを発行し、オブジェクト内の設定やデータにアクセスします。 本章では、オブジェクトをいつ、どのように使用するかという観点から、オブジェクトの機能を分類しています。

下記も参照してください。

- *メッセージの分割, ページ 47*
- *Anybus モジュールオブジェクト, ページ 61*
- *機能の分類, ページ 137*

各オブジェクトの詳細情報については、下記を参照してください。

- *アプリケーションオブジェクト (FFh) , ページ 114*
- *アプリケーションデータオブジェクト (FEh) , ページ 105*
- *Energyコントロールオブジェクト (F0h) , ページ 130*
- *SYNC オブジェクト (EEh), ページ 129*
- *モジュラーデバイスオブジェクト (ECh) , ページ 127*
- *アセンブリマッピングオブジェクト (EBh) , ページ 124*
- *アプリケーション・ファイルシステムインターフェース・オブジェクト (EAh), ページ 122*
- *ファンクショナルセーフティオブジェクト (E8h), ページ 103*
- *Energyレポーティングオブジェクト (E7h) , ページ 102*

13.2 実装ガイドライン

通常、オブジェクトの実装では、受信コマンドの構文解析と適切な応答の作成が問題になります。 これらをどのように実現するかの詳細は本ドキュメントの範囲外ですが、以下の基本ルールに従うようにしてください。

- 実装されるオブジェクトは、本ドキュメントやネットワークインターフェースのAppendixで指定されているすべてのオブジェクトアトリビュート（インスタンス#0）をサポートしなければなりません。
- 何らかの理由によりコマンドを実行できない場合（すなわち、実装されていないオブジェクトやアトリビュート、コマンドがある場合）、問題の原因を示すため、適切なエラーコードで応答してください。
- アプリケーションオブジェクトとアプリケーションデータオブジェクトは必ずサポートしなければなりません。
- ネットワーク固有オブジェクトのサポートは任意ではありますが、実装することが推奨されます。 ただし、ネットワークの機能は、Anybusモジュールにより提供される標準機能では、あらかじめ定義された一部のデバイス情報やサービスの使用に制限されることに注意してください。 この制限は多かれ少なかれ重要なものであり、各ネットワークインターフェースのAppendixに記載されています。 標準機能では不十分な場合、すなわち、ベンダー固有の情報やネットワークの拡張機能が必要な場合、ネットワーク固有オブジェクトをホストアプリケーションに実装することが必要になるかもしれません。
- モジュールは、起動時にネットワーク固有オブジェクトのアトリビュートの値を取得しようと試みます。 実装されていないオブジェクトにモジュールがアクセスしようとした場合は、エラーメッセージ（03h、Unsupported Object）を返してください。 ホストアプリケーションにアトリビュートが実装されていない場合、エラーメッセージ（06h、「Invalid CmdExt[0]」）を返します。 このとき、モジュールはデフォルト値を使用します。 また、本モジュールがネットワークのAppendixに示されていないアトリビュートの値を取得しようとした場合、エラーメッセージ（06h、「Invalid CmdExt [0]」）を返します。
- プロセスデータのリマッピング（コマンドはRemap_ADI_Write_AreaおよびRemap_ADI_Read_Area）のサポートはAnybus CompactCom 40ではオプションとなっていますが、ネットワークによってはより優れたネットワーク機能を提供できます。

下記も参照してください。

- エラーコード, ページ 52

> **!** オブジェクトリビジョンアトリビュートの目的は、Anybusモジュールが、ホストアプリケーションに実装されているオブジェクトとAnybusモジュールにおけるオブジェクトとの間に互換性があるか判断できるようにし、必要に応じて別の実装を使用できるようにすることです。 そのため、オブジェクトリビジョンアトリビュートに実際の実装が反映されるようにし、本ドキュメントやネットワークガイドの内容にのみ基づきインクリメントする必要があります。

ご不明な点がございましたら、www.anybus.com/supportよりHMS Industrial Networks ABの技術サポートサービスにお問い合わせください。

13.3 Energyレポーティングオブジェクト（E7h）

カテゴリ

拡張

オブジェクトの説明

ホストアプリケーションはこのオブジェクトを使用して、消費または生成された電力を報告する方法を標準化します。このオブジェクトのレポーティング機能は限定的なものです。 より詳細なレポーティング機能を提供するネットワークでは、透過性のある方法でホストアプリケーション側に実装する必要があります。

サポートコマンド

オブジェクト：	Get_Attribute
インスタンス：	Get_Attribute

オブジェクトアトリビュート（インスタンス#0）

#	名前	アクセス	データ型	値
1	Name	Get	CHAR配列	“Energy Reporting”
2	Revision	Get	UINT8	01h
3	Number of instances	Get	UINT16	0001h
4	Highest instance no.	Get	UINT16	0001h

インスタンスアトリビュート（インスタンス#1）

拡張

#	名前	アクセス	種類	説明
1	Energy Reading	Get	構造体： UINT32 UINT32	消費または生成された電力の量（単位：Wh）。 不揮発性メモリに格納される。最初のUINT32はEnergy Readingの下位部分を表し、2番目のUINT32はEnergy Readingの上位部分を表す。
2	Direction	Get	BOOL	ホストが電力を消費しているか生成しているかを示す。 値 意味 0： 生成 1: 消費
3	Accuracy	Get	UINT16	精度: 読み取りの0.01% 0： 不明 現在の消費電力（単位：定格消費電力の0.01%） 定格消費電力（単位：mW）
4	Current Power Consumption	Get	UINT16	現在の消費電力（単位：定格消費電力の0.01%）
5	Nominal Current Consumption	Get	UINT32	定格消費電力（単位：mW）

13.4 ファンクショナルセーフティオブジェクト (E8h)

カテゴリ

拡張

オブジェクトの説明

> ! このオブジェクトはセーフティモジュールが未使用のときは実装しないでください。

このオブジェクトはアプリケーションのセーフティ設定を指定します。 ファンクショナルセーフティのサポートが必要であり、セーフティモジュールをAnybus CompactCom モジュールと接続する場合は必須です。

サポートコマンド

オブジェクト：	Get_Attribute
インスタンス：	Get_Attribute

オブジェクトアトリビュート (インスタンス**#0**)

#	名前	アクセス	データ型	値
1	Name	Get	CHAR配列	“Functional Safety”
2	Revision	Get	UINT8	01h
3	Number of instances	Get	UINT16	0001h
4	Highest instance no.	Get	UINT16	0001h

インスタンスアトリビュート (インスタンス**#1**)

#	名前	アクセス	データ型	デフォルト値	コメント
1	Safety enabled	Get	BOOL	-	TRUEのときセーフティモジュールと通信が可能です。 注：ファンクショナルセーフティがサポートされていない場合、このアトリビュートはFALSEに設定されなければなりません。
2	Baud Rate	Get	UINT32	1020 kbit/s	このアトリビュートはAnybus CompactComとセーフティモジュール間の通信ボーレート (bits/s) を設定します。 有効値 : • 625 kbit/s • 1000 kbit/s • 1020 kbit/s (デフォルト) このアトリビュートに他の値を設定すると、モジュールがEXCEPTIONステートになる原因になります。 このアトリビュートはオプションです。 実装されない場合、デフォルト値が使用されます。 注：IXXAT Safe T100を使用する場合、ホストアプリケーションはこのアトリビュートを実装してはいけません。
3	(予約)				

#	名前	アクセス	データ型	デフォルト値	コメント
4	Cycle Time	Get	UINT8	-	Anybusとセーフティモジュール間の通信周期時間（ミリ秒）。 注：IXXAT Safe T100を使用する場合、ホストアプリケーションはこのアトリビュートを実装してはいけません。 有効値： • 2 ms • 4 ms • 8 ms • 16 ms 他の値がこのアトリビュートに設定されると、Anybusは Exceptionステートに入ります。 このアトリビュートはオプションです。実装されない場合、選択されたボーレートに対応した最小周期時間が適用されます： • 1020 kbit/sの場合は 2 ms • 1000 kbit/sの場合は 2 ms • 625 kbit/sの場合は 4 ms Anybus CompactComは上記の最小値に従い周期時間の妥当性を確認します。 例えばボーレートが625 kbit/sで周期時間が2 msに設定されていると、Anybus CompactComは EXCEPTION（例外）状態になります。
5	FW upgrade in progress	Set	BOOL	False	接続されているセーフティモジュールのファームウェアを Anybus CompactComがアップグレードしているかどうかを示します。 これは、Anybus CompactComが通常よりも長く NW_INIT状態にとどまることを意味します。

13.5 アプリケーションデータオブジェクト（FEh）

カテゴリ

基本。 このオブジェクトは必須。

オブジェクトの説明

このオブジェクト内の各インスタンス（アプリケーションデータインスタンス、またはADI）は、ネットワーク上で表されるデータブロックと関連しています。 そのようなデータがネットワークからアクセスされるたびに、モジュールは、その要求をこのオブジェクト（またはこのオブジェクト内のインスタンス）に対するオブジェクト要求に変換します。 また、モジュールは、必要に応じてこのオブジェクトに自発的にアクセスします。 ネットワークにおける実際の表現は、ネットワークによって大きく異なります。例えばDeviceNetでは、ADIは専用のCIPオブジェクトとして表されるのに対し、PROFIBUSでは、ADIは非周期DP-V1リード/ライトサービスによりアクセスされます。

アプリケーションデータオブジェクトのインスタンスは、次のような異なるクラスのデータをモデリングするのに使用されます（変数、配列、または構造体）。 各クラスは、以下の表に示すようにインスタンスアトリビュートにより分類されます。

クラス	分類方法	注意事項
Variable（変数）	Number of elementsが1。	アプリケーションデータオブジェクトのリビジョン3以降、文字列変数（ここで言う文字列変数とは、複数のサブエレメントを持つ1つのエレメントで構成されるもの）を作成する場合は、CHAR型変数と一緒にNumber of subelementsアトリビュートを使用することを推奨します。
Array（配列）	Number of elements > 1、かつ、Data typeにおけるデータ型の定義数が1。	Number of subelementsアトリビュートは、配列に対して有効ではありません。 CHAR型配列は、ネットワークでサポートされる文字列の文字列変数に変換されます。 文字列変数を表すための推奨方法は、Variable（変数）に関する上記の注意事項を参照ください。
Structure（構造体）	Number of elements > 1、かつ、Data typeにおけるデータ型の定義数と等しい。	構造体は、データ型が異なる要素で構成されます。 オブジェクトリビジョン3以降で使用可能です。

ADIのインスタンス番号にかかわらず、ネットワークやAnybusモジュールがホストアプリケーション内のADIを効率よくスキャンできるように、このオブジェクトにはGet_Instance_Number_By_Orderコマンドが追加されています。 このコマンドは、あたかもADIが番号付きリストに格納されているかのように、インスタンス番号を取得します。これにより、Anybusモジュールは、ホストアプリケーションに実際に実装されているインスタンスに対してのみ問い合わせを行うことが可能です。 オーダー番号は、ADIをプロセスデータにマッピングする際にも使用されます。コマンドMap_ADI_Write_AreaおよびMap_ADI_Write_Ext_Areaについては、ネットワーク オブジェクト *(03h)*, ページ *73*の説明を参照してください。

下の例では、ホストアプリケーションはインスタンス番号1、3、100を持つ3つのADIを持っています。

インスタンス番号	実装状況	オーダー番号
1	あり	1
2	なし	-
3	あり	2
4 ~ 99	なし	-
100	あり	3

このケースでは、ホストアプリケーションは、オーダー番号3へのGet_Instance_Number_By_Order要求に対してインスタンス番号100で応答します。

アプリケーションを設計する際には、以下の点を考慮に入れてください。

- Anybusモジュールは、要求内容が明らかに間違っている場合でも、ホストアプリケーションが実行するパラメータ要求に対するエラーチェックは実行しません（例えば、ADIに対するゼロバイトデータの書き込み要求や、存在しないアトリビュートへのアクセスは、モジュールによってフィルターされません）。
- ネットワークによっては、ある一定のタイミング要求が課せられる場合があるため、ホストアプリケーションにおける応答時間（すなわち、このオブジェクトに送信された要求を処理して応答するまでの時間）を考慮する必要があります。 必要に応じて、特殊なタイミング要件などが各ネットワークのAppendixで指定されています。
- プロセスデータのリマッピングをサポートしようとしている場合、オブジェクトコマンドRemap_ADI_Write_Area、Remap_ADI_Read_Area、Get_Instance_numbersの実装が必須です。
- プロセスデータのリマッピングがサポートされている場合は、オブジェクトアトリビュート#11および#12が必須です。
- アプリケーションデータオブジェクト内でデータクラスが配列または構造体となっているすべてのアトリビュートに対して、コマンドGet_Indexed_AttributeおよびSet_Indexed_Attributeを実装することをお勧めします。

サポートコマンド

オブジェクト：	Get_Attribute（01h）
	Get_Instance_Number_By_Order (10h)
	Remap_ADI_Write_Area (13h)
	Remap_ADI_Read_Area (14h)
	Get_Instance_Numbers (15h)
インスタンス：	Get_Attribute（01h）
	Set_Attribute (02h)
	Get_Enum_String (06h)
	Get_Indexed_Attribute (07h)
	Set_Indexed_Attribute (08h)

オブジェクトアトリビュート（インスタンス**#0**）

#	名前	アクセス	データ型	値
1	Name	Get	CHAR配列	“Application Data”
2	Revision	Get	UINT8	03h
3	Number of instances	Get	UINT16	（アプリケーションによって異なる）
4	Highest instance no.	Get	UINT16	
11	No. of read process data mappable instances	Get	UINT16	
12	No. of write process data mappable instances	Get	UINT16	

プロセスデータのリマッピングをサポートするアプリケーションでは、アトリビュート#11および#12は必須です。

インスタンスアトリビュート（インスタンス **#1 ~ n**）

#	名前	アクセス	種類	説明
1	Name	Get	CHAR配列	ADI名（多言語対応）
2	Data type	Get	UINT8の配列	各UNIT8は、構造体や変数のインスタンス値に該当する要素のデータ型を定義します。 配列の場合、ある1つのUNIT8が、該当する配列要素のすべてのサブエレメントのデータ型を定義します。
3	Number of elements	Get	UINT8	アトリビュート#5に含まれる要素数を表します。 Number of elementsがゼロに設定されているADIは、ネットワークによっては受け付けられないことがあるため、そのようなADIは使用しないことを強く推奨します。
4	Descriptor	Get	UINT8の配列	各UNIT8はビットフィールドで、構造体や変数のインスタンス値に該当する要素のアクセス権などを指定します。 配列の場合、1つのUINT8で配列要素全てのサブエレメントに対応する定義を指定できます。 プロセスデータのリマッピングがサポートされている場合、b3およびb4は必須です。 ビット　アクセス権 プロセスデータのリマッピングがサポートされている場合、b3およびb4は必須です。 b0: 1: Getアクセス b1: 1: Setアクセス b2: -: (予約、ゼロに設定) b3: 1: ライトプロセスデータとしてマッピング可能 b4: 1: リードプロセスデータとしてマッピング可能
5	Value(s)	アトリビュート#4によって決定	アトリビュート #2によって決定	ADIの値 インデックス付きの要素は、タイプとサイズがアトリビュート#2で指定されたものと異なる場合があります。 このアトリビュートには、ビット境界で配置されたすべての要素が含まれます。暗黙のパディングは使用しないでください。 アライメントに関する具体的な制約や明示的なパディングについては、以下の表を参照ください。
6	Max. value	Get	アトリビュート #2によって決定	ADIの許容最大値。 このアトリビュートの実装は任意です。 実装しない場合、モジュールは指定されたデータ型の最大値をこのアトリビュートの値として使用します。
7	Min. value	Get	アトリビュート #2によって決定	ADIの許容最小値。 このアトリビュートの実装は任意です。 実装しない場合、モジュールは指定されたデータ型の最小値をこのアトリビュートの値として使用します。
8	Default value	Get	アトリビュート #2によって決定	ADIのデフォルト値。 このアトリビュートの実装は任意です。 実装されていない場合、ゼロ値（floatでは +Min.値）が使用されます。
9	Number of subelements	Get	UINT16の配列	各UNIT16は、構造体や変数のインスタンス値に該当するサブエレメントの数を定義します。 このアトリビュートの実装は任意です。また、配列には実装しないでください。 該当する要素がCHAR型またはOCTET型の場合のみ、サブエレメントの数が1とは異なります。 このアトリビュートを実装しない場合、各要素に1つのサブエレメントが存在すると想定されます。
10	Element name	Get	文字列の構造体（CHAR の配列、NULL バイトで区切られます）	このアトリビュートは、構造体クラスのADI内にある各要素の名前の読み取りを有効にするために使用されます。 各文字列は、NULL バイトで区切られます。 最終文字列の末尾にはNULLバイトはありません。 このアトリビュートは、ネットワークで使用される要素名を反映します。 構造体の要素数は、アトリビュート#3（Number of elements）の値と等しくなければなりません。 このアトリビュートに対して可能なコマンドはGet_Attribute（応答には文字列が含まれ、NULLバイトで区切られる）とGet_Indexed_Attribute（NULLバイトなしで文字列が返される）です。 応答全体がメッセージデータフィールドに収まる必要があります。 受け付け可能な応答の最大サイズは、使用しているチャネルに応じて255または1524バイトです。

- アトリビュート#5 ~ 8のバイトオーダーは、ネットワークによって異なります。Anybusはバイト入れ替えを行いません。
- データクラスが構造体でない限り、Max/Min/Defaultアトリビュートは、ADIのすべての要素で共通です。 すなわち、配列の各要素に対するMax/Min/Default値は個別には設定されません。 構造化されているADIの場合、フォーマットはアトリビュート#5と同じです。
- インスタンスの値は、メッセージデータフィールド全体にフィットする必要があります。 そのため、使用しているチャネルに応じて、全要素の合計バイトサイズは255または1524バイトを超えてはなりません。
- 稼働中に変更される可能性のあるアトリビュートは、アトリビュート#1と#5のみです。定義が完了したら、その他すべてのアトリビュートは固定されているとみなす必要があります。稼働中はそれらの変更は許可されていません。

パラメータアクセスに関する注意事項

次のリストは、Anybus CompactCom 40のパラメータへのアクセスに関する規則と注意事項を示しています。

- 1つ以外のサブエレメントは、CHARおよびOCTETにのみ使用可能です
- 構造体にENUM型の要素を含めることはできません
- 構造化されたADIにいわゆる穴がある場合、そのデータ型はPAD0として定義されます。 Descriptor（アトリビュート#4）は設定可能とも取得可能とも定義されてはならず、コマンドGet_Indexed_Attribute/Set_Indexed_Attributeに対しては「Invalid CmdExt[1]」（07h）が返されます。
- 必要に応じて、Anybus CompactComによって「ADIName.0」などの要素名が生成されます。
- データ型BIT1～BIT7およびPAD0～PAD16の要素を除き、すべての要素でバイト境界に配置されている必要があります。
- パラメータアクセスに対してのみ行われる暗黙のパディングは、メッセージが常に完全なバイト列であるため、最後にアクセスした要素からバイト境界までとなります。
- 明示的なパディングは、PADxデータ型の要素を使用して行われます。
- バイト境界に配置されていない要素は、Get_Indexed_attributeを介してアクセスされるとバイト境界に配置するようにシフトダウンし、Set_Indexed_Attributeを介してアクセスされるとその逆になります。
- Descriptorは、同じADIの要素間で異なる場合があります
  - Descriptorビット「Get アクセス」の設定が複数の要素間で一貫していない構造体クラスのADIのGet_Attributeに対しては、アプリケーションは読み取り不可能な要素にゼロを埋め込んで応答する必要があります。「Get アクセス」Descriptorビットが一貫して0に設定されている場合、Get_Attributeに返されるエラーコードは「Attribute not gettable (09h)」です。
  - Descriptorビット「Set アクセス」の設定が複数の要素間で一貫していない構造体クラスのADIのSet_Attributeに対しては、アプリケーションは設定不可能な要素を無視して設定可能な要素の値を適用します。「Set アクセス」Descriptorビットが一貫して0に設定されている場合、Set_Attributeに返されるエラーコードは「Attribute not settable」（08h）です。
- 少なくとも1つの要素が、オプションの最小/最大アトリビュートと比較したうえで範囲外である構造体クラスのADIへのSet_Attributeに対しては、アプリケーションはすべての要素を無視して 「Out of range」と応答します。「Out of range」は一部の要素では小さすぎたり大きすぎたりした場合に競合が発生しないようにするためのもので、範囲外の要素が1つあることが分かった時点でチェックを停止できるという利点もあります。
- ビット境界で配置されている要素は、Get_Indexed_Attribute/Set_Indexed_Attributeを介してアクセスされると、常にビット0にシフトダウンされます。

コマンドの詳細: Get_Instance_Number_By_Order

詳細

コマンドコード:	10h
有効な対象	オブジェクト

説明

このコマンドは、オーダーリストでソートされているかのように、ADIの実際のインスタンス番号を要求します。

- コマンド詳細：

フィールド	内容
CmdExt[0]	要求されたオーダー番号（下位バイト）
CmdExt[1]	要求されたオーダー番号（上位バイト）

- 応答詳細（成功）：

フィールド	内容
MsgData[0～1]	提示されたオーダー番号に対応するADIのインスタンス番号。

- 応答詳細（エラー）：

エラー	内容
Invalid CmdExt[0]	要求されたオーダー番号がADIと関連付けられていない

コマンドの詳細: Remap_ADI_Write_Area

詳細

コマンドコード:	13h
有効な対象	オブジェクト

説明

Anybusモジュールは、ネットワークからプロセスデータマップの変更を要求されるたびにこのコマンドを発行します。ADIは、コマンドで指定されたのと同じ順序で挿入位置にマッピングされます。 このコマンドを使用すると、CmdExt [0]のマッピングアイテム番号で指定された場所から、複数のマッピングアイテムを削除/挿入できます。ここでいうマッピングアイテムとは、Map_ADI_Write_Areaコマンドでマッピングされたadi、または、Remap_ADI_Write_Areaコマンドでマッピングされたadi (またはマルチエレメントadiの要素) のことです。

挿入された各マッピングアイテムに対するコマンドデータには、以下のデータセットが含まれます。

- adi番号
- マッピングする最初の要素へのインデックス
- 後ろに続くマッピングする要素の数

このコマンドは、Anybus CompactComが次の状態にあるときに発行されます： NW_INIT、WAIT_PROCESS、IDLE、ERROR。

このコマンドで指定されたアクションは、すべて実行されるか、すべて拒否されるかのいずれかです。すなわち、コマンドが受け付けられなかった場合、プロセスデータマップは変更されてはなりません。

Anybusモジュールは、未実行のマッピングコマンドを1度に1つしか実行できません。

下記も参照してください。

- ネットワーク オブジェクト *(03h), ページ 73*
- 稼働中のプロセスデータのリマッピング, *ページ 149*

> **!** この処理をサポートするには、アプリケーションが実行時にプロセスデータをリマッピングできなければなりません。 これは、オブジェクトリビジョン2では必須で、オブジェクトリビジョン3では任意です。このコマンドをサポートすることを強く推奨します。

- コマンド詳細：

フィールド	内容
CmdExt[0]	リマップ開始（下位バイト）（マッピングアイテム番号、0 =先頭）
CmdExt[1]	リマップ開始（上位バイト）（マッピングアイテム番号、0 =先頭）
Data[0-1]	削除するマッピングアイテムの数
Data[2-3]	挿入するマッピングアイテムの数（0 ~ 62）
Data[4-5]	新規マッピングアイテム1： ADI番号
Data[6]	新規マッピングアイテム1： マッピングする最初の要素へのインデックス
Data[7]	新規マッピングアイテム1： 後ろに続くマッピングする要素の数。
Data[8-9]	新規マッピングアイテム2： ADI番号
Data[10]	新規マッピングアイテム2： マッピングする最初の要素へのインデックス
Data[11]	新規マッピングアイテム2： 後ろに続くマッピングする要素の数。
...	（など）

- 応答詳細（成功）：

フィールド	内容
MsgData[0]	結果として得られるライトプロセスデータのバイト単位のトータルサイズ（下位バイト）。
MsgData[1]	結果として得られるライトプロセスデータのバイト単位のトータルサイズ（上位バイト）。

- 応答詳細（エラー）：

エラー コード	エラー	意味
01h	Mapping item error	1つ以上のマッピングアイテムに対してNAKが発行されたため、要求されたマッピングが拒否された。
02h	Invalid total size	結果として得られるトータルデータサイズがアプリケーションで許容される最大値を超えたため、要求されたマッピングが拒否された。

コマンドの詳細: Remap_ADI_Read_Area

詳細

コマンドコード:	14h
有効な対象	オブジェクト

説明

このコマンドは、ADIをリードプロセスデータ領域に（リ）マッピングするのに使用します。 それ以外の点についてはRemap_ADI_Write_Areaと同じです。

リマッピングコマンドに対して正しくACKが送信された場合、プロセスデータマップが変更される場所を表します。 シリアルアプリケーションの場合、これはACK送信後に、変更されたプロセスデータマップが空のテレグラム（再送の場合はテレグラム）に続くテレグラムにて期待/使用されることを意味します（稼働中のプロセスデータのリマッピング, ページ 149を参照）。

- ネットワーク オブジェクト (03h), ページ 73
- 稼働中のプロセスデータのリマッピング, ページ 149

> この処理をサポートするには、アプリケーションが実行時にプロセスデータをリマッピングできなければなりません。 このコマンドのサポートは任意ですが、サポートすることを強く推奨します。

コマンドの詳細: Get_Instance_Numbers

詳細

コマンドコード:	15h
有効な対象	オブジェクト

説明

このコマンドは、特定のプロパティを持つADIのリストを作成するのに使用します。 リストタイプ01h～03hは、プロセスデータの動的マッピングをサポートするすべてのアプリケーションで必須です。 リストタイプの一覧については、以下の「リストタイプ一覧表」を参照してください。

アプリケーションは、要求された数と等しい数、またはそれより少ない数のインスタンス（要求された数より少ないインスタンスしかアプリケーションに存在しない場合）で応答します。 要求された開始オーダー番号が、最大のインスタンス番号より大きい場合、空の応答が返されます。 サポートされていないリストタイプが要求された場合、エラー応答「Invalid CmdExt[1]」が生成されます。

Anybus CompactComモジュールは、完全なリストを取得するため、昇順のオーダー番号を持つ複数のコマンドを発行する場合があります。

- コマンド詳細：

フィールド	内容
CmdExt[0]	予約= 00h
CmdExt[1]	リストタイプ（後述の「リストタイプ一覧」を参照）
Data[0-1]	開始オーダー番号
Data[2-3]	要求されたインスタンスの数

- 応答詳細：

フィールド	種類	内容
MsgData[0-1]	UINT16	選択したリストタイプに一致する、ADIのインスタンス番号（オーダー番号に対応するインスタンス番号）。
MsgData[2-3]	UINT16	選択したリストタイプに一致する、ADIのインスタンス番号（オーダー番号+1に対応するインスタンス番号）。
MsgData[4-5]	UINT16	選択したリストタイプに一致する、ADIのインスタンス番号（オーダー番号+ 2に対応するインスタンス番号）。
...	...	...

リストタイプ一覧

リスト番号	リストタイプ
00h	予約
01h	すべてのADI
02h	リードプロセスデータにマッピング可能なすべてのADI（Descriptorアトリビュートのビット4が 1 に設定されているすべてのADI。 詳細については113ページの「インスタンスアトリビュート（インスタンス#1～n）」を参照してください。）
03h	ライトプロセスデータにマッピング可能なすべてのADI（Descriptorアトリビュートのビット3が 1 に設定されているすべてのADI。 詳細については113ページの「インスタンスアトリビュート（インスタンス#1～n）」を参照してください。）

13.6 アプリケーションオブジェクト（FFh）

カテゴリ

基本、拡張

オブジェクトの説明

このオブジェクトは必須です。ホストアプリケーションの全般的な設定がグループ化されています。 このオブジェクトとコマンドを使用すると、多言語のサポートやネットワークリセットの要求、診断イベントのラッチが可能になります。

現在のパラメータ設定を表すcontrol sum（アプリケーションにて設定可能）は、起動時間を向上させるために使用できます。

利用可能な候補ファームウェアがあるか否か、ハードウェアスイッチを介したモジュールのアドレス設定が可能であるか否かなどの情報も、このオブジェクトから読み取られます。

サポートコマンド

オブジェクト：	Get_Attribute（01h）
	Reset (05h)
	Reset_Request (10h)
	Change_Language_Request (11h)
	Reset Diagnostic (12h)
インスタンス：	Get_Attribute（01h）
	Set_Attribute (02h)
	Get_Enum_String (06h)

オブジェクトアトリビュート（インスタンス#0）

#	名前	アクセス	データ型	値
1	Name	Get	CHAR配列	“Application”
2	Revision	Get	UINT8	02h
3	Number of instances	Get	UINT16	0001h
4	Highest instance no.	Get	UINT16	0001h

インスタンスアトリビュート（インスタンス#1）

#	名前	アクセス	種類	説明
1	Configured	Get	BOOL	アプリケーションパラメータが出荷時の値から変更されたかどうかを表します。 値 文字列 False: 出荷時状態。 True: 設定済み。設定が変更されている。 コマンド「Reset」および「Reset_Request」については、以下の詳細を参照してください
2	Supported languages	Get	ENUM配列	ホストアプリケーションによってサポートされている言語を指定するリストです。 値 意味 00h: “English”. 01h: “Deutsch”. 02h: “Español”. 03h: “Italiano”. 04h: “Français”. 下記も参照してください。 • *Anybusオブジェクト (01h), ページ 62*、インスタンス#1、アトリビュート#9 • 下記のコマンドChange_Language_requestに関する詳細。
3	Serial number	Get	UINT32	ベンダーが設定したデバイスのシリアル番号。 該当するネットワーク固有のホストオブジェクトからシリアル番号を取得できない場合、モジュールは代わりにこの番号を使用し、ネットワークの要件に応じてその値を変換します。 ネットワーク固有のアトリビュートもこのアトリビュートも実装されていない場合、モジュールは自身が持つシリアル番号を使用します。*Anybusオブジェクト (01h), ページ 62*、アトリビュート#3を参照してください。
4	Parameter control sum	Get	UINT8の配列（128ビット）	このアトリビュートは、アプリケーションにおける現在のパラメータ設定を表すcontrol sumを保持します。 アプリケーションがcontrol sumをどのように計算するかは指定されておらず、唯一の要件は、アプリケーションのパラメータを変更するとcontrol sumも即座に変更することとなっています。 control sumは、POWERLINKおよびPROFINETでAnybus CompactComの起動時間を短縮するために使用されます。 これらのネットワークに共通するのは、マスターがスレーブに転送するパラメータ設定を指定するためのパラメータ（POWERLINKにおける設定データ・設定時間 ならびにPROFINETにおけるUUID）を、マスターが設定することです。 これらのパラメータ設定をアプリケーションが既に持っているか否かをスレーブが判断するためには、マスターから受け取ったパラメータとアプリケーションから受け取ったパラメータを比較します。 保存されている（不揮発性メモリ内）マスターパラメータとアプリケーションパラメータが、アプリケーションとマスターから受け取った新しいパラメータと一致すれば、アプリケーションの再パラメータ化は不要となります。 このアトリビュートの実装はアプリケーションに必要なわけではありませんが、上記のネットワークにて高速起動/Fast Startupを使用する場合は実装を推奨します。

#	名前	アクセス	種類	説明
5	Candidate firmware available	Get/Set	BOOL	次回の再起動時に、ファームウェアのアップグレードに利用可能なファームウェアファイルが候補領域にあるか否かを示します。 アプリケーションはこれを使用して、ファームウェアのアップグレードによって次回の再起動が長引くかどうかを判断できます。 本アトリビュートは起動時にクリアされます。 値 意味 False: 利用可能なファームウェアファイルが候補領域に無い。 True: 利用可能なファームウェアファイルが候補領域に有る。 下記も参照してください。 • ファームウェアダウンロード, ページ 24 • 起動手順, ページ 57
6	Hardware configurable address	Get	BOOL	ハードウェアスイッチを使用してモジュールのアドレスを設定できるかどうかを示します。 アドレスはハードウェアで設定可能かもしれませんが、必ずしもハードウェアで設定される必要はありません。 EtherNet/IPなど一部のネットワークでは、この区別ができることが必要です。 値 意味 False: アドレスはハードウェアで設定できない。 True: アドレスはハードウェアで設定できる。
7	Mode	Get	BITS32	モジュールのLED表示を変更します。 値 意味 00h: 通常のLED表示モード (デフォルト) 01h: AIDA LED表示モード。 Anybus CompactCom 40 PROFINETにのみ使用されます。詳細についてはネットワークガイドを参照してください。 その他： (予約)
8	Vendor name	Get	CHAR配列	ベンダー名。 該当するネットワーク固有ホストオブジェクト内にあるベンダー名を取得できない場合、ネットワークの要件に従って変換されたこの名前を、デバイスは代わりに使用します。 ネットワーク固有のアトリビュートもこのアトリビュートも実装されていない場合、デバイスは “HMS Industrial Network” を使用します。 このアトリビュートの実装は任意です。
9	製品名	Get	CHAR配列	デバイスに付けられたベンダーの製品名。 該当するネットワーク固有ホストオブジェクト内にある製品名を取得できない場合、ネットワークの要件に従って変換されたこの名前を、デバイスは代わりに使用します。 ネットワーク固有のアトリビュートもこのアトリビュートも実装されていない場合、デバイスはAnybusの製品名を使用します。 このアトリビュートの実装は任意です。
10	Firmware version	Get	構造体 UINT8 メジャー UINT8 マイナー UINT8 ビルド	デバイスに付けられたベンダーのファームウェアバージョン。 フォーマット： バージョン「2.18.3」は以下のように表されます。 最初のバイト = 0x02、2番目のバイト = 0x12、3番目のバイト = 0x03。 該当するネットワーク固有ホストオブジェクト内にあるファームウェアバージョンを取得できない場合、ネットワークの要件に従って変換されたこのバージョンを、デバイスは代わりに使用します。 ネットワーク固有のアトリビュートもこのアトリビュートも実装されていない場合、デバイスはAnybusファームウェアバージョンを使用します。5.6.1.3 Anybusオブジェクトインスタンス、アトリビュート2を参照してください。 このアトリビュートの実装は任意です。
11	Hardware Version	Get	UINT16	デバイスに付けられたベンダーのハードウェアバージョン。 該当するネットワーク固有ホストオブジェクト内にあるハードウェアバージョンを取得できない場合、ネットワークの要件に従って変換されたこのバージョンを、デバイスは代わりに使用します。 ネットワーク固有のアトリビュートもこのアトリビュートも実装されていない場合、デバイスはAnybusのハードウェア機能IDを使用します。 このアトリビュートの実装は任意です。

コマンドの詳細: Reset

詳細

コマンドコード: 05h

有効な対象： オブジェクト

説明

このコマンドは、リセットが必要なときにモジュールにより発行されます。 ネットワークの種類によっては、このコマンドの前に「Reset_Request」コマンドが先に発行されることがあります。

スタンドアロンのシフトレジスタモードでは、使用可能なアプリケーションがないことから、ネットワークからのリセット要求時にリセットがモジュールによって自動的に処理されます。

- コマンド詳細：

フィールド	内容	コメント
CmdExt[0]	(予約、無視)	-
CmdExt[1]	00h: パワーオンリセット	これはデバイスリセットとみなされます。すなわち、ホストアプリケーションは/RESET信号を用いてモジュールをリセットします。 Anybusモジュールは、この種の要求を発行する前にEXCEPTION状態に移行します。
	01h: 工場出荷時状態へのリセット	ホストアプリケーションをアプリケーション固有の出荷時状態に戻します。 モジュールをこの状態に移行させるために必要なネットワーク固有の処理は、自動的に行われます。 この要求が発行される前のAnybusモジュールの状態は、ネットワークによって異なります。
	02h: パワーオンリセット + 工場出荷時状態へのリセット	上記2つの組み合わせ。 Anybusモジュールは、この種の要求を発行する前にEXCEPTION状態に移行します。

- 応答詳細：

  （データなし）

コマンドの詳細: Reset_Request

詳細

コマンドコード:	10h
有効な対象：	オブジェクト

説明

ネットワークによっては、Resetコマンドの前にこのコマンドが発行されます（下記参照）。 このコマンドは、その名が示すように要求であり、実際のリセットコマンドではありません。

要求するリセットは、パワーオンリセット、工場出荷状態へのリセット、またはその両方が可能です。 パワーオンリセットはデバイスリセットとみなされます。

ホストアプリケーションは、この要求を受け付けた場合、対応するResetコマンドも受信できなければなりません（図を参照）。

またホストアプリケーションは、何らかの理由でリセットを実行できなかった場合にエラーを返すことも可能です。 この場合、モジュールによりResetコマンドは発行されません。

スタンドアロンのシフトレジスタモードでは、ネットワークからのリセット要求はモジュールが自動的に処理して自動的にリセットします。

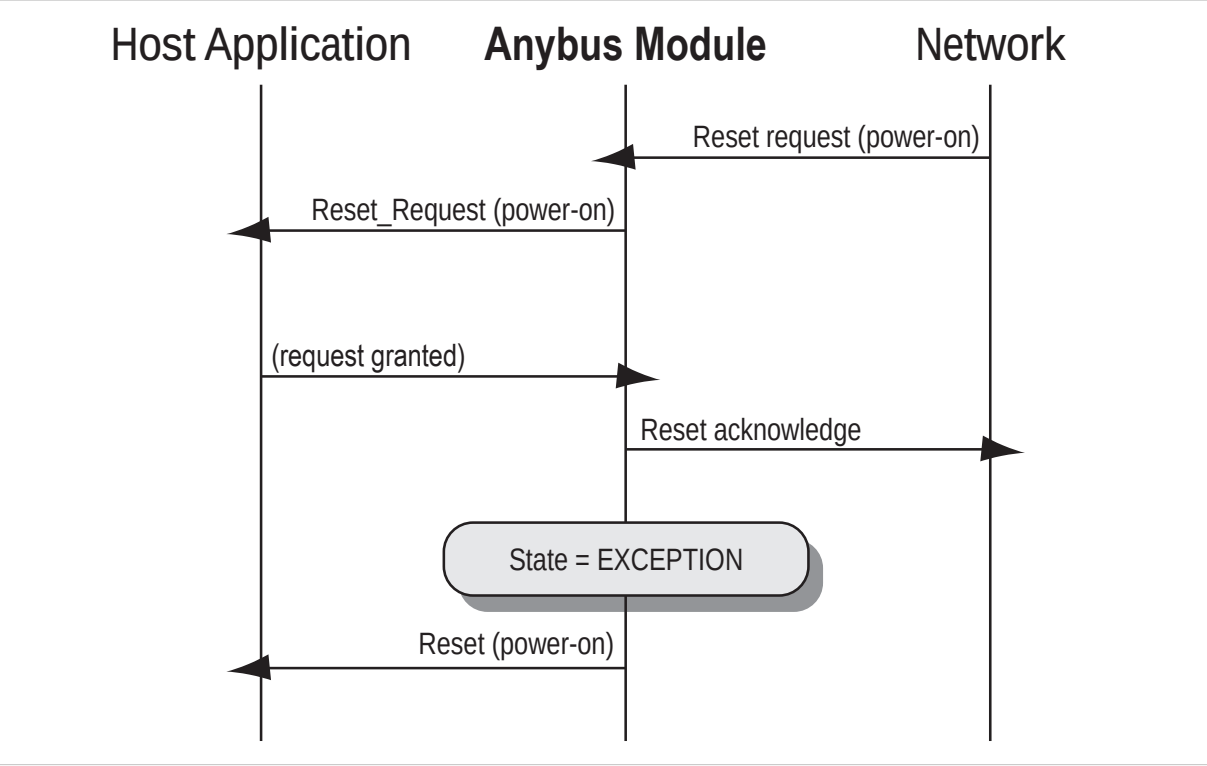

Fig. 21

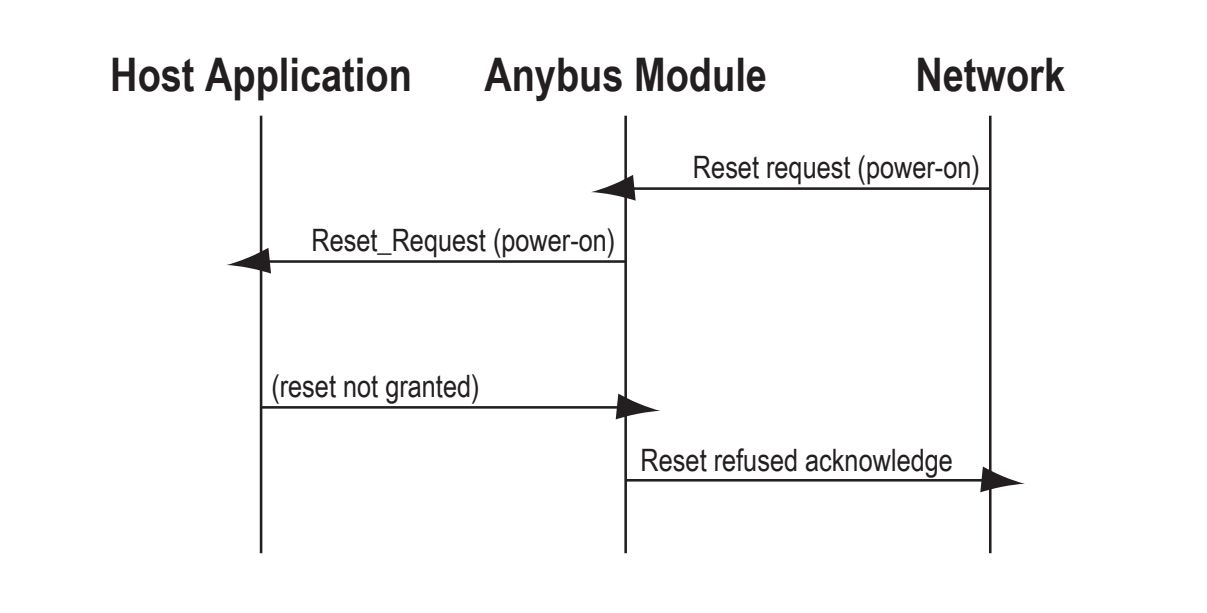

Fig. 22

- コマンド詳細：

フィールド	内容
CmdExt[0]	(予約、無視)
CmdExt[1]	00h: パワーオンリセット
	01h: 工場出荷時状態へのリセット
	02h: パワーオンリセット + 工場出荷時状態へのリセット

- 応答詳細：

  （データなし）

コマンドの詳細: Change_Language_Request

詳細

コマンドコード: 11h

有効な対象： オブジェクト

説明

このコマンドは、ネットワークから現在の言語を変更するよう要求されたときにモジュールにより発行されます。

コマンドが受け付けられると、Anybusオブジェクト（01h）のLanguageアトリビュート（#9）がコマンドに従って変更されます。 またホストアプリケーションは、これに応じて内部の言語設定も変更する必要があります。

- コマンド詳細：

フィールド	内容	
CmdExt[0]	予約。 値= 00h	
CmdExt[1]	要求された言語	
	値	言語:
	00h:	English
	01h:	German
	02h:	Spanish
	03h:	Italian
	04h:	French

- 応答詳細：

（データなし）

コマンドの詳細: Reset_Diagnostic

詳細

コマンドコード:	12h
有効な対象：	オブジェクト

説明

ネットワークのマスターが1つまたは複数のラッチされた診断イベントを了解/リセットしたいとき、アプリケーションオブジェクトにReset_Diagnostic要求が送信されます。

このサービスは、アプリケーションが診断イベントのラッチをサポートする場合にのみ必須です。

診断イベントを削除可能かどうかはアプリケーションが判断します。 アプリケーションは、Reset_Diagnosticの応答において、削除可能な（既にエラーが存在しない）診断インスタンスのリストを提供することが要求されます。 このリストは、Reset_Diagnostic要求におけるリストと同一か、またはそのサブセットとなる可能性があります。 削除可能なインスタンスが存在しない場合、または、Reset_Diagnostic要求を拒否してエラーを返す場合、アプリケーションはサイズがゼロのリストを返すことが可能です。

詳細は診断オブジェクト *(02h), ページ 68*を参照してください。

- コマンド詳細：

フィールド	内容
CmdExt[0]	予約。 値= 00h
CmdExt[1]	予約。 値= 00h
MsgData[0–n]	Anybus CompactComモジュールが削除の許可を要求する診断インスタンスのリスト（UINT16）。

- 応答詳細：

フィールド	内容
CmdExt[0]	予約。 値= 00h
CmdExt[1]	予約。 値= 00h
MsgData[0–n]	Anybus CompactComモジュールが削除することを許された診断インスタンスのリスト（UINT16）。

13.7 アプリケーション・ファイルシステムインターフェース・オブジェクト (EAh)

カテゴリ

拡張

オブジェクトの説明

このオブジェクトはアプリケーションでファイルシステムを作成するために使用され、アプリケーションは使用可能なファイルシステムをAnybus CompactCom内で拡大できます。

オブジェクトは、実行時にファイルシステムインターフェース・インスタンスを動的に作成/削除するのに使用されます。各インスタンスはファイルストリームへのハンドルとなっており、ファイルシステムの操作に関するサービスが用意されています。 オブジェクトは、構造においてAnybusファイルシステムインターフェース・オブジェクト（0Ah）とほぼ同様です。

サポートコマンド

オブジェクト固有コマンドの詳細については、*Anybus ファイルシステムインターフェース・オブジェクト (0Ah), ページ 81*を参照してください。

オブジェクト：	Get_Attribute（01h）
	Set_Attribute (02h)
	Create (03h)
	Delete (04h)
インスタンス：	Get_Attribute（01h）
	File Open (10h)
	File Close (11h)
	File Delete (12h)
	File Copy (13h)
	File Rename (14h)
	File Read (15h)
	File Write (16h)
	Directory Open (20h)
	Directory Close (21h)
	Directory Delete (22h)
	Directory Read (23h)
	Directory Create (24h)
	Directory Change (25h)

オブジェクトアトリビュート（インスタンス#0）

#	名前	アクセス	データ型	値/説明
1	Name	Get	CHAR配列	“Application File System Interface”
2	Revision	Get	UINT8	01h
3	Number of instances	Get	UINT16	-
4	Highest instance no.	Get	UINT16	-
11	Max no. of instances	Get	UINT16	20（推奨）
13	Total disc size	Get	UINT32	ディスクサイズ（単位：バイト）。
14	Free disc size	Get	UINT32	空きディスクサイズ（単位：バイト）。

インスタンスアトリビュート（インスタンス#1 ~ 20

#	名前	アクセス	種類	説明
1	Instance Type	Get	UINT8	値 意味 0： 予約 1: ファイルインスタンス 2: ディレクトリインスタンス
2	File size	Get	UINT32	ファイルサイズ（ディレクトリは0）
3	Path	Get	CHAR配列	インスタンスが処理を行うファイルのパス

ファイルシステムエラー

ファイルシステムインターフェースオブジェクトを呼び出すサービスでエラーが発生した場合、モジュールはFFh（オブジェクト固有のエラー）を返します。 ファイルシステムエラーに関する説明は、エラー応答のデータフィールドに返されます。

#	名前	説明
1	FILE_OPEN_FAILED	ファイルをオープンできない
2	FILE_CLOSE_FAILED	ファイルをクローズできない
3	FILE_DELETE_FAILED	ファイルを削除できない
4	DIRECTORY_OPEN_FAILED	ディレクトリをオープンできない
5	DIRECTORY_CLOSE_FAILED	ディレクトリをクローズできない
6	DIRECTORY_CREATE_FAILED	ディレクトリを作成できない
7	DIRECTORY_DELETE_FAILED	ディレクトリを削除できない
8	DIRECTORY_CHANGE_FAILED	ディレクトリを変更できない
9	FILE_COPY_OPEN_READ_FAILED	コピー元のファイルをオープンできない
10	FILE_COPY_OPEN_WRITE_FAILED	コピー先のファイルをオープンできない
11	FILE_COPY_WRITE_FAILED	コピー時にファイルを書き込めない
12	FILE_RENAME_FAILED	ファイルの名前を変更できない

13.8 アセンブリマッピングオブジェクト（EBh）

カテゴリ

拡張

オブジェクトの説明

このオブジェクトを使用すると、各種データセット（アセンブリ）へのI/Oコネクションを確立できます。 アセンブリは、例えば、EtherCATのPDOや、EtherNet/IPのアセンブリインスタンスを表します。 各アセンブリは、ホストアプリケーションで実装されたこのオブジェクトのインスタンスによって表されます。

すべてのライトアセンブリの合計サイズは、サポートされているライトプロセスデータの最大サイズを超えてはなりません。

アプリケーションがモジュラーデバイスオブジェクトをサポートする場合、アセンブリマッピング内のすべてのADIは、スロット順に並んでいる必要があります。

このオブジェクトが実装されていない場合、モジュールは、ネットワーク上に1個のリードアセンブリと1個のライトアセンブリのみ提供します。

サポートコマンド

オブジェクト：	Get_Attribute（01h）
インスタンス：	Get_Attribute（01h）
	Set_Attribute (02h)
	Write_Assembly_Data (10h)
	Read_Assembly_Data (11h)

オブジェクトアトリビュート（インスタンス#0）

#	名前	アクセス	データ型	値
1	Name	Get	CHAR配列	“Assembly mapping”
2	Revision	Get	UINT8	01h
3	Number of instances	Get	UINT16	アセンブリマッピングの数
4	Highest instance no.	Get	UINT16	アセンブリマッピング番号の最大値
11	Write PD instance list	Get	UINT16の配列	ライトプロセスデータにマッピング可能な現在存在するインスタンスのリスト。
12	Read PD instance list	Get	UINT16の配列	リードプロセスデータにマッピング可能な現在存在するインスタンスのリスト。

インスタンスアトリビュート（インスタンス**#1 ~ n**）

#	名前	アクセス	タイプ（メッセージサイズが**263**バイトの場合）	タイプ（メッセージサイズが**1536**バイトの場合）	説明
1	Assembly descriptor	Get	UINT32	UINT32	ビット0: 0 =ライトアセンブリ 1 =リードアセンブリ ビット1: 0 = 静的なADIアセンブリマップ 1 = 動的なADIアセンブリマップ Bit 2 ~ 31 0 =予約
2	ADI Assembly Map 0	Get	BITS32[0-61]の配列	BITS32[0-379]の配列	このアセンブリに含まれるADIアイテムの配列。 下記の「ADIアセンブリマップ」を参照。
3	ADI Assembly Map 1	Get	BITS32[62-123]の配列	BITS32[380-759]の配列	
...	...	...	...	...	
12	ADI Assembly Map 10	Get	BITS32[620-681]の配列	BITS[3800-4095]の配列	

Setアクセスは、ネットワークからの動的リマッピングが許可されている場合（すなわち、Assembly descriptorのビット1が1に設定されている場合）にのみ、アトリビュート#2 ~ 12に対してサポートされます。

ADIアセンブリマップ

アセンブリを構成するADIは、ADI アセンブリマップで定義されます。 各アセンブリで、それぞれ1536/263のメッセージサイズに対して合計4096/682のADI が許容されます。 大型のADIアセンブリマップは、380/62のADIアイテムを含むセグメントに分割する必要があります。 各セグメントは、インスタンスアトリビュート#2 ~ #12のリストとして入力する必要があります。ADIアセンブリマップの各アトリビュートは、ADIアイテムで全て満たされるまで次のアトリビュートを使用できません。

ADIアイテムの形式は以下のとおりです。

ビット数	説明
0 ~ 15	ADI番号
16 ~ 23	マッピングする最初の要素のインデックス。
24 ~ 31	後ろに続くマッピングする要素の数。

リードアセンブリとのネットワーク接続が確立すると、Anybus CompactComモジュールは、ADIアセンブリマップのすべてのアトリビュートを読み取り、その内容に対応するRemap_ADI_Read_Areaコマンドを生成します。

コマンドの詳細: Write_Assembly_Data

詳細

コマンドコード: 10h

有効な対象： インスタンス

説明

このコマンドは、ライトアセンブリマッピング内のすべてのADIにデータを書き込むのに使用します。

- コマンド詳細：

フィールド	内容
CmdExt[0]	(予約、0)
CmdExt[1]	(予約、0)
MsgData[0-n]	アセンブリデータ

- 応答詳細：

  アプリケーションは、ライト要求を受け付けるか、エラー応答を返すことが可能です。

  – 書き込みを行ったアセンブリに、プロセスデータチャネルにマッピングされているADIが含まれており、かつ、Anybus CompactComモジュールの状態がPROCESS_ACTIVEの場合、要求に対してNAKを返し、次のエラー応答を返すことを推奨します。「Attribute controlled from another channel」.

  – アセンブリのデータサイズが間違っている要求が送信された場合、「Not enough data」、「Too much data」、または「Segmentation data overflow」といったエラー応答を返す必要があります。

コマンドの詳細: Read_Assembly_Data

詳細

コマンドコード: 11h

有効な対象： インスタンス

説明

このコマンドは、リードアセンブリマッピング内のすべてのADIからデータを読み取るのに使用されます。

- コマンド詳細：

  -

- 応答詳細：

フィールド	内容
CmdExt[0]	(予約、0)
CmdExt[1]	(予約、0)
MsgData[0-n]	アセンブリデータ

13.9 モジュラーデバイスオブジェクト（ECh）

カテゴリ

拡張

オブジェクトの説明

このオブジェクトは、モジュラーデバイスを記述するのに使用されます。 モジュラーデバイスは、多数の「スロット」を持つバックプレーンで構成されます。 最初のスロットはAnybus CompactComモジュールを含む「カプラー」で占められています。 その他のすべてのスロットは、空であるか、またはモジュールで占められています。

このオブジェクトの各インスタンスは、モジュラーデバイスのスロットを表します。 インスタンス#1はカプラーに対応します。 インスタンス#2およびそれ以降はバックプレーンのスロットに対応し、占有されているものと空いているものの両方を表します。 インスタンスアトリビュートはありませんが、Get_Listコマンドはアプリケーション上のモジュールのリストを返します。

ADIをプロセスデータにマッピングするとき、アプリケーションは、各モジュールのプロセスデータをスロット順にマッピングします。 アプリケーションがそれ以外の順序でマッピングすると、Anybus CompactComモジュールはEXCEPTION状態に遷移します。

> **!** モジュラーデバイス機能の実装はネットワークによって異なります。 詳細については、それぞれのネットワークガイドを参照してください。

サポートコマンド

オブジェクト：	Get_Attribute（01h）
	Get_List（15h）
インスタンス：	-

オブジェクトアトリビュート（インスタンス#0）

#	名前	アクセス	データ型	値
1	Name	Get	CHAR配列	“Modular device”
2	Revision	Get	UINT8	01h
3	Number of instances	Get	UINT16	カプラーを含む、バックプレーンに物理的に接続されたモジュールの数。
4	Highest instance no.	Get	UINT16	最後に占有されたスロットのインスタンス番号、すなわち現在使用している最も大きいインスタンス番号。
11	Number of slot	Get	UINT16	バックプレーンにて利用可能なスロットの数（カプラーを含む）。 多くのネットワークでは、このアトリビュートに256より大きな値を設定してはいけません。
12	Number of ADIs per slot	Get	UINT16	以下の式に基づいて、どのADIがどのスロットに属するかを決定します。 ADI = slot * x + index + 1 slot = (ADI - 1) / x index = (ADI - 1) MOD x （xはこのアトリビュートの値） EtherCATとの互換性を保つため、このアトリビュートの値にスロット数（上記のアトリビュート#11）を乗じた値が4096を超えてはいけません。

コマンドの詳細: Get_List

詳細

コマンドコード: 15h

有効な対象： オブジェクト

説明

このコマンドは、バックプレーン上のモジュールと空のスロットを表すモジュールタイプ番号のリストを返します。 サポートされているリストの種類については、以下を参照してください。 リストタイプ01hの実装は必須です。

アプリケーションは、要求された数と等しい数、またはそれより少ない数（要求された数より少ないインスタンスしかアプリケーションに存在しない場合）のモジュールタイプID数（空きスロットも含む）で応答します。 モジュールタイプIDの選択は、実装者が行います。 これは、バックプレーン上のモジュールタイプごとに一意の番号となります。 事前に指定されている値は「empty slot」（空きスロット）のみで、00000000hです。 要求された開始オーダー番号が、最大のインスタンス番号より大きい場合、空の応答が返されます。 サポートされていないリストタイプが要求された場合、エラー応答「Invalid CmdExt[1]」が生成されます。

Anybus CompactComは、完全なリストを取得するため、昇順のアイテム番号を持つ複数のコマンドを発行する場合があります。

- コマンド詳細：

フィールド	内容
CmdExt[0]	予約（0）
CmdExt[1]	リストタイプ（以下を参照）
MsgData[0–1]	開始インスタンス番号
MsgData[2–3]	要求されたインスタンスの数

- 応答詳細：

フィールド	種類	内容
MsgData[0-3]	UINT32	開始インスタンスのモジュールタイプID。
MsgData[4-7]	UINT32	開始インスタンス + 1のモジュールタイプID
...	...	...

リストタイプ

リスト番号	リストタイプ
00h	予約
01h	すべてのモジュールタイプIDのリスト

13.10 SYNC オブジェクト (EEh)

カテゴリ

拡張

オブジェクトの説明

このオブジェクトには、ホストアプリケーションのSYNCに関する設定が格納されます。 アプリケーションにおけるSYNCの使い方についての詳細は、以下を参照してください。

- アプリケーションステータスレジスタ, ページ 30
- SYNC, ページ 18

サポートコマンド

オブジェクト：	Get_Attribute (01h)
インスタンス：	Get_Attribute (01h)
	Set_Attribute (02h)

オブジェクトアトリビュート (インスタンス#0)

#	名前	アクセス	データ型	値
1	Name	Get	CHAR配列	“Sync”
2	Revision	Get	UINT8	01h
3	Number of instances	Get	UINT16	0001h
4	Highest instance no.	Get	UINT16	0001h

インスタンスアトリビュート (インスタンス#1)

#	名前	アクセス	データ型	値
1	Cycle time	Get/Set	UINT32	アプリケーションのサイクルタイム (単位：ナノ秒) 。
2	Output valid	Get/Set	UINT32	SYNCイベント発生からOutput valid pointまでの時間 (単位：ナノ秒) 。 デフォルト値： 0
3	Input capture	Get/Set	UINT32	SYNCイベント発生からInput capture pointまでの時間 (単位：ナノ秒) 。 デフォルト値： 0
4	Output processing	Get	UINT32	RDPDI割り込みからOutput validまでに最低限必要な時間 (単位：ナノ秒) 。
5	Input processing	Get	UINT32	Input captureからAnybus CompactComモジュールへのライトプロセスデータの書き込みが完了するまでに必要な最大時間 (単位：ナノ秒)
6	Min cycle time	Get	UINT32	アプリケーションがサポートする最小サイクルタイム。
7	Sync mode	Get/Set	UINT16	このアトリビュートは、同期モードの選択で使用されます。 これは、アトリビュート#8のビットの列挙になります。 0： 非同期動作。 (同期動作がサポートされていない場合のデフォルト値) 1: 同期動作 2 - 65535: 予約。 サポートされていない同期モードの値に設定しようとすると、エラー応答が生成されます。
8	Supported sync modes	Get	UINT16	アプリケーションがサポートする同期モードのリストです。 各ビットは、アトリビュート7のモードに対応します。 ビット0: 1 = 非同期モードをサポート ビット1: 1 = 同期モードをサポート ビット2 ~ 15: 予約 (0)

13.11 Energyコントロールオブジェクト（F0h）

カテゴリ

拡張

オブジェクトの説明

このオブジェクトを使用すると、ホストアプリケーションにEnergy コントロール機能（電力固有の設定）が実装されます。 このオブジェクトの実装は任意です。 すべてのインスタンスアトリビュートが必要であるとされているため、アプリケーションに実装する必要があります。 実行時にアトリビュートが欠落していることをAnybusモジュールが検出した場合、適切なネットワークエラーが送信されて、Anybusオブジェクトのインスタンスアトリビュートであるエラーカウンターで「破棄された応答」のカウンターが増加します。

オブジェクトで利用可能な各インスタンスは、省電力モードに対応しています。 利用可能なモードの数は装置によって異なります。それらのモードはアプリケーションにて定義する必要があります。 インスタンスの番号が大きいほど、より多くの電力を節約できます。 最も番号が大きいインスタンスは常に「Power off」に相当します。すなわち、装置は基本的にシャットダウンされます。 オブジェクトのインスタンス1は「Ready to operate」を表します。すなわち、装置は完全に機能し、電力は全く節約されません。 そのため、意味のある実装では、常に 2 つのインスタンス、すなわち、省電力用のインスタンスと動作用のインスタンスを持ちます。 インスタンスの最大番号は8です。これらのモードは常に存在しており、動的に作成されたり削除されたりすることはありません。 インスタンスのリストに穴を空けておく（未サポートインスタンスが途中に存在する）ことは許容されません。

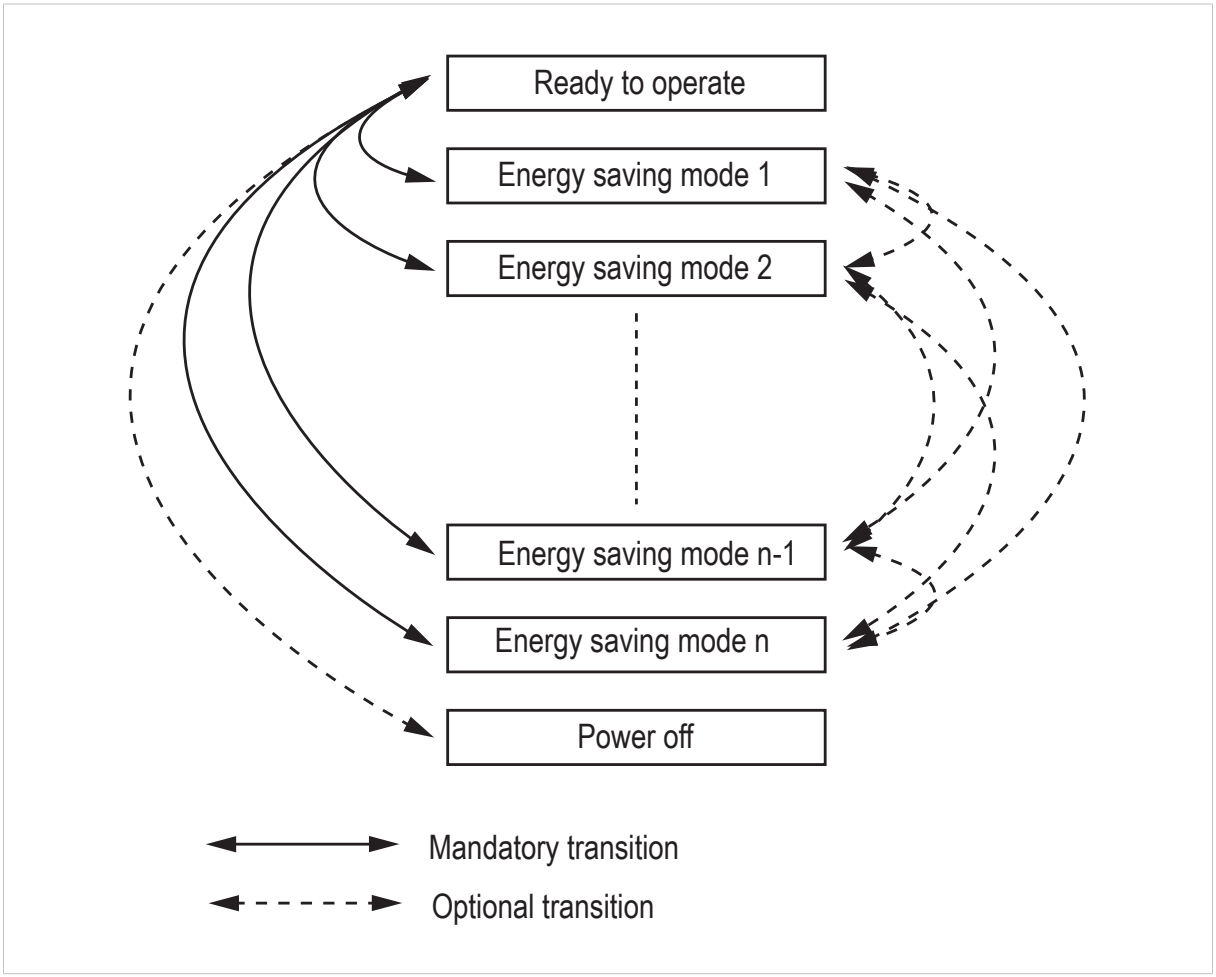

Fig. 23

サポートコマンド

オブジェクト：	Get_Attribute
	StartPause
	EndPause
	Preview_Pause_Time（PROFINETではありません）
インスタンス：	Get_Attribute

オブジェクトアトリビュート（インスタンス**#0**）

#	名前	アクセス	データ型	値
1	Name	Get	CHAR配列	“Energy Control”
2	Revision	Get	UINT8	02h
3	Number of instances	Get	UINT16	-
4	Highest instance no.	Get	UINT16	作成されたインスタンス番号のうち最も大きい番号。 最大値は8です。
11	Current Energy Saving Mode	Get	UINT16	現在使用されている省電力モードのインスタンス番号。 新しい省電力モードは、モード遷移中に表されます。「Ready to operate」はインスタンス#1と同等、「Power off」モードは最大インスタンス番号と同等になります。
12	Remaining time to destination	Get	UINT32	モードを変更したとき、このパラメータは移行が完了するまでの実際の残り時間（単位：ミリ秒）を反映します。 値を動的に生成できない場合、モードの遷移元から遷移先への移行に関する静的な値が使用されます。 この値が無限大または不明な場合、最大値0xFFFFFFFFが使用されます。
13	Energy consumption to destination	Get	FLOAT	モードを変更したとき、このパラメータは移行が完了するまでに実際に消費される電力（単位：kWh）を反映します。 値を動的に生成できない場合、モードの遷移元から遷移先への移行に関する静的な値が使用されます。 値が定義されていない場合、0.0という値が使用されます。
14	Transition to 「Power off」 mode supported	Get	BOOL	「Power off」モードへの遷移がサポートされているかどうかを示します。 0： サポートなし 1: サポートあり

インスタンスアトリビュート（インスタンス**#1 ~ #8**）

#	名前	アクセス	データ型	説明
1	ModeAttributes	Get	BITS16	ビット0: 意味 0： 静的時間および電力値のみ使用可能（アトリビュートのビット0の値は実装されていない） 1: 時間と電力に関する動的な値を使用可能 ビット1 ~ 15: 予約
2	TimeMinPause	Get	UINT32	最小休止時間（単位：ミリ秒）（t_{pause}） この値が無限大または不明な場合、最大値0xFFFFFFFFが使用されます。
3	TimeToPause	Get	UINT32	この省電力モードに移行するまでの最大所要時間。（ミリ秒、t_{off}） この値が無限大または不明な場合、最大値0xFFFFFFFFが使用されます。
4	TimeToOperate	Get	UINT32	「Ready to operate」モードに移行するのに必要な時間。（ミリ秒、t_{on}） この値が無限大または不明な場合、最大値0xFFFFFFFFが使用されます。
5	TimeMinLengthOfStay	Get	UINT32	装置がこのモードを維持しなければならない最小時間。 単位はミリ秒です。（ミリ秒、t_{off_min}） この値が無限大または不明な場合、最大値0xFFFFFFFFが使用されます。
6	TimeMaxLengthOfStay	Get	UINT32	このモードを維持できる最大時間。 単位はミリ秒です。 最大値が使用できない場合、または実装されていない場合は、最大値0xFFFFFFFFが使用されます。
7	ModePowerConsumption	Get	FLOAT	このモードで消費される電力量。（kWh） 値が定義されていない場合、0.0という値が使用されます。
8	EnergyConsumptionToPause	Get	FLOAT	このモードへ移行するのに必要な電力量。（kWh） 値が定義されていない場合、0.0という値が使用されます。
9	EnergyConsumptionToOperate	Get	FLOAT	このモードから「Ready to operate」へ移行するのに必要な電力量。（kWh） 値が定義されていない場合、0.0という値が使用されます。
10	Availability	Get	BOOL	現在のデバイスの状態において、この省電力モードが使用可能かどうかを示します。 False 使用不可 True 使用可能（アトリビュートが実装されていない場合は値）
11	Power Consumption	Get	UINT32	この状態にあるときのデバイスの消費電力を示します。

コマンドの詳細: Start_Pause

詳細

コマンドコード	10h
有効な対象：	オブジェクト

説明

このコマンドは、システムがシステムの休止を開始したいときに、ホストアプリケーションに送られます。 休止時間はミリ秒単位で指定します。 メッセージの応答には、移行先のモード (選択した省電力モードのインスタンス番号) が含まれます。

- コマンド詳細

フィールド	内容	コメント
Data[0]	休止時間 (下位ワード、下位バイト)	休止時間 (ms)
Data[1]	休止時間 (下位ワード、上位バイト)	
Data[2]	休止時間 (上位ワード、下位バイト)	
Data[3]	休止時間 (上位ワード、上位バイト)	

- 応答詳細

フィールド	内容	コメント
Data[0]	インスタンス番号 (下位バイト)	選択した電力モードのインスタンス番号
Data[1]	インスタンス番号 (下位バイト)	

要求された休止時間の間にアプリケーションが状態を選択できない場合、下表のエラーコードのいずれかを返します。

#	エラーコード	説明
0x0D	Invalid state	現在のデバイスの状態と要求された休止時間では、何らかの省電力モードに入ることができません
0x12	Value too low	要求された休止時間が短すぎます

コマンドの詳細: End_Pause

詳細

コマンドコード	11h
有効な対象：	オブジェクト

説明

このコマンドは、システムがシステム休止モードから「Ready to operate」モードへ戻りたいときに、ホストアプリケーションに送られます。 応答メッセージで、スイッチを有効化するために必要なミリ秒数が返されます。

- コマンド詳細

  （なし）

- 応答詳細

フィールド	内容	コメント
Data[0]	動作するまでの時間（下位ワード、下位バイト）	「Ready to operate」に切り替えるのに必要な時間
Data[1]	動作するまでの時間（下位ワード、上位バイト）	
Data[2]	動作するまでの時間（上位ワード、下位バイト）	
Data[3]	動作するまでの時間（上位ワード、上位バイト）	

アプリケーションが休止状態を終了できない場合、下表のエラーコードを返します。

#	エラーコード	説明
0x0D	Invalid state	現在のデバイスの状態では、休止の終了は今のところ不可能です

コマンドの詳細: **Preview_Pause_Time**

詳細

コマンドコード	12h
有効な対象：	オブジェクト

説明

PROFINETデバイスでは使用されません。

このコマンドは、アプリケーションの省電力モードの選択をシステムがプレビューしたいときに、ホストアプリケーションに送られます。 休止時間はミリ秒単位で指定します。 応答には、StartPauseサービスが送られていればアプリケーションが選択していたであろう移行先モード (すなわち選択した省電力モードのインスタンス番号) が含まれます。 省電力モードへの移行は行われません。

- コマンド詳細

フィールド	内容	コメント
Data[0]	休止時間 (下位ワード、下位バイト)	休止時間 (ms)
Data[1]	休止時間 (下位ワード、上位バイト)	
Data[2]	休止時間 (上位ワード、下位バイト)	
Data[3]	休止時間 (上位ワード、上位バイト)	

- 応答詳細

フィールド	内容	コメント
Data[0]	インスタンス番号 (下位バイト)	選択した電力モードのインスタンス番号
Data[1]	インスタンス番号 (下位バイト)	

要求された休止時間の間にアプリケーションが状態を選択できない場合、下表のエラーコードのいずれかを返します。

#	エラーコード	説明
0x0D	Invalid state	現在のデバイスの状態と要求された休止時間では、何らかの省電力モードに入ることができません
0x12	Value too low	要求された休止時間が短すぎます

13.12 ホストアプリケーションスペシフィック・オブジェクト（80h）

カテゴリ

拡張

オブジェクトの説明

このオブジェクトの機能は指定されていません。 この機能はアプリケーションで自由に指定できます。例えば、Ethernet対応のモジュールでSSIインターフェースを使用してアプリケーションのデータにアクセスするために、このオブジェクトを使用できます。

A 機能の分類

Anybus CompactComとアプリケーションのアトリビュートやサービスを含むオブジェクトは、次の二つのカテゴリに分けられます。 基本および拡張。

A.1 基本

このカテゴリには、実装または使用しなければならない必須のオブジェクト、アトリビュート、サービスが含まれます。 Anybus CompactComを起動し、選択したネットワークプロトコルでデータを送受信するにはこのカテゴリで十分であり、 産業用ネットワークの基本機能が使用されます。

製品認証を可能にする追加オブジェクトなどもここに分類されます。

A.2 拡張

このカテゴリのオブジェクトを使用すると、アプリケーションの機能を拡張できます。 ネットワークにおける基本的なデータ交換だけでなく、産業用ネットワーク固有の機能を利用できるようになります。 これにより、アプリケーションの価値が高まります。

一部の機能は非常に高度であったり、使用されることが稀であったりすることがあります。 提供されているほとんどのネットワーク機能が使用可能となり、アクセスできるため、産業用ネットワーク仕様を確認することが必要な場合があります。

B ネットワーク比較

Anybus CompactCom 40のソフトウェアインターフェースは、ネットワーク機能やホストシステムとの統合を損なうことなく、可能な限り汎用的になるように設計されています。

ホストアプリケーションを設計する際、各ネットワークシステムの制限事項と実現可能事項を理解しておくことが重要です。 多くの場合、ある特定のネットワークをサポートするために追加のソフトウェア実装は必要ありません。 ただし、ネットワーク機能の特徴を完全に引き出すために、ある程度の専用的なソフトウェア実装が必要となる場合もあります。

各種ネットワークの実装によって提供される機能の一覧を次のページの表に示します。

表の解釈方法は以下のとおりです。

- 数字は、典型的な一般的実装で期待される値を示します。
- カッコ内の数字は、専用的なソフトウェアの実装によって実現可能な値を示します。
- 最大数の診断インスタンスのうちの1つは常に、モジュールを強制的にEXCEPTION状態に移行させるため、重大度レベル「Major, unrecoverable」のインスタンス用に予約されています。
- あるデータ型がサポートされていない場合、これは当該ネットワークにその特定の型に直接対応する型がないことを意味します。 ただし、それでもそのデータを他の形式で表現できる場合があります (例えば、UINT64を4つのUINT16で表現できる、など) 。
- 表に対するネットワーク固有のコメントは、表の後に一覧表示されています。

> **!** この章の情報は、各種ネットワーク実装で実現できることの概略を示します。 特定のネットワークに関する詳しい情報は、対応するネットワークガイドを参照してください。

項目	EtherNet/IP	CC-Link	CC-Link IE Field	PROFIBUS DP-V1	PROFINET IRT	DeviceNet	Modbus-TCP	EtherCAT	Ethernet POWERLINK	BACnet/IP
ネットワークデータのフォーマット	LSBファースト	LSBファースト	LSBファースト	MSBファースト	MSBファースト	LSBファースト	LSBファースト	LSBファースト	LSBファースト	MSBファースト
アサイクリックデータのサポート	あり	なし	あり	あり	あり	あり	あり	あり	あり	あり
ADIあたりの最大要素数	255	112/255	255	240	255	255	32 (255)	255	254	1
ADIの最大サイズ（単位：バイト）	1524	112/256	1524	240	1308	512	32 (1524)	1524	N/A	4
アドレス指定可能な最小のADI番号	1	1	1	1	1	1	1	1	1	1
アドレス指定可能な最大のADI番号	65535	65535	65535	65025	32767	65535	3839 (61424)	57343 (16383)	57343	65535
最大のライトプロセスデータ（単位：バイト）	1448	368	1536	244	1308	512	1536	1486	1490	N/A
最小のライトプロセスデータ（単位：バイト）	0	0	0	0	0	0	0	0	0	N/A
最大のリードプロセスデータ（単位：バイト）	1448	368	1536	244	1308	512	1536	1486	1490	N/A
最小のリードプロセスデータ（単位：バイト）	0	0	0	0	0	0	0	0	0	N/A
最大のプロセスデータ（リード + ライト、単位：バイト）	2896	736	3072	488	2616	1024	3072	2972	2980	N/A
最小のプロセスデータ（リード + ライト、単位：バイト）	0	0	0	1	0	0	0	0	0	N/A
「Get/Set_Indexed_Attribute」が必要	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ	はい	はい	いいえ
「Get_Instance_Number_By_Order」が必要	はい	いいえ	いいえ	いいえ	はい	はい	いいえ	はい	いいえ	はい
稼働中のリマッピング	可	不可	不可	可	可	不可	不可	可	可	不可
診断インスタンスの最大数	6	6	2	6	6	6	6	6	1	1
ネットワークリセットタイプ0のサポート：「Power-on-reset」	あり	なし	なし	なし	あり	あり	なし	あり	あり	あり
ネットワークリセットタイプ1のサポート：「Factory default reset」	あり	なし	なし	なし	あり	あり	なし	あり	なし	なし
SINT64のサポート	あり	あり	あり	あり	あり	あり	あり	あり	あり	なし
UINT64のサポート	あり	あり	あり	あり	あり	あり	あり	あり	あり	なし
FLOATのサポート	あり	あり	あり	あり	あり	あり	あり	あり	あり	あり
Cycle time	1 ms	~1 ms	200 µs ~ 200 ms	-	250 µs	10 ms	-	100 µs	200 µs ~ 2147483 µs	N/A

- EtherNet/IP（EIP）：
  - コマンドGet_Instance_Number_By_Orderは、CIPネットワークからパラメータオブジェクトのアトリビュートにアクセスするときに必要です。
  - コマンドGet_Instance_Number_By_Orderは、内部Webページをサポートするモジュールでは、パラメータWebページを開くときにも使用されます。
- CC-Link（CCL）：
  - ADI当たりの最大要素数と最大ADIサイズはADIマッピングによって異なります。Anybus CompactCom 40 CC-Linkネットワークガイドを参照してください。
  - サイクルタイムは伝送速度10 Mbps、1局占有のリモートデバイス局がネットワーク上に1台だけ存在する場合になります。 ネットワークコンフィグレーションに依存します。
- PROFIBUS（DPV1）：
  - 技術的な理由により、1～256のADI番号を使用することは一般に推奨されません。特定のPROFIBUSコンフィグレーションツールを使用した場合に問題が起こる場合があります。 その場合、アドレス指定可能な最小ADI番号は257となります。
- DeviceNet（DEV）：
  - コマンドGet_Instance_Number_By_Orderは、CIPネットワークからパラメータオブジェクトのアトリビュートにアクセスするときに必要です。
- Modbus-TCP（EIT）：
  - アドレス指定可能な最大ADI番号のデフォルトは3839です。
  - ADIインデックスビットの数を変更してADIのサイズを制限する場合、アドレス指定可能な最大ADIは61424まで大きくなります。
- EtherCAT（ECT）：
  - モジュールが汎用モードの場合、アドレス指定可能な最大ADIは57343です。
  - モジュラーデバイスプロファイルが有効化されていると、アドレス指定可能な最大ADIは16383です。
  - コマンドGet_Instance_Number_By_Order（またはGet_Instance_Numbers）は初期化中にADIの数を検出するために使用されます。
  - ネットワークリセットタイプ0は、ファームウェアをアップグレードする目的でサポートされています。
- Ethernet POWERLINK（EPL）：
  - このネットワークには、最大ADIサイズの制限がありません。 現在の実装ではADIあたり30 kBまで許容されます。
- BACnet/IP：
  - リードプロセスデータはサポートされていません
  - BACnet製品がアドバンスドモードではない場合、コマンドGet_Instance_Number_By_Orderを使用して初期マッピングが実行されます。

C 産業用Ethernetの比較

Anybus CompactCom 40シリーズの製品ファミリーは、多くの産業用Ethernetをサポートしています。また、製品ファミリーにはCommon（共通）Ethernetモジュールもあり、そのままの状態でEthernet基本機能を使用するか、所定のネットワークに対応したファームウェアをダウンロードして使用するかを選べるプラットフォームも提供します。

さまざまなネットワークで使用できるEthernet機能を下表に示します。

> 特定のネットワークに固有の機能や特定のネットワークに関する詳しい情報については、対応するネットワークガイドを参照してください。

項目	**EtherNet/IP**	**PROFINET IRT**	**Modbus-TCP**	**EtherCAT**	**Ethernet POWERLINK**	**Common Ethernet**	**CC-Link IE Field**	**BACnet/IP**
アプリケーションインターフェース	8/16 ビット DPRAM（30 ns） SPI（最大 20 Mbit/s） シフトレジスタ（12.5 MHz）	8/16 ビット DPRAM（30 ns） SPI（最大 20 Mbit/s） シフトレジスタ（12.5 MHz）	8/16 ビット DPRAM（30 ns） SPI（最大 20 Mbit/s） シフトレジスタ（12.5 MHz）	8/16 ビット DPRAM（30 ns） SPI（最大 20 Mbit/s） シフトレジスタ（12.5 MHz）	8/16 ビット DPRAM（30 ns） SPI（最大 20 Mbit/s） シフトレジスタ（12.5 MHz）	8/16 ビット DPRAM（30 ns） SPI（最大 20 Mbit/s） シフトレジスタ（12.5 MHz）	8/16 ビット DPRAM（30 ns） SPI（最大 20 Mbit/s） シフトレジスタ（12.5 MHz）	8/16 ビット DPRAM（30 ns） SPI（最大 20 Mbit/s） シフトレジスタ（12.5 MHz）
ネットワークサイクルタイム	≥ 1 ms	≥ 250 µs	N/A	≥ 100 µs	≥ 200 µs	N/A	≥ 230 µs	N/A
レイテンシ（遅延）	32バイトのプロセスデータで160µs	tbd	N/A	< 250 ns	< 15 µs	N/A	< 2µs	N/A
ジッタ	32バイトのプロセスデータで40 µs	tbd	N/A	< 200 ns	200 ns	N/A	N/A	N/A
最大プロセスデータ（入/出力、バイト）	1448 / 1448	1440/1440（statusバイトを含む）	1536 / 1536	1486 / 1486	1490 / 1490	-	1536 / 1536	N/A
最大パラメータデータ（入/出力、バイト）	1448	1512 / 1294	248 / 248	1524 /1524	1490 / 1490	-	960 / 960	4 / 4
ITプロトコルのサポート	TCP、UDP、FTP、HTTP、SMTP	TCP、UDP、FTP、HTTP、SMTP	TCP、UDP、FTP、HTTP、SMTP	TCP、UDP、FTP、HTTP、SMTP	なし	TCP、UDP、FTP、HTTP、SMTP	なし	TCP、UDP、FTP、HTTP、SMTP
Webサーバー搭載	あり	あり	あり	あり	なし	あり	なし	あり
HTTPフォワーディングオプション	あり	あり	あり	あり	なし	なし	なし	あり
トランスペアレントEthernet（RMII、別製品バージョン）	あり	あり	あり	あり	なし	なし	なし	なし
ネットワークアクセス可能なFLASHディスク	あり、28 Mbyte	あり、28 Mbyte	あり、28 Mbyte	あり、28 Mbyte	なし	あり、28 Mbyte	なし	あり、28 Mbyte
ソケット通信サポート	同時に最大20の接続	同時に最大20の接続	同時に最大20の接続	同時に最大20の接続	なし	同時に最大20の接続	なし	同時に最大8の接続

項目	EtherNet/IP	PROFINET IRT	Modbus-TCP	EtherCAT	Ethernet POWERLINK	Common Ethernet	CC-Link IE Field	BACnet/IP
ソケットのデータスループット	最大70 Mbit/s	20 Mbit/s	-	-	-	-	-	-
ポート無効化サポート	あり	なし	あり	なし	N/A	なし	なし	あり
診断機能	LED出力、ABCC診断オブジェクト、EtherNet/IP診断カウンターサポート、Webサイト	LED出力、ABCC診断オブジェクト、Webサイト、SNMP MIB2	LED、ABCC診断オブジェクト、Webサイト	LED出力、ABCC診断オブジェクト、ESCでの診断	LED出力、ABCC診断オブジェクト	-	LED出力、ABCC診断オブジェクト、アサイクリック「Get Statistics」サービス	LED、ABCC診断オブジェクト、Webサイト
ノードアドレス設定	コンフィグレーションオブジェクト、DIPスイッチ、ネットワーク経由、IPconfig、HTTP	コンフィグレーションオブジェクト、ネットワーク経由、IPconfig、HTTP	コンフィグレーションオブジェクト、DIPスイッチ、IPconfig、HTTP	コンフィグレーションオブジェクト、DIPスイッチ、IPconfig、HTTP	コンフィグレーションオブジェクト、DIPスイッチ	コンフィグレーションオブジェクト、DIPスイッチ、IPconfig、HTTP	コンフィグレーションオブジェクト、DIPスイッチ	コンフィグレーションオブジェクト、DIPスイッチ、IPconfig、HTTP
認証	UL、cUL	UL、cUL	UL、cUL	UL、cUL	UL、cUL	UL、cUL	UL、cUL	UL、cUL
ネットワークコンフォーマンス	CT11、EIP PlugFest	PROFINET 2.31	Modbus TCPコンフォーマンステスト3.0	EtherCATコンフォーマンステスト合格（2014年6月6日）	DS301 V1.1.0	N/A	CC-Link IE フィールドネットワークインテリジェントデバイスコンフォーマンステスト	N/A
ITセキュリティ	HTTP認証（basic + digest）、FTPパスワード	HTTP認証（basic + digest）、FTPパスワード	HTTP認証（basic + digest）、FTPパスワード	HTTP認証（basic + digest）、FTPパスワード	N/A（ITサポートなし）	HTTP認証（basic + digest）、FTPパスワード	N/A（ITサポートなし）	HTTP認証（basic + digest）、FTPパスワード
セーフティサポート	CIP Safety、T100 IO&ブラックチャンネル。	PROFIsafe、T100 IO&ブラックチャンネル	なし	FSoE、T100 IO&ブラックチャンネル、予定	TBD	なし	なし	なし
Secure Host IP Configuration Protocol (HICP)	あり	あり	あり	あり	なし	あり	なし	あり
HMS Firmware Managerサポート	あり	あり	あり	あり	あり	あり	あり	あり

D オブジェクトの概要

Anybus CompactCom 40シリーズの各デバイスは、この設計ガイドと各ネットワークガイドで説明されているオブジェクトのサブセットをサポートします。 概要を下表に示します。

モジュールのファームウェアが最近アップグレードされている場合、本ドキュメントの次回改訂時に下表は更新される可能性があります。

D.1 Anybus モジュールオブジェクト

これらのオブジェクトは製品に実装されています。

		EtherNet/IP	CC-Link	CC-Link IE Field	PROFIBUS DP-V1	PROFINET	DeviceNet	Modbus-TCP	EtherCAT	Ethernet POWERLINK	BACnet/IP	Common Ethernet
01h	Anybusオブジェクト	可	可	可	可	可	可	可	可	可	可	可
02h	診断オブジェクト	可	可	可	可	可	可	可	可	可	可	可
03h	ネットワークオブジェクト	可	可	可	可	可	可	可	可	可	可	可
04h	ネットワーコンフィグレーションオブジェクト	可	可	可	可	可	可	可	可	可	可	可
07h	ソケットインターフェースオブジェクト	可	不可	不可	不可	可	不可	可	可	不可	可	可
08h	ネットワークCC-Linkオブジェクト	不可	可	不可	不可	不可	不可	不可	不可	不可	不可	不可
09h	SMTPクライアントオブジェクト	可	不可	不可	不可	可	不可	可	可	不可	可	可
0Ah	Anybusファイルシステムインターフェース・オブジェクト	可	可	可	可	可	可	可	可	可	可	可
0Ch	ネットワークEthernetオブジェクト	可	不可	不可	不可	可	不可	可	可	不可	可	可
0Dh	CIPポートコンフィグレーションオブジェクト	可	不可	不可	不可	不可	不可	不可	不可	不可	不可	不可
0Eh	ネットワークPROFINET IOオブジェクト	不可	不可	不可	不可	可	不可	不可	不可	不可	不可	不可
10h	PROFIBUS DP-V0診断オブジェクト	不可	不可	不可	可	不可	不可	不可	不可	不可	不可	不可
11h	ファンクショナルセーフティモジュール・オブジェクト	可	不可	不可	不可	可	不可	不可	不可	不可	不可	不可
12h	ネットワークCC-Link IEフィールドオブジェクト	不可	不可	可	不可	不可	不可	不可	不可	不可	不可	不可

*Anybus*ファイルシステムインターフェース・オブジェクト（*0Ah*）は、次のフィールドバスおよびネットワークに対して、ホストアプリケーションからのファームウェア更新にのみ使用できます。 *CC-Link*、*CC-Link IE Field*、*PROFIBUS DP-V1*、*DeviceNet*、*Ethernet POWERLINK*、*BACnet/IP*

D.2 ホストアプリケーションオブジェクト

これらのオブジェクトはホストアプリケーションで実装することが可能です。 アプリケーションによっては、ネットワークで利用可能なオブジェクトのすべてが必要となるわけではない場合もあります。

		EtherNet/IP	CC-Link	CC-Link IE Field	PROFIBUS DP-V1	PROFINET	DeviceNet	Modbus-TCP	EtherCAT	Ethernet POWERLINK	BACnet/IP	Common Ethernet
E6h	CC-Link IE Fieldネットワークホスト・オブジェクト	不可	不可	可	不可	不可	不可	不可	不可	不可	不可	不可
E7h	Energyレポーティングオブジェクト	可	不可	不可	不可	不可	可	不可	不可	不可	不可	不可
E8h	ファンクショナルセーフティオブジェクト	可	不可	不可	不可	可	不可	不可	不可	不可	不可	不可
E9h	POWERLINKオブジェクト	不可	不可	不可	不可	不可	不可	不可	不可	可	不可	不可
EAh	アプリケーション・ファイルシステムインターフェース・オブジェクト	可	不可	不可	不可	可	不可	可	可	不可	可	可
EBh	アセンブリマッピングオブジェクト	可	不可	不可	不可	不可	不可	不可	可	不可	不可	不可
ECh	モジュラーデバイスオブジェクト	可	不可	不可	可	可	可	不可	可	不可	不可	不可
EDh	CIP Identityホストオブジェクト	可	不可	不可	不可	不可	可	不可	不可	不可	不可	不可
EEh	SYNCオブジェクト	可	不可	不可	不可	可	不可	不可	可	可	不可	不可
EFh	BACnetホストオブジェクト	不可	不可	不可	不可	不可	不可	不可	不可	不可	可	不可
F0h	Energyコントロールオブジェクト	可	不可	不可	不可	可	可	不可	不可	不可	不可	不可
F5h	EtherCATオブジェクト	不可	不可	不可	不可	不可	不可	不可	可	不可	不可	不可
F6h	PROFINET IOオブジェクト	不可	不可	不可	不可	可	不可	不可	不可	不可	不可	不可
F7h	CC-Linkホストオブジェクト	不可	可	不可	不可	不可	不可	不可	不可	不可	不可	不可
F8h	EtherNet/IPホストオブジェクト	可	不可	不可	不可	不可	不可	不可	不可	不可	不可	不可
F9h	Ethernet ホストオブジェクト	可	不可	可	不可	可	不可	可	可	可	可	可
FAh	Modbusホストオブジェクト	不可	不可	不可	不可	不可	不可	可	不可	不可	不可	不可
FCh	DeviceNetホストオブジェクト	不可	不可	不可	不可	不可	可	不可	不可	不可	不可	不可
FDh	PROFIBUS DP-V1オブジェクト	不可	不可	不可	可	不可	不可	不可	不可	不可	不可	不可
FEh	アプリケーションデータオブジェクト	可	可	可	可	可	可	可	可	可	可	可
FFh	アプリケーションオブジェクト	可	可	可	可	可	可	可	可	可	可	可

E コンフォーマンステスト情報

スタンドアロンモード

スタンドアロンのシフトレジスタモードでコンフォーマンステストに合格するには、ホストアプリケーションはいくつかの仮想アトリビュートを実装する必要があります。

E.1 EtherCAT

シフトレジスタモードでEtherCAT認定テストに合格するために必要な仮想アトリビュート：

E.1.1 実装必須

EtherCATオブジェクト（F5h）、インスタンス#1:

アトリビュート#	アトリビュート名	値の範囲	**Default value**	説明
1	Vendor ID	UINT32	0xE000001B HMS	HMSセカンダリベンダーID。セカンダリベンダーIDがCTテストに合格することはありません。

E.1.2 任意 – 製品の機能性、カスタマイズ、識別を改善

EtherCATオブジェクト（F5h）、インスタンス#1:

アトリビュート#	アトリビュート名	値の範囲	**Default value**	説明
2	Product code	UINT32	00000036h ABCC 40 ECT	
6	Manufacture Device Name	CHAR配列（最大64バイト）	“Anybus CompactCom 40 EtherCAT”	

E.2 CC-Link

シフトレジスタモードでCC-Link認定テストに合格するために必要な仮想アトリビュート：

E.2.1 実装必須

ホストCC-Linkオブジェクト（F7h）、インスタンス #1:

アトリビュート#	アトリビュート名	値の範囲	**Default value**	説明	テストリファレンスにおける検証
1	Vendor code	NA	NA	CLPAによって割り当てられた会員番号から抽出。	3. (1) BAP-C0401-012-Fに準拠した局情報の確認。

E.2.2 任意 – 製品の機能性、カスタマイズ、識別を改善

ホストCC-Linkオブジェクト（F7h）、インスタンス #1:

アトリビュート#	アトリビュート名	値の範囲	**Default value**	説明	テストリファレンスにおける検証
2	SW Version	1~63	ABCCバージョンによって異なります。	CC-Linkネットワークの動作が影響を受けるときにインクリメントされます。	3. (1) BAP-C0401-012-Fに準拠した局情報の確認。
3	Model code	1~127	127	モジュールプロファイルに対応します。	3. (1) 局情報の確認。 8 (1) BAP-C0401-012-Fに準拠したプロファイルの確認。

E.3 Ethernet POWERLINK

シフトレジスタモードでEthernet POWERLINK認定テストに合格するために必要な仮想アトリビュート：

E.3.1 実装必須

POWERLINK オブジェクト（E9h）、インスタンス#1:

アトリビュート#	アトリビュート名	値の範囲	**Default value**	説明	注。
1	Vendor ID	UINT32	NA	この値は、CANopen IDオブジェクト（0x1018、サブインデックス0x01）のデフォルト値を置き換えます。	一意のベンダーIDがEPSGによってベンダーに割り当てられます（**SHALL** be assigned）。

E.3.2 任意 – 製品の機能性、カスタマイズ、識別を改善

POWERLINK オブジェクト（E9h）、インスタンス#1:

アトリビュート#	アトリビュート名	値の範囲	**Default value**	説明
2	Product code	UINT32	0x00000028	この値は、CANopen IDオブジェクト（0x1018、サブインデックス0x02）のデフォルト値を置き換えます。
3	Revision High word	UINT16	0x0000	この値は、CANopen IDオブジェクト（0x1018、サブインデックス0x03、上位ワード）のデフォルト値を置き換えます。
4	Revision low word	UINT16	ABCC（ハードウェアリビジョン）	この値は、CANopen IDオブジェクト（0x1018、サブインデックス0x03、下位ワード）のデフォルト値を置き換えます。
6	Manufacture Device Name	CHAR配列（最大64バイト）	“Anybus CompactCom 40 Ethernet POWERLINK”	メーカーのデバイス名オブジェクト（0x1008）に対応します。
14	Manufacture Name	CHAR配列（最大64バイト）	“HMS”	インターフェ - スグループオブジェクト（0x1030）でインターフェース記述文字列の一部として使用されます。

SYNC信号が確実に機能するためには、SYNCオブジェクト（EEh）、インスタンス#1のアトリビュートを次のように定義します。

アトリビュート#	アトリビュート名	値の範囲	**Default value**	説明
1	Cycle time	UINT32	-	NMT Cycle Length Object 0x1006（マイクロ秒からナノ秒に変換）。
7	Sync mode	UINT16	-	アトリビュート#8が同期動作のサポートを示している場合、ABCCはこのアトリビュートを同期動作の開始時には1に、非同期動作の開始時には0に設定します。 同期動作がサポートされていない場合、ABCCは決して値を変更しません。
8	Supported sync modes	UINT16	-	ビット0: 非同期動作。（同期動作がサポートされていない場合のデフォルト値。） ビット1: 1 = 同期動作サポート Bit 2 ~ 15: 予約。 ゼロに設定します。

E.4 EtherNet/IP

シフトレジスタモードでEtherNet/IP認定/plugfestに合格するために必要な仮想アトリビュート：

E.4.1 実装必須

EtherNet/IPオブジェクト（F8h）、インスタンス#1:

アトリビュート#	アトリビュート名	値の範囲	**Default value**	説明
1	Vendor ID	UINT16	0x005A HMS	-

E.4.2 任意 – 製品の機能性、カスタマイズ、識別を改善

EtherNet/IPオブジェクト（F8h）、インスタンス#1:

アトリビュート#	アトリビュート名	値の範囲	**Default value**	説明
3	Product code	UINT16	0x0037 ABCC 40 EIP	製品コードはベンダーによって割り当てられます
4	Revision	UINT8構造体 UINT8	Anybus CompactCom ファームウェアリビジョン	最終製品のリビジョン
6	Product Name	CHAR配列（最大32バイト）	“CompactCom 40 EtherNet/IP(TM)”	製品名

E.5 DeviceNet

シフトレジスタモードでDeviceNet認定テストに合格するために必要な仮想アトリビュート：

E.5.1 実装必須

DeviceNetオブジェクト（FCh）、インスタンス#1:

アトリビュート#	アトリビュート名	値の範囲	**Default value**	説明
1	Vendor ID	UINT16	0x005A HMS	-

E.5.2 任意 – 製品の機能性、カスタマイズ、識別を改善

DeviceNetオブジェクト（FCh）、インスタンス#1:

アトリビュート#	アトリビュート名	値の範囲	**Default value**	説明
3	Product code	UINT16	0x003F ABCC 40 DEV	製品コードはベンダーによって割り当てられます
4	Revision	UINT8構造体 UINT8	Anybus CompactCom ファームウェアリビジョン	最終製品のリビジョン
6	Product Name	CHAR配列（最大32バイト）	“CompactCom 40 DeviceNet(TM)”	製品名

E.6 Modbus-TCP

シフトレジスタモードでModbus-TCP認定テストに合格するために必要な仮想アトリビュート：

E.6.1 実装必須

Modbusホストオブジェクト（FAh）、インスタンス#1:

アトリビュート#	アトリビュート名	値の範囲	**Default value**	説明
1	Vendor Name	CHAR配列	HMS	-

E.6.2 任意 – 製品の機能性、カスタマイズ、識別を改善

Modbusホストオブジェクト（FAh）、インスタンス#1:

アトリビュート#	アトリビュート名	値の範囲	**Default value**	説明
2	Product Code	CHAR配列（最大244文字）	“Anybus CompactCom 40 Modbus-TCP”	Product codeはベンダーによって割り当てられます。
3	Major Minor Revision	CHAR配列（最大244文字）	“”	最終製品のリビジョン
4	Vendor URL	CHAR配列（最大244文字）	“”	ベンダーから提供されたURL
5	Product Name	CHAR配列（最大244文字）	“”	ベンダー定義の製品名
6	Model Name	CHAR配列（最大244文字）	“”	ベンダー定義のモデル名
7	User Application Name	CHAR配列（最大244文字）	“”	ユーザーアプリケーションの名称

F 稼働中のプロセスデータのリマッピング

このAppendixでは、ネットワークから送られてくるリード/ライトプロセスデータのリマッピング要求を処理する方法について説明します。

この機能はEtherNet/IP、EtherCAT、PROFINET、PROFIBUS、Ethernet POWERLINKで使用できます。

F.1 SPIモード

SPIモードでは、テレグラムは全二重で送信されます。 図中では、MISOおよびMOSIとしてペアリングされたテレグラムとして示されています。 SPIモードの詳細については、*SPIホスト通信, ページ 37*を参照してください

F.1.1 リードプロセスデータ

アプリケーションがAnybus CompactCom 40からRemap_ADI_Read_Area要求を受信し、同要求に対して肯定応答を返すと、Anybus CompactCom 40はネットワークから次回以降新しいプロセスデータを受信した時から、新しいマッピングに従ってアプリケーションにリードプロセスデータの送信を開始します。 更新されていないリードプロセスデータは、古いマッピングに従って送信されます。

Anybus CompactCom 40は、リードプロセスデータが非アクティブ/無効である状態で、アプリケーションに Remap_ADI_Read_Area要求を送信します。 新しいマッピングによる有効なプロセスデータは、通常、Anybus CompactCom 40が次回にPROCESS_ACTIVE状態になるまで検出されません。

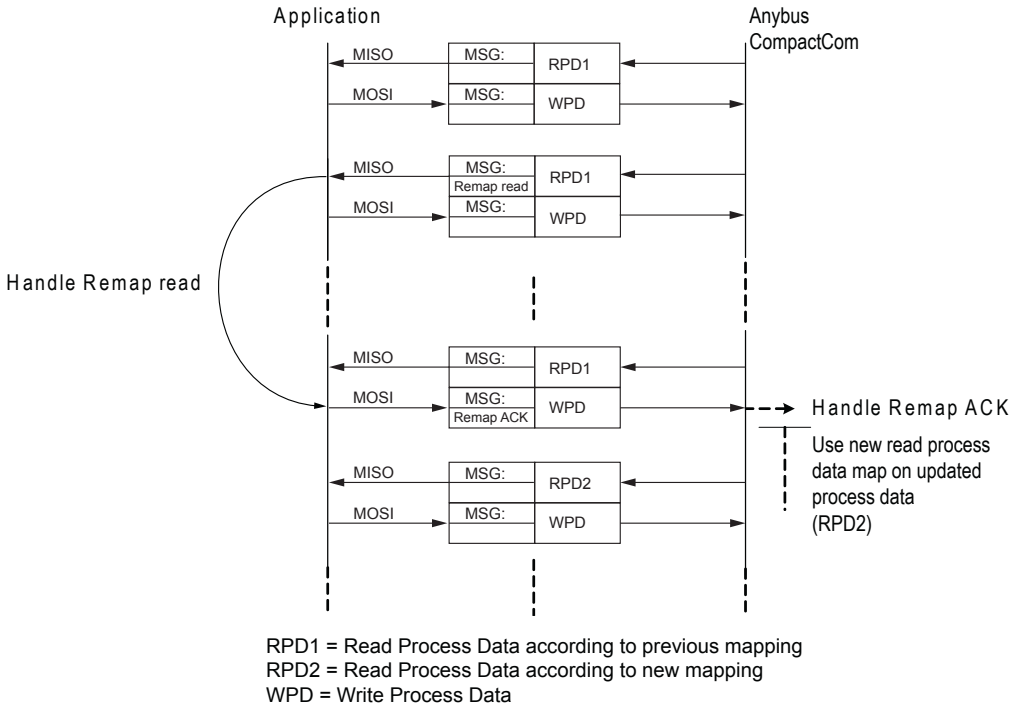

Fig. 24

F.1.2 ライトプロセスデータ

Remap_ADI_Write_Area要求を受信すると、アプリケーションはRemap_ADI_Write_Area要求に対する肯定応答を返すSPIテレグラムから新しいマッピングに従って、プロセスデータを送信します。

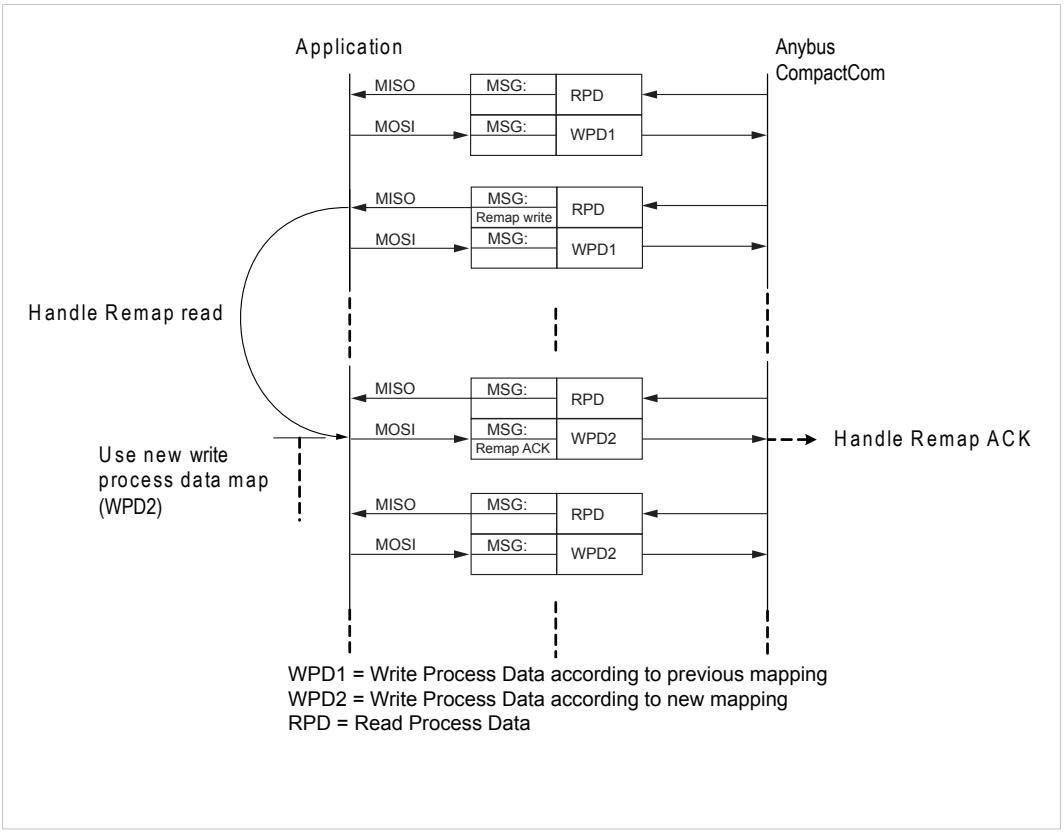

Fig. 25

F.2 パラレルモード、8/16ビット、イベント駆動型

F.2.1 リードプロセスデータ

アプリケーションがAnybus CompactCom 40からRemap_ADI_Read_Area要求を受信し、同要求に対し肯定応答を返すと、Anybus CompactCom 40はネットワークから次回以降新しいプロセスデータを受信した時から、新しいマッピングに従ってアプリケーションにリードプロセスデータの送信を開始します。

Anybus CompactCom 40は、リードプロセスデータが非アクティブ/無効である状態で、アプリケーションに Remap_ADI_Read_Area要求を送信します。 新しいマッピングによる有効なプロセスデータは、通常、Anybus CompactCom 40が次回にPROCESS_ACTIVE状態になるまで検出されません。

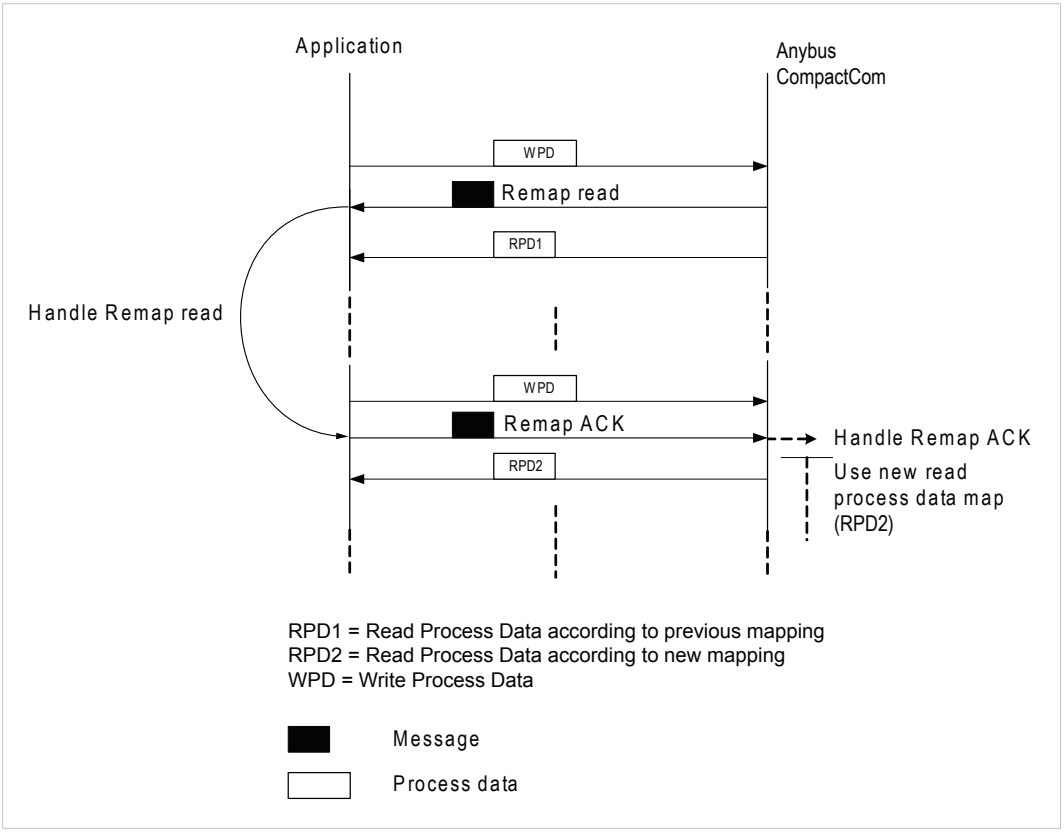

Fig. 26

F.2.2 ライトプロセスデータ

Remap_ADI_Write_Area要求を受信すると、アプリケーションはRemap_ADI_Write_Area要求に対する肯定応答を返す前に、新しいマッピングによるプロセスデータのAnybus CompactCom 40への送信を開始します。

Anybus CompactCom 40は、Remap_ADI_Write_Area要求をアプリケーションに送信してからこのリマッピング要求に対して肯定応答されるかどうかが判明するまでの間、ライトプロセスデータは無効であるとみなします。

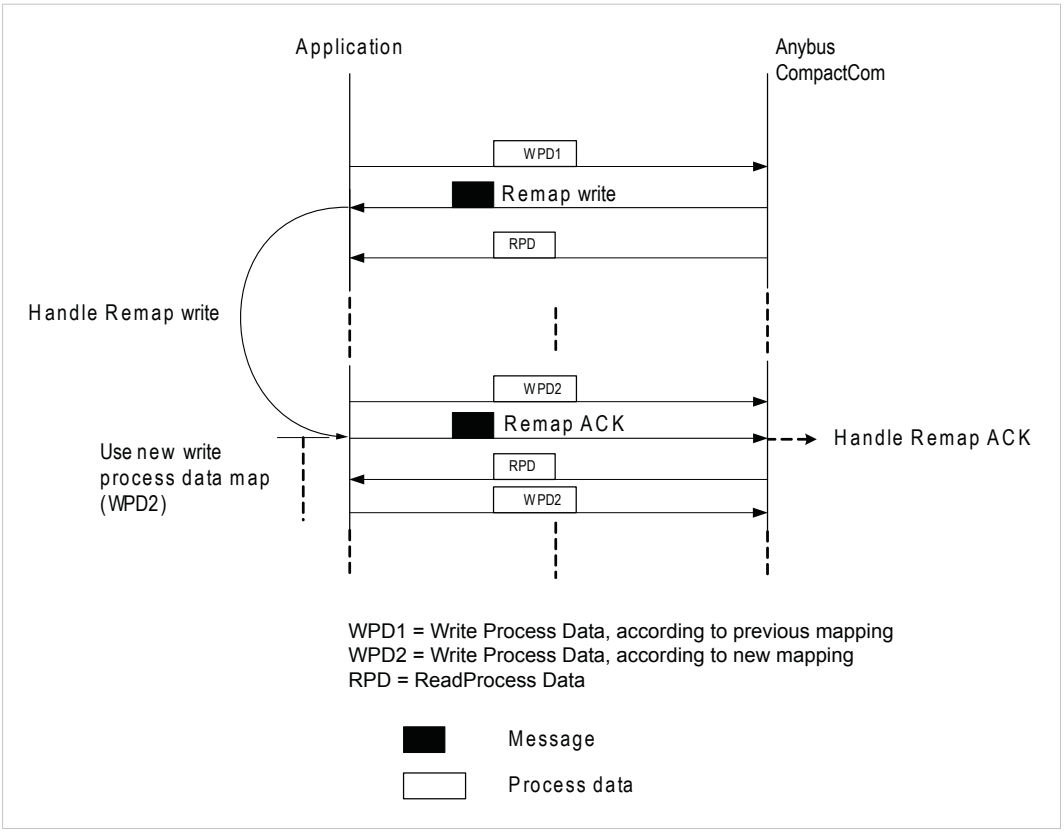

Fig. 27

F.3 後方互換モード

このセクションでは、Anybus CompactCom 30シリーズとの後方互換性を持つ、パラレルおよびシリアルモードでのプロセスデータの稼動時リマッピングについて説明します。

テレグラムはピンポン方式で交換されます。

F.3.1 パラレルモード

パラレルモードでのプロセスデータの稼動時リマッピングは非常に単純です。下図を参照してください。

リードプロセスデータ

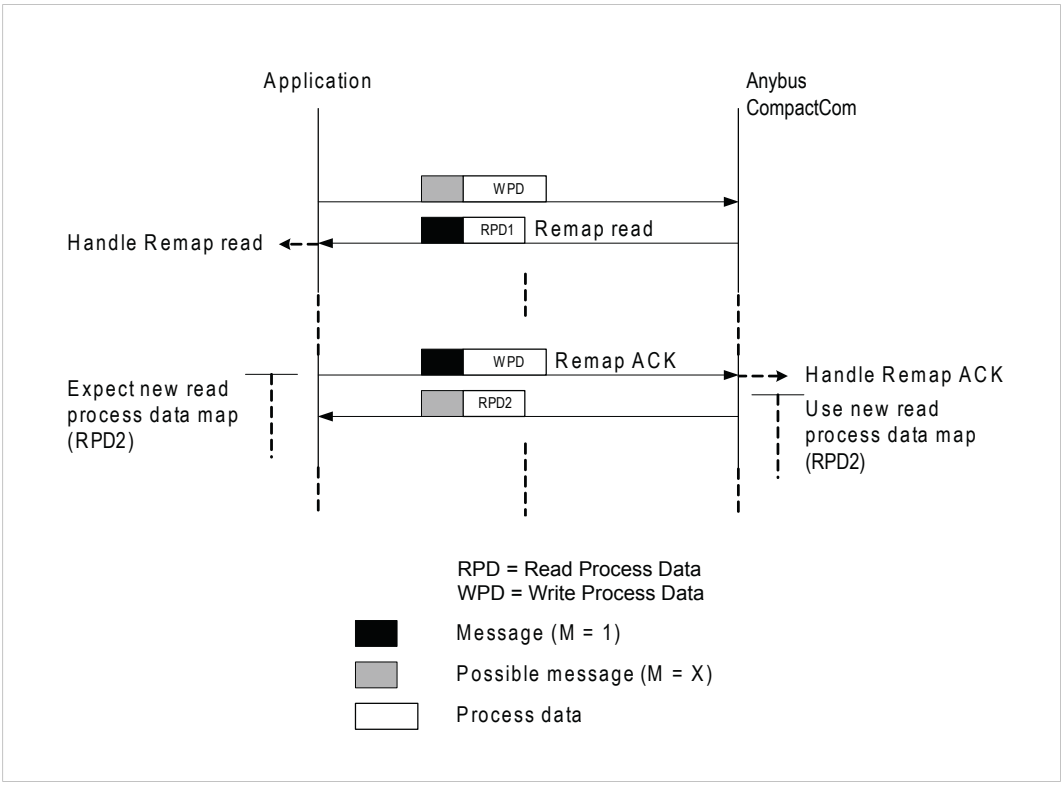

Fig. 28

ライトプロセスデータ

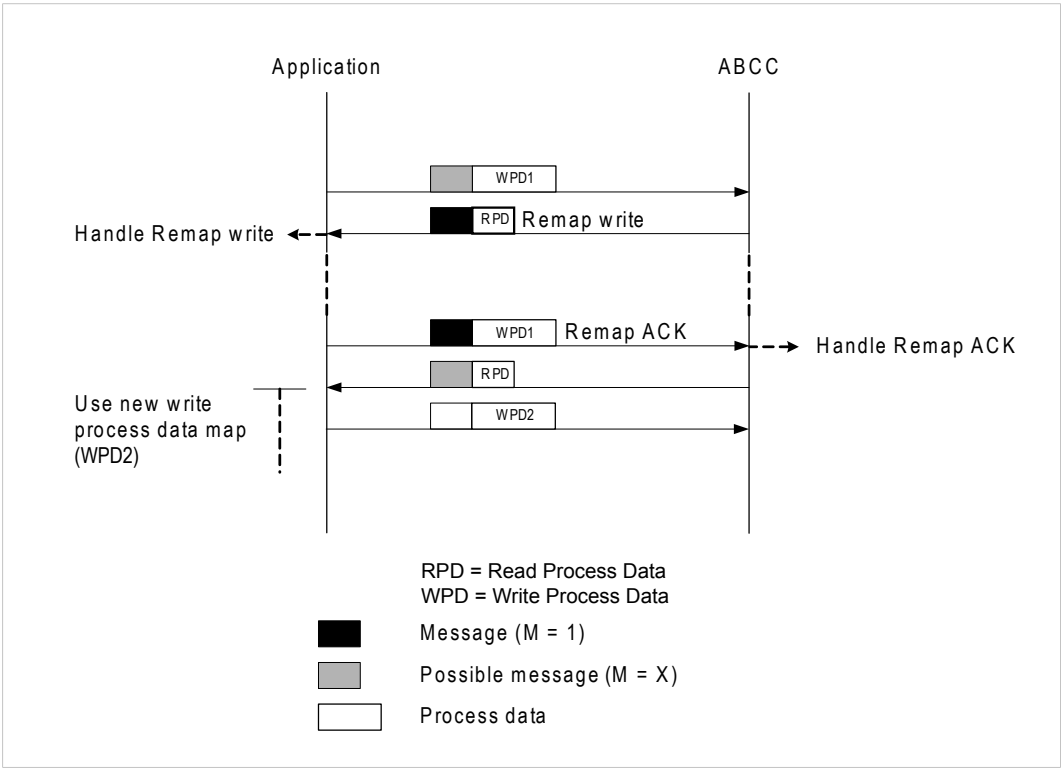

Fig. 29

F.3.2 シリアルモード

テレグラムはピンポン方式で交換され、メッセージのないテレグラムが各コマンドの終了となります。この方式では、リマッピングの効果が現れる前に、多数のテレグラムが交換される必要があります。

このモードはAnybus CompactCom 30と後方互換性があります。

リードプロセスデータ

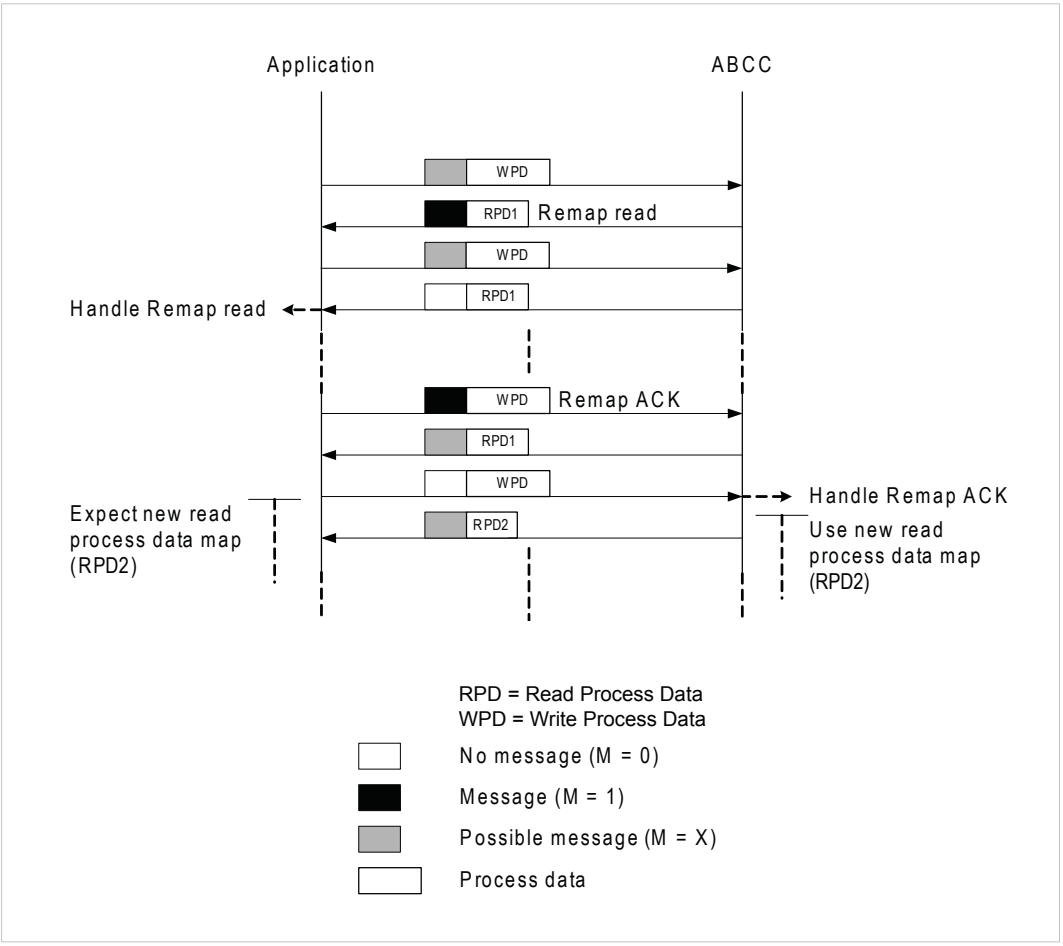

Fig. 30

ライトプロセスデータ

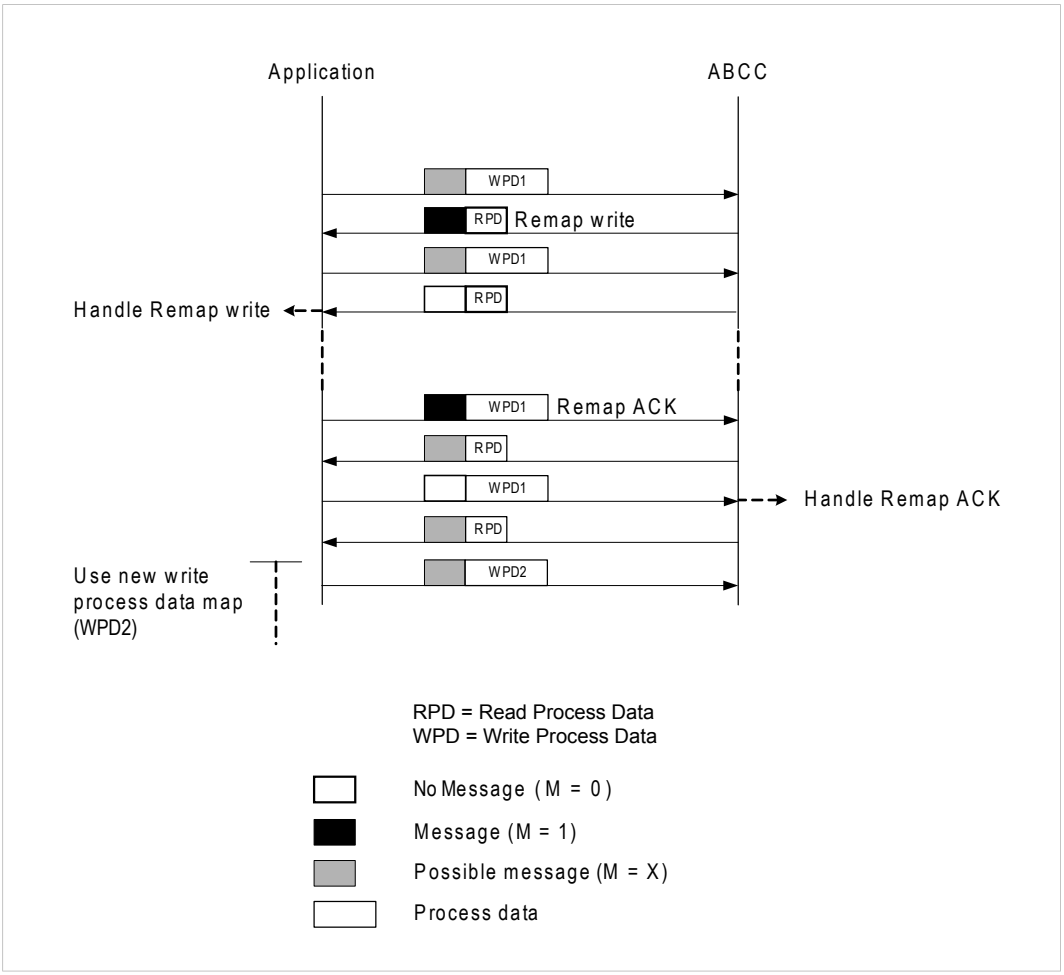

Fig. 31

F.4 例： Remap_ADI_Write_Area

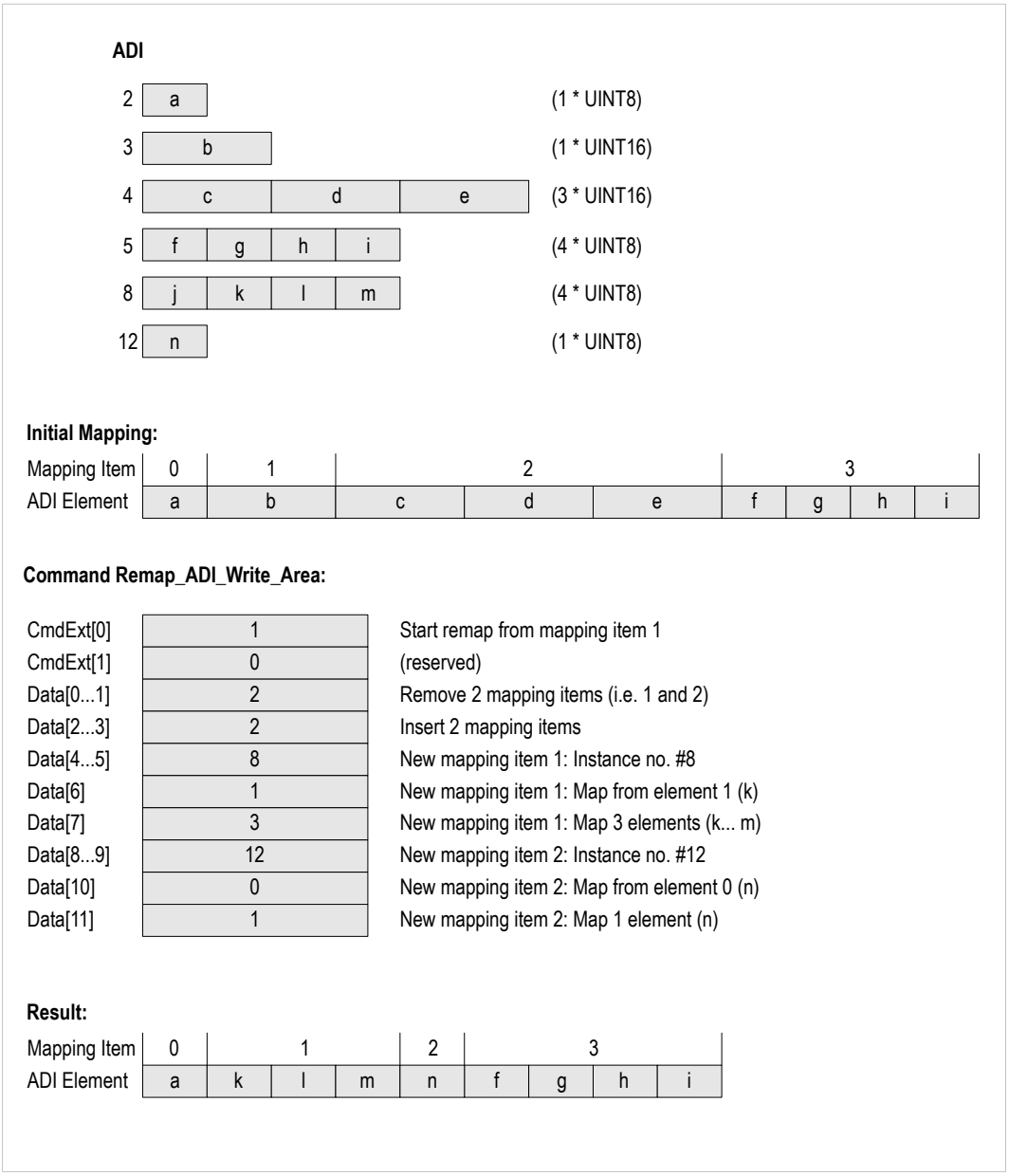

Fig. 32

G CRCの計算（16ビット）

G.1 概要

以下の情報は、シリアルインターフェース使用時にのみ当てはまります。

受信側で送信エラーを検出できるように、シリアルテレグラムの各フレームには16ビットの巡回冗長検査（CRC）が格納されています。

CRCは以下のように計算します。

1. 16ビットレジスタにFFFFhを読み込みます。（説明を簡単にするために、これをCRCレジスタと呼ぶことにします）
2. メッセージの最初のバイトとCRCレジスタの下位バイトとの間でXOR演算を行い、その結果をCRCレジスタに格納します。
3. CRCレジスタを右（LSBの方向に）に1ビットシフトし、MSBに0を埋めます。
4. シフトによってレジスタから押し出されたLSBをチェックします。 セットされている場合、CRCレジスタと多項式値であるA001h（1010 0000 0000 0001）との間でXOR演算を行います。
5. 8ビット分シフトするまで手順3と4を繰り返します。
6. メッセージの次のバイトとCRCレジスタの下位バイトとの間でXOR演算を行い、その結果をCRCレジスタに格納します。
7. メッセージ全体が処理されるまで手順3～6を繰り返します。
8. CRCレジスタに最終的なCRC16の値が格納されます。

G.2 例

CRC計算アルゴリズムを実装するとき、以下の一連の例を使用して、実装したアルゴリズムの計算結果がAnybus CompactComモジュールと同じになることを確認してください。

配列が`{ 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08 }`の場合、CRC16は次のようになります。`{ 0xb0, 0xcf }`.

配列が`{ 0x00, 0x55, 0xAA, 0xFF, 0x0F, 0x5A, 0xA5, 0xF0 }`の場合、CRC16は次のようになります。`{ 0x11 , 0x03 }`.

配列が`{ 0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80 }`の場合、CRC16は次のようになります。`{ 0x77 , 0x28 }`.

G.3 コード例

この例では、CRCを高速に計算する手法を使用します。すなわち、CRCが取り得るすべての値を2つの配列にあらかじめ読み込んでおき、関数がメッセージバッファの次の値を取り出すたびに、単純にインデックスを作成していきます。

```
const UINT8 abCrc16Hi[] =
{
   0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80,
   0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1,
   0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01,
   0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40,
   0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80,
   0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
   0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01,
   0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
   0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81,
   0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1,
   0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00,
   0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
   0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80,
   0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
   0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01,
   0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
   0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81,
   0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
   0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01,
   0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40,
   0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81,
   0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1,
   0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00,
   0xC1, 0x81, 0x40
};
const UINT8 abCrc16Lo[] =
{
   0x00, 0xC0, 0xC1, 0x01, 0xC3, 0x03, 0x02, 0xC2, 0xC6, 0x06, 0x07,
   0xC7, 0x05, 0xC5, 0xC4, 0x04, 0xCC, 0x0C, 0x0D, 0xCD, 0x0F, 0xCF,
   0xCE, 0x0E, 0x0A, 0xCA, 0xCB, 0x0B, 0xC9, 0x09, 0x08, 0xC8, 0xD8,
   0x18, 0x19, 0xD9, 0x1B, 0xDB, 0xDA, 0x1A, 0x1E, 0xDE, 0xDF, 0x1F,
   0xDD, 0x1D, 0x1C, 0xDC, 0x14, 0xD4, 0xD5, 0x15, 0xD7, 0x17, 0x16,
   0xD6, 0xD2, 0x12, 0x13, 0xD3, 0x11, 0xD1, 0xD0, 0x10, 0xF0, 0x30,
   0x31, 0xF1, 0x33, 0xF3, 0xF2, 0x32, 0x36, 0xF6, 0xF7, 0x37, 0xF5,
   0x35, 0x34, 0xF4, 0x3C, 0xFC, 0xFD, 0x3D, 0xFF, 0x3F, 0x3E, 0xFE,
   0xFA, 0x3A, 0x3B, 0xFB, 0x39, 0xF9, 0xF8, 0x38, 0x28, 0xE8, 0xE9,
   0x29, 0xEB, 0x2B, 0x2A, 0xEA, 0xEE, 0x2E, 0x2F, 0xEF, 0x2D, 0xED,
   0xEC, 0x2C, 0xE4, 0x24, 0x25, 0xE5, 0x27, 0xE7, 0xE6, 0x26, 0x22,
   0xE2, 0xE3, 0x23, 0xE1, 0x21, 0x20, 0xE0, 0xA0, 0x60, 0x61, 0xA1,
   0x63, 0xA3, 0xA2, 0x62, 0x66, 0xA6, 0xA7, 0x67, 0xA5, 0x65, 0x64,
   0xA4, 0x6C, 0xAC, 0xAD, 0x6D, 0xAF, 0x6F, 0x6E, 0xAE, 0xAA, 0x6A,
   0x6B, 0xAB, 0x69, 0xA9, 0xA8, 0x68, 0x78, 0xB8, 0xB9, 0x79, 0xBB,
   0x7B, 0x7A, 0xBA, 0xBE, 0x7E, 0x7F, 0xBF, 0x7D, 0xBD, 0xBC, 0x7C,
   0xB4, 0x74, 0x75, 0xB5, 0x77, 0xB7, 0xB6, 0x76, 0x72, 0xB2, 0xB3,
   0x73, 0xB1, 0x71, 0x70, 0xB0, 0x50, 0x90, 0x91, 0x51, 0x93, 0x53,
   0x52, 0x92, 0x96, 0x56, 0x57, 0x97, 0x55, 0x95, 0x94, 0x54, 0x9C,
   0x5C, 0x5D, 0x9D, 0x5F, 0x9F, 0x9E, 0x5E, 0x5A, 0x9A, 0x9B, 0x5B,
   0x99, 0x59, 0x58, 0x98, 0x88, 0x48, 0x49, 0x89, 0x4B, 0x8B, 0x8A,
   0x4A, 0x4E, 0x8E, 0x8F, 0x4F, 0x8D, 0x4D, 0x4C, 0x8C, 0x44, 0x84,
   0x85, 0x45, 0x87, 0x47, 0x46, 0x86, 0x82, 0x42, 0x43, 0x83, 0x41,
   0x81, 0x80, 0x40
};

UINT16 CRC_Crc16( UINT8* pbBufferStart, UINT16 iLength )
```

```
{
   UINT8   bIndex;
   UINT8   bCrcLo;
   UINT8   bCrcHi;
   bCrcLo = 0xFF;
   bCrcHi = 0xFF;
   while( iLength > 0 )
   {
      bIndex = bCrcLo ^ *pbBufferStart++;
      bCrcLo = bCrcHi ^ abCrc16Hi[ bIndex ];
      bCrcHi = abCrc16Lo[ bIndex ];
      iLength--;
   }
   return( bCrcHi << 8 | bCrcLo );
}
```

H CRCの計算（32ビット）

以下の例は、SPIモードでのCRC計算アルゴリズム実装方法を示します。 SPIとCRC計算アルゴリズムはバイトオーダーおよびビットオーダーが異なるため、ビットオーダーとバイトオーダーを変更する必要があります。

H.1 例

CRC計算アルゴリズムを実装するとき、以下の一連の例を使用して、実装したアルゴリズムの計算結果がAnybus CompactComモジュールと同じになることを確認してください。

配列が`{ 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08 }`の場合、CRC32は次のようになります。 `{ 0xeb 0xf4 0x72 0x27 }`.

配列が`{ 0x00, 0x55, 0xAA, 0xFF, 0x0F, 0x5A, 0xA5, 0xF0 }`の場合、CRC32は次のようになります。 `{ 0xbe 0xa7 0x3a 0x2d }`.

配列が`{ 0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80 }`の場合、CRC32は次のようになります。 `{ 0x9a 0xf6 0x4b 0x49 }`.

H.2 コード例

この例では、CRCを高速に計算する手法を使用します。

```
const UINT8 abBitReverseTable16[] = { 0x0, 0x8, 0x4, 0xC, 0x2, 0xA,
      0x6, 0xE, 0x1, 0x9, 0x5, 0xD, 0x3, 0xB, 0x7, 0xF };
const UINT32 crc_table32[] = {
     0x4DBDF21CUL, 0x500AE278UL, 0x76D3D2D4UL, 0x6B64C2B0UL,
     0x3B61B38CUL, 0x26D6A3E8UL, 0x000F9344UL, 0x1DB88320UL,
     0xA005713CUL, 0xBDB26158UL, 0x9B6B51F4UL, 0x86DC4190UL,
     0xD6D930ACUL, 0xCB6E20C8UL, 0xEDB71064UL, 0xF0000000UL
};
UINT32 CRC_Crc32( UINT8* pbBufferStart, UINT16 iLength )
{
   UINT8 bCrcReverseByte;
   UINT16 i;
   UINT32 lCrc = 0x0;
   for(i = 0; i < iLength; i++)
   {
      bCrcReverseByte =
         lCrc ^ abBitReverseTable16[ (*pbBufferStart >> 4 ) & 0xf ];
      lCrc = (lCrc >> 4) ^ crc_table32[ bCrcReverseByte & 0xf ];
      bCrcReverseByte =
         lCrc ^ abBitReverseTable16[ *pbBufferStart  & 0xf ];
      lCrc = (lCrc >> 4) ^ crc_table32[ bCrcReverseByte & 0xf ];
      pbBufferStart++;
   }

   lCrc = ((UINT32)abBitReverseTable16 [ (lCrc & 0x000000F0UL) >> 4 ] )|
      ((UINT32)abBitReverseTable16 [ (lCrc & 0x0000000FUL) ] ) << 4 |
      ((UINT32)abBitReverseTable16 [ (lCrc & 0x0000000FUL) ] ) << 4 |
      ((UINT32)abBitReverseTable16 [ (lCrc & 0x0000F000UL ) >> 12 ] << 8)|
      ((UINT32)abBitReverseTable16 [ (lCrc & 0x00000F00UL ) >> 8 ] << 12)|
      ((UINT32)abBitReverseTable16 [ (lCrc & 0x00F00000UL ) >> 20 ]<< 16)|
      ((UINT32)abBitReverseTable16 [ (lCrc & 0x000F0000UL ) >> 16 ]<< 20)|
      ((UINT32)abBitReverseTable16 [ (lCrc & 0xF0000000UL ) >> 28 ]<< 24) |
      ((UINT32)abBitReverseTable16 [ (lCrc & 0x0F000000UL ) >> 24 ]<< 28);
   return lCrc;
}
```

I タイミングと性能

I.1 概要

この章では、Anybus CompactCom 40ファミリーの各製品について検証のうえ文書化されているタイミングと性能に関するパラメータについて説明します。

以下の項目に関するタイミングが測定されています（詳細は後述）。

カテゴリ	パラメータ
起動時の遅延	T1, T2
NW_INITの処理	T100
イベントベースのWrMsgビジー時間	T103
イベントベースのプロセスデータ遅延時間	T101, T102

> **!** 本ドキュメント作成時点では、すべてのネットワークに関するネットワーク固有のタイミング仕様は公開されていません。 それらの情報は、利用可能になった時点で各ネットワークガイドに随時追加します。

I.2 内部タイミング

I.2.1 起動時の遅延

以下のパラメータは、/RESETが解放されてから指定のイベントが発生するまでの時間として定義されています。

パラメータ	説明	最大	単位
T1	Anybusが最初のアプリケーション割り込みを生成（パラレルモード）。	1.5	s
T2	Anybusが最初のアプリケーションテレグラムを受信/処理可能となった（シリアルモード）。	1.5	s

I.2.2 NW_INITの処理

AnybusモジュールがNW_INIT状態にて行うべき動作に要する時間は、ネットワークによって大きく異なります。 さらに、この状態においてホストアプリケーションに発行されるコマンドの数は、ネットワークの違いだけでなく、実装の違い（例えば、実際のプロセスデータの実装など）によっても異なる可能性があります。 これはすなわち、ホストアプリケーションの応答時間が、このパラメータにも大きな影響を与えることを意味しています。 そのため、ホストアプリケーションの標準的な実装に加え、Anybusのすべてのバージョンに適用できるようにするには、その最大値を定義することしかできません。

このパラメータを定義することは、そのパラメータが要求を満足していることの監視をホストアプリケーションに求めること（または、期待すること）を意味するものではありません。つまり、プロトコルが実行され、正しい状態が示されている事実をもって、Anybusモジュールの健全性が十分示されているものとします。 ただし、この点に関するAnybusのコンセプトを信頼できない場合、ホストアプリケーションは、タイムアウト時間が経過するのを待ってから、正常に機能していないことをエンドユーザーに対して通知することが可能です。 結局これは起動フェーズの話であるため、タイムアウトを使用する場合は比較的長いタイムアウト値を使用すれば十分です。

パラメータ	条件
ネットワーク固有のコマンドの数。	最大
各方向における、プロセスデータにマッピングされたADI（UINT8 × 1）の数。	32、またはネットワーク固有の最大数がそれより小さい場合は、その最大数。
イベントベースで処理する場合の、アプリケーションのメッセージ応答時間。	> 1 ms
ピンポン方式で処理する場合の、アプリケーションの応答時間。	> 10 ms
アプリケーションが同時に処理可能な、Anybusの未処理コマンドの数。	1

パラメータ	説明	通信	最大	単位
T100	NW_INITの処理	イベントベースのすべてのモード クイックコネクトをサポートするモジュールに必要 他のすべてのモジュールで推奨。	100	ms
		シリアル19.2 kbps	30	s
		（他のすべてのモード）	10	s

I.2.3 イベントベースのWrMsgビジー時間

アプリケーションがメッセージをポストしてから、モジュールがH_WRMSG領域をアプリケーションに返すまでに要する時間として、イベントベースのWrMsgビジー時間が定義されています。

パラメータ	説明	最大	単位
T103	H_WRMSG領域のビジー時間	500	µs

I.2.4 イベントベースのプロセスデータ遅延時間

「Read process data delay」は、ネットワークフレームの最終ビットがモジュールに取り込まれてから、アプリケーションに対してRDPDI割り込みをアサートするまでの時間として定義されます。

「Write process data delay」は、ネットワークの種類に応じて2つの方法で定義されます。

- ソフトウェアスタックベースのサイクリック/ポーリングネットワークの場合、モジュールがライトプロセスデータバッファを交換してから、新規プロセスデータフレームの最初のビットがネットワークに送出されるまでの時間として定義されます。
- COS（状態遷移）ネットワークの場合、アプリケーションがライトプロセスデータバッファを交換してから、新規プロセスデータフレームの最初のビットがネットワークに送出されるまでの時間として定義されます。

古いピンポン方式の性能要件との互換性を保つため、32バイトのプロセスデータに対して最大500 µsの遅延時間が定義されていますが、高性能の同期イベントベースのモジュールでは、32バイトのプロセスデータに対して15 µsを超える遅延が生じることはありません。

パラメータ	説明	**32**バイトのプロセスデータに対する推奨最大値	**32**バイトのプロセスデータに対する絶対最大値	単位
T101	リードプロセスデータ遅延時間	15	500	µs
T102	ライトプロセスデータ遅延時間	15	500	µs

本ドキュメント作成時点では、すべてのネットワークに関するネットワーク固有のタイミング仕様は公開されていません。 それらの情報は、利用可能になった時点で各ネットワークガイドに随時追加します。

J 後方互換性

産業用ネットワークモジュールのAnybus CompactCom 40シリーズは、Anybus CompactCom 30シリーズよりも性能が大きく向上しており、機能も追加されています。 40シリーズは30シリーズに対する後方互換性があり、30シリーズ用に開発されたアプリケーションは、大幅な変更を施さずに40シリーズでも使用することが可能です。 また、同じアプリケーションで30シリーズと40シリーズのモジュールを混在させることも可能です。

ここでは、1つのアプリケーションを両シリーズに適用できるように設計する場合や、30シリーズ用のアプリケーションを40シリーズ用に適応させる場合にAnybus CompactCom 40について考慮すべき後方互換性の問題について説明します。 後方互換性に関するネットワーク固有の情報については、Anybus CompactCom 40ネットワークガイドを参照してください。

J.1 初期段階における考慮事項

Anybus CompactCom30シリーズのモジュール用に開発されたホストアプリケーションを、40シリーズのモジュールとも互換性を持つように変更する作業を開始する際には、次の2点を考慮する必要があります。

- できる限り少ない作業で、すなわち可能な限り現在の設計を再利用して、実装を追加します。
  - これが最も迅速で簡単なソリューションですが、40シリーズで利用可能な多くの新機能（高速通信インターフェース、拡大されたメモリ領域、高速通信プロトコルなど）を有効化できないという欠点があります。
  - ホストアプリケーションが40シリーズのモジュールと互換性があることを確認するには、以下のハードウェアとソフトウェアの相違を確認する必要があります。 現在の設計に小規模な変更が必要になる場合があります。
- 再設計を行って、40シリーズに用意されている新機能すべてが利用できるようにします。
  - ホストアプリケーションと40シリーズ間の新しい操作手順をサポートするために、新しいドライバーとホストアプリケーションのサンプルコードがwww.anybus.com/starterkit40で入手できます。このドライバーは、30シリーズと40シリーズの両方のモジュールをサポートします。
  - ホストアプリケーションが40シリーズのモジュールと互換性があることを確認するには、以下のハードウェアの相違を確認する必要があります。

本章では、*30*シリーズと*40*シリーズとの相違についてのみ扱います。 *Anybus CompactCom 40*シリーズの新規および拡張された機能の説明については、すべてのドキュメントを確認できるサポートページを参照してください。

サポートページへのリンク： www.anybus.com/support.

J.2 ハードウェアの互換性

Anybus CompactComは3つのハードウェア形態、モジュール、チップ、およびブリックで利用できます。

J.2.1 モジュール

30シリーズおよび40シリーズのモジュールは、寸法、外形、コネクタ、LEDインジケータ、取付部品などの物理的特性を共有しています。また、両シリーズ共に、ハウジングなしモジュールも使用可能です。

Fig. 33 Anybus CompactCom M30/M40

J.2.2 チップ

Anybus CompactComのチップ（C30/C40）は、バージョン間で物理的な寸法が完全に異なります。

> ! ハードウェアを大幅にアップデートしない限り、チップソリューションを30シリーズから40シリーズへ移行する方法はありません。

J.2.3 ブリック

Anybus CompactCom B40-1は、Anybus CompactCom B30と寸法を共有していません。 そのため、B40-1は移行に適していません。 しかし、HMS Industrial Networks ABは、移行に使用可能な40シリーズ用ブリックバージョンを別途開発しました。 同製品B40-2は、B30と寸法などを共有しています。 Anybus CompactCom B40-2の詳細についてはHMS Industrial Networks ABまでお問い合わせください。

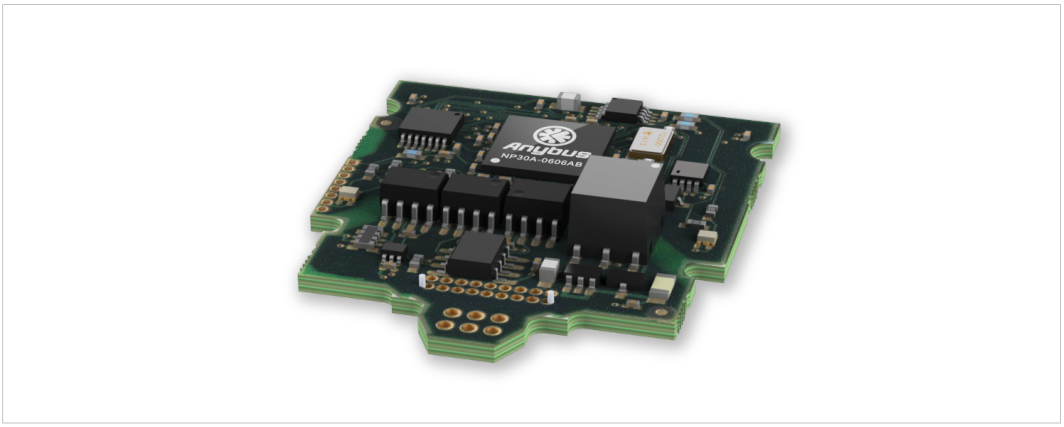

Fig. 34 Anybus CompactCom B30

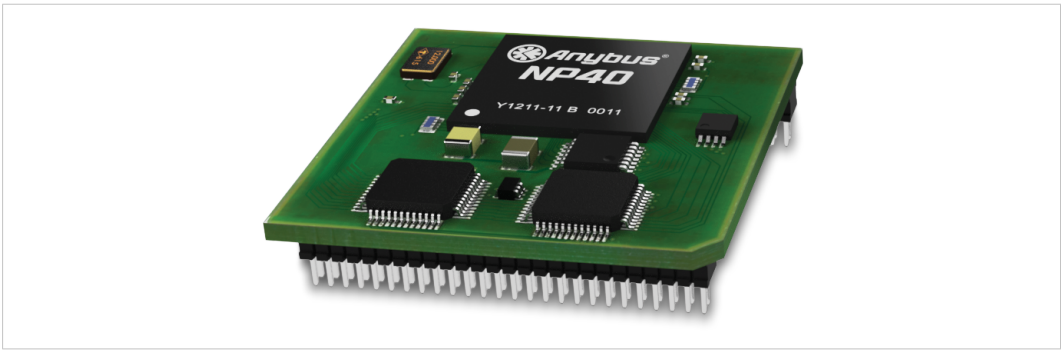

Fig. 35 Anybus CompactCom B40 – 1（移行用ではありません）

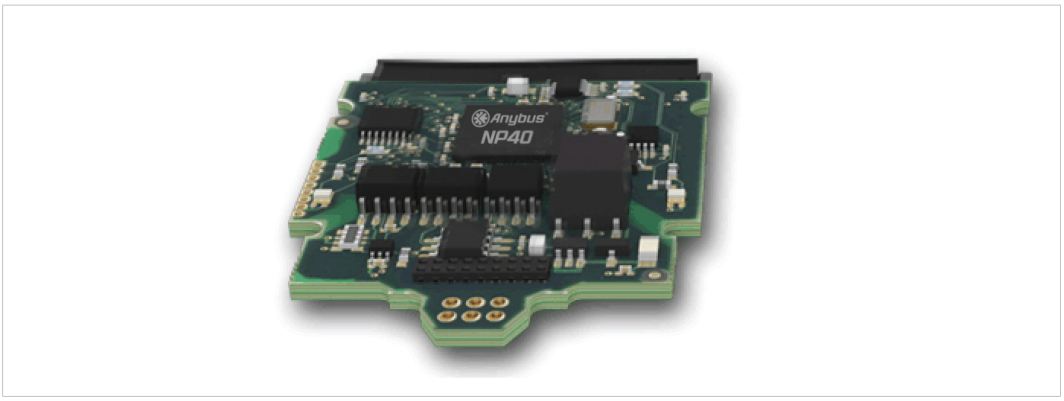

Fig. 36 Anybus CompactCom B40–2

J.2.4 ホストアプリケーションインターフェース

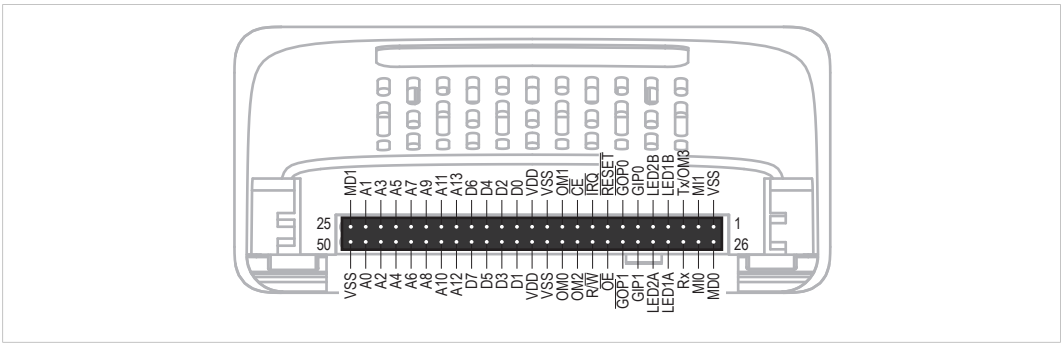

Fig. 37

ホストアプリケーションインターフェースの信号の一部は機能性および/または機能が変更になっているため、これらの互換性をチェックする必要があります。 以下のセクションを参照してください。

Tx/OM3

このピンは、30シリーズではTxのみです。 電源投入時は3ステートで、初期化後はAnybus CompactCom UARTによって駆動されます。 40シリーズでは、このピンは第4の動作モード設定ピン（OM3）として使用されます。 リセット解除後の起動時にこのピンが読み取られて、使用する動作モードが決定されます。 その後、ピンはTx出力に変更されます。

40シリーズでは、弱いプルアップ抵抗がこのピンに内蔵されています。 30シリーズのモジュールまたはブリック上でこのピンが接続されていなかったり、Highにプルアップされていたり、ホストプロセッサーのHigh-Zデジタル入力に接続されている場合、40シリーズとの互換性を保ちます。 外部プルアップ抵抗は推奨されますが、必須ではありません。

> **!** このピンが起動中にホストによってLowへ引き下げられた場合は、40シリーズモジュールまたはブリックは期待されている動作モードになりません。

関連情報： Anybus CompactCom M40 Hardware Design Guide (HMSI-216-126)、「Application Connector Pin Overview」

モジュールの識別（**MI[0~1]**）

これらのピンは、どの種類のAnybus CompactComが搭載されているかを特定するために、ホストアプリケーション（お使いの製品）によって使用されます。 40シリーズと30シリーズでは、この識別信号が異なります。

> **(i)** ソフトウェアでこの識別信号を使用する場合は、新しい識別値を処理する必要があります。

MI1	**MI0**	モジュールの種類
LOW	LOW	Active Anybus CompactCom 30
HIGH	LOW	Active Anybus CompactCom 40

アプリケーションによるMI[0~1]のサンプリングは、起動からSETUP状態終了までの間のみです。 起動時およびリセット解除前のピンはLowです。

関連情報： Anybus CompactCom M40 Hardware Design Guide (HMSI-216-126)、「Settings/Sync」。

GIP[0~1]/LED3[A~B]

これらのピンは、30シリーズではデフォルトで3ステート入力となっています。 40シリーズでは、NW_INIT状態まで3ステートです。 その後はオープンドレイン方式のActive Low LED出力（LED3A/LED3B）となります。

現在の設計が以下の条件を満たしている場合、ハードウェアの変更は必要ありません。

- これらのピンがGNDに接続されている
- ピンのプルアップが行われている
- ピンのプルダウンが行われている
- ピンが未接続のままである

ただし、アプリケーションがピンをHighにすると、短絡が発生します。

ピンをLEDに接続する場合はプルアップ抵抗が必要です。

40シリーズでは、Anybusオブジェクト（01h）のアトリビュート#16（GPIO構成）を使用して、GIP[0~1]とGOP[0~1]をハイインピーダンスの状態（3ステート）に設定することが可能です。 つまり、ホストアプリケーションのハードウェアを変更できない場合、このアトリビュートを使用して、NW_INIT状態を離れる前にGIPとGOPがハイインピーダンス状態になるように設定できます。

関連情報： Anybus CompactCom M40 Hardware Design Guide (HMSI-216-126)、「LED Interface/D8-D15 (Data Bus)」

GOP[0~1]/LED4[A~B]

これらのピンは、30シリーズではデフォルトにより出力（High状態）になっています。 40シリーズではNW_INIT状態まで3ステートで、その後、プッシュプル方式のActive Low LED出力（LED4A/LED4B）となります。

この変化はお使いの製品に影響しません。

関連情報： Anybus CompactCom M40 Hardware Design Guide (HMSI-216-126)、3.2.3, LED Interface/D8-D15 (Data Bus)

アドレスピン**A[11~13]**

アドレスピン11、12、13は30シリーズでは無視されます。 後方互換性のある8ビットパラレルモードで40シリーズモジュールにアクセスする場合、これらのピンはHighでなければなりません。 これらのピンが未接続になっている場合やGNDに接続されている場合は、ハードウェアの変更を行って、これらをHighにする必要があります。

最大入力信号レベル（V_{IH}）

30シリーズの最大入力信号レベルは$V_{IH}=V_{DD}+0.2$ V、40シリーズでは$V_{IH}=3.45$ Vと指定されています。論理Highレベルで3.45Vを超えないようにしてください。

J.3 ソフトウェア全般

J.3.1 拡張されたメモリ領域

40シリーズではメモリ領域が拡張されており、より大きいサイズのプロセスデータ（以前の最大256バイトに代わり最大4096バイト）およびメッセージデータ（以前の最大255バイトに代わり最大1524バイト）にアクセスできるようになりました。 30シリーズには、アプリケーションでは使用できない、予約されたメモリ領域があります。 40シリーズは、これらのメモリ領域の一部に新機能を実装しています。

拡張されたメモリ領域を使用するには、本章には記載のない新しい操作手順を実装する必要があります。

メモリ領域が特定のネットワークでサポートされていない場合、そのメモリ領域は使用できません。 メモリの読み取り/書き込みテストなどの目的で、これらの領域にアクセスしないようにしてください。

関連情報： Anybus CompactCom 40 Software Design Guide (HMSI-216-125)、「Memory Map」

J.3.2 より高速なピンポンプロトコル

40シリーズでは、ピンポンプロトコル（30シリーズで使用されているプロトコル）が高速化されています。 30シリーズのモジュールは、通常10～100 µs内に「ping」に応答します。 40シリーズは、通常2 µs内に「ping」に応答します。

割り込み駆動型のアプリケーション（パラレル動作モード）では、速度向上によりCPU負荷が増大する可能性があります。

J.3.3 起動時のCompactComからホストアプリケーションへの要求

ホストアプリケーション内のソフトウェアオブジェクトに対する要求はすべて、（オブジェクトが存在しない場合でも）処理と応答が行われる必要があります。 これは、30シリーズと40シリーズの両方に適用されます。 40シリーズには、新機能のための追加オブジェクトが導入されています。

また、40シリーズによって既存のオブジェクトにコマンドが追加された場合も、（たとえサポートされていなくても）応答が必要です。

処理不可能であってもすべてのコマンドに応答するという、しかるべき動作をする実装をお使いの場合は、何も変更する必要はありません。

J.3.4 Anybusオブジェクト (01h)

アトリビュート	30シリーズ	40シリーズ	変更/アクション/コメント
#1, Module Type	0401h	0403h	ホストアプリケーションが、40シリーズの新しいモジュールタイプの値を受け入れることを確認してください。
#15, Auxiliary Bit	利用可能	削除されました	40シリーズでは「Changed Data Indication」を無効にできません。 また、以下の「Control Register CTRL_AUX-bit」および「Status Register STAT_AUX-bit」も参照してください。
#16, GPIO Configuration	デフォルト: 一般的な入出力ピン	デフォルト: LED3およびLED4出力	以下も参照してください。 • *GIP[0~1]/LED3[A~B], ページ 168* • *GOP[0~1]/LED4[A~B], ページ 168*

J.3.5 コントロースレジスタCTRL_AUXビット

30シリーズ　現在のテレグラムのプロセスデータが以前のものと比べて変化していることをコントロールレジスタのCTRL_AUXビットによってAnybus CompactComが示します。

40シリーズ　CTRL_AUX ビットの値は常に無視されます。 プロセスデータは常に受け入れられます。

HMSからリリースされたすべてのAnybus CompactCom 30サンプルドライバーが、この相違に従っています。

関連情報： Anybus CompactCom 40 Software Design Guide (HMSI-216-125)、「Control Register」。

J.3.6 ステータスレジスタSTAT_AUXビット

30シリーズ　現在のテレグラムの出力プロセスデータが以前のものと比べて変化していることをステータスレジスタのSTAT_AUXビットによって示します。 この機能はAnybusオブジェクト（01h）、アトリビュート#15で有効にする必要があります。デフォルトでは、STAT_AUX ビット機能は無効になっています。

40シリーズ　以前のテレグラムに対して、ネットワークから送られた出力プロセスデータが更新されていることを（必ずしもデータが変更されている必要はありません）STAT_AUXビットが示します。 この機能は常に有効になっています。

HMSからリリースされたすべてのAnybus CompactCom 30サンプルドライバーが、この相違に従っています。

関連情報： Anybus CompactCom 40 Software Design Guide (HMSI-216-125)、「Status Register」。

J.3.7 コントロースレジスタCTRL_Rビット

30シリーズ　このビットは、アプリケーションによっていつでも変更される可能性があります。

40シリーズ　8ビットパラレル動作モードでは、STAT_Mビットがステータスレジスタにセットされるとき、このビットは1から0への遷移しか許可されません。 シリアル動作モードを使用している場合は、最終を示す空フラグメント直後のテレグラム内で、1から0への遷移も許可されます。

HMSからリリースされたすべてのCompactCom 30サンプルドライバーが、この相違に従っています。

関連情報： Anybus CompactCom 40 Software Design Guide (HMSI-216-125)、「Control Register」。

J.3.8 ステータスレジスタ、プロセスデータリード領域、メッセージデータリード領域の変更

40シリーズでは、ステータスレジスタ、プロセスデータリード領域、およびメッセージデータリード領域は、ハードウェア（パラレルインターフェース）内で書き込み保護されています。 何らかの理由でソフトウェアがこれらの領域に書き込みを行う場合は、変更が必要です。

HMSからリリースされたすべてのAnybus CompactCom 30サンプルドライバーが、この相違に従っています。

このページは意図的に空白になっています

Box 4126
300 04 Halmstad, Sweden

info@hms.se