

Tutorial

Anybus[®] CompactCom Implementation

Doc.Id. JCM-1201-011

Rev. 1.00



HALMSTAD • CHICAGO • KARLSRUHE • TOKYO • BEIJING • MILANO • MULHOUSE • COVENTRY • PUNE • COPENHAGEN

HMS Industrial Networks
Mailing address: Box 4126, 300 04 Halmstad, Sweden
Visiting address: Stationsgatan 37, Halmstad, Sweden

E-mail: info@hms-networks.com
Web: www.anybus.com

重要なユーザ情報

このマニュアルは、Anybus CompactCom を使用したアプリケーションの開発方法について十分にご理解いただくことを意図しています。このマニュアルでは、Anybus CompactCom ファミリの全ての製品に共通する機能や動作について説明します。特定の Anybus CompactCom モジュールに関する情報については、"Anybus CompactCom Network Interface Appendix" を参照してください。

このマニュアルの読者は、ハイレベルなソフトウェア設計および一般的な通信システムに詳しいことが前提とされています。このマニュアルでは、Anybus CompactCom モジュールを使用したシンプルなアプリケーションについて説明します。さらに高度なアプリケーションについては、"Anybus CompactCom Software Design Guide" および各 Network Interface Appendix を参照してください。

責任

このマニュアルはあらゆる点を考慮して作成しています。不正確な記述や記載漏れがあった場合には、HMS Industrial Networks AB までご連絡ください。このマニュアルに含まれるデータや説明には拘束力はありません。HMS Industrial Networks AB は、継続的な製品開発を旨とする当社のポリシーに則って、弊社の製品を改良する権利を留保します。

このマニュアルに含まれる情報は予告なく変更される場合があります。ただし、HMS Industrial Networks AB は変更に関して義務を負うものではありません。HMS Industrial Networks AB はこのマニュアルに現れるあらゆるエラーに対して責任を負いません。

この製品には多くのアプリケーションがあります。この装置の使用責任者は、アプリケーションが該当する法律、規則、規定、および規格を含む全ての性能および安全要求事項を満たしており、これを確認するために全ての必要な手順がとられたことを保証する必要があります。

HMS Industrial Networks AB は、いかなる状況においても、文書化されていない機能の使用、タイミング、またはこの製品の文書化された範囲外で見つかった機能面での副次的な影響によって発生する可能性がある問題に対する義務または責任を負いません。製品のこのような側面の直接的または間接的な使用によって発生する結果は不明確であり、互換性の問題や安定性の問題などを含む可能性があります。

このマニュアルに含まれる例および図表は、説明のためにのみ記載されています。特定の実装には多くの要素や要件が関連しているため、HMS Industrial Networks AB は、これらの例や図表に基づいた実際の使用に対する責任を負いません。

知的財産権

HMS Industrial Networks AB は、このマニュアルに記載された製品に組み入れられた技術に関する知的財産権を所有します。これらの知的財産権には、米国およびその他の国での特許および出願中の特許が含まれる可能性があります。

商標について

Anybus[®] は、HMS Industrial Networks AB の登録商標です。その他の全ての商標は、各所有者の資産です。

警告 :	これはクラス A 製品です。国内の環境では、この製品は無線妨害を発生させる可能性があります。この場合、ユーザは適切な対策をとる必要があります。
ESD に関する注意 :	この製品は ESD (Electrostatic Discharge : 静電気放電) に敏感な部分が含まれているため、ESD 対策が十分でない場合には破損する可能性があります。製品を直接手で扱うときは静電気対策が必要です。これらを行わないと製品を破損させる可能性があります。

Anybus CompactCom Tutorial

Rev 1.00

Copyright© HMS Industrial Networks AB

Apr 2010 Doc Id JCM-1201-011

目次

はじめに	このマニュアルについて	1
	関連マニュアル	1
	マニュアル更新履歴	1
	慣例と用語集	2
	定義	2
第1章	このマニュアルの使い方	3
第2章	Anybus CompactCom について	4
第3章	チュートリアル.....	5
	はじめに	5
	Anybus CompactCom モジュールの接続	6
	ホスト・インターフェース信号	6
	操作モードの設定	8
	テレグラム	9
	パラレル・インターフェース・モード	10
	最初のハンドシェーク	10
	DPRAM	11
	テレグラムの送信：パラレル通信	13
	シリアル・インターフェース・モード	18
	最初のハンドシェーク	18
	シリアル・テレグラム・フレーム	19
	テレグラムの送信：シリアル通信	21
	セットアップの続き	28
	Anybus CompactCom ステート・マシン	28
	Anybus CompactCom へのアクセス	29
	ADI のマッピング	30
	セットアップの完了	32
	ネットワークの初期化	33
	さらなるコンフィグレーションと認証	34

第 4 章	リソース	35
	機能のカテゴリ化	35
	基本	35
	拡張	35
	高度	35
	設計ガイド	36
	<i>Anybus CompactCom Software Design Guide</i>	36
	<i>Anybus CompactCom Hardware Design Guide</i>	36
	Network Interface Appendix	36
	ドライバ	36
	スタータ・キット	36
	ドライブ・プロファイル	37
アペンディックス A	トレース、DeviceNet.....	38
アペンディックス B	トレース、Profibus DP-V1.....	44
サポート	サポート	50

P. このマニュアルについて

詳細な情報や資料などについては、HMS のウェブサイト（www.anybus.com）を参照してください。

P.1 関連マニュアル

チュートリアル の欄外にこれらの文書を示し、右の欄外の枠内に文書名の略語を示します。全ての文書は www.anybus.com で入手できます。

マニュアル名	略語	マニュアル ID	作成者
Anybus CompactCom Software Design Guide	SWDG	SCM-1200-037	HMS
Anybus CompactCom Hardware Design Guide	HWDG	SCM-1200-061	HMS
Anybus CompactCom Standard Driver Implementation Guide	SDRV	SCM-1200-043	HMS
Anybus CompactCom Lite Driver Implementation Guide	LDRV	SCM-1200-042	HMS
Anybus CompactCom Network Interface Appendix	" ネットワー ク " APP.	www.anybus.com を参 照	HMS

P.2 マニュアル更新履歴

最新の更新（ ... 1.00）

変更内容	ページ
-	-
-	-
-	-

改定版リスト

改定番号	改定日	作成者	章	説明
1.00	2010-04-19	KeL	-	最初の公式リリース

P.3 慣例と用語集

このマニュアルでは下記の慣例を使用しています。

- 番号が付いたリストは、連続した手順を示します。
- 黒丸が付いたリストは、手順ではなく情報を示します。
- 'Anybus' または 'モジュール' という用語は、Anybus CompactCom モジュールを意味します。
- 'ホスト' または 'ホスト・アプリケーション' という用語は、Anybus モジュールのホストとして機能する装置を意味します。
- 16 進値は、通常では NNNNh というフォーマットで書かれます。NNNN は 16 進法の値です。16 進値が例のコードまたは疑似コードに存在する場合、その値は 0xnnnn というフォーマット (nnnn は 16 進法の値) で書かれます。

P.3.1 定義

用語	説明
プロセス・データ	高速サイクリック・ネットワーク I/O データ
アサイクリック・データ	要求に応じてアップデートされるデータ (アプリケーション・パラメータなど)
ADI	アプリケーション・データ・インスタンス。ADI はアサイクリック・データのアップデートまたはプロセス・データのマッピングのためにアクセス可能なアプリケーション変数で、まさにアプリケーションによって定義されるものです。
ABCC	Anybus CompactCom
オブジェクト・モデル	29 ページの "Anybus CompactCom へのアクセス" を参照
オブジェクト	
インスタンス	
アトリビュート	
テレグラム	9 ページの "テレグラム" を参照
メッセージ	

1. このマニュアルの使い方

このチュートリアルは、簡単な適用例について説明することによって、ユーザが Anybus CompactCom のコンセプトおよびモジュールの設定方法を理解できるように手助けすることです。

マニュアルをいつ読んで使用するか

Anybus CompactCom を使用してアプリケーションを開発することを決定したかどうかにかかわらず、モジュールとの通信方法に関する最初の導入として、このチュートリアルを読むことができます。チュートリアルでは、いくつかの例を挙げて、モジュールの接続方法および通信の設定方法を説明します。モジュールの機能をより深く理解するために、" 実際に使用する " ことをお奨めします。

このチュートリアルには、ネットワーク特有の問題は含まれていません。各 Network Interface Appendix には短いチュートリアルが含まれており、製品の認定のために必要なエレメントについて、アプリケーションを準備するために役立ちます。

Anybus CompactCom のコンセプトは、Anybus CompactCom Software Design Guide (SWDG) で詳しく説明しています。

2. Anybus CompactCom について

背景

今日、産業用の通信に対する規格は1つではありません。いくつかの異なる産業用ネットワークが存在し、ネットワークに接続されたデバイスと情報を交換するために異なるプロトコルを使用しています。産業用イーサネットおよびフィールドバス通信が広範囲で使用され、細分化されるようになったため、新しい製品をリリースした場合、サポートするネットワークを決定することがさらに難しくなっています。1つのネットワークやデバイスを別のものに交換するには費用と時間がかかります。開発努力や機器の追加についても同様です。

Anybus CompactCom

Anybus CompactCom は十分なフレキシビリティを提供します。Anybus CompactCom ファミリの様々なモジュールは、ユーザがデバイスを産業用ネットワークに簡単に接続できるように設計されています。ネットワークの変更に伴う開発努力はほとんど必要なく、1つのネットワークのために開発されたアプリケーションは、簡単に別のネットワークに移動できます。

Anybus CompactCom ファミリの各モジュールは、特定の産業用ネットワーク専用です。モジュールには、ネットワーク上で交換されるデータをホスト・アプリケーションで使用できるようにするネットワーク特有のファームウェアが含まれています。これによって、ホスト・アプリケーションは、モジュールを通してネットワーク上のデバイスとして動作します。

ネットワークに関係なく、ホスト・アプリケーションとディベロッパを満たすソフトウェア・インターフェースは同じです。インターフェースはオブジェクト指向の方法で構成されています。各オブジェクトは関連する情報およびサービスのグループに集中しています。これによって、情報を分類し、アドレス指定する効率的な方法が提供されます。

Anybus CompactCom モジュールのファームウェアは、一般的に、モジュールの挙動およびネットワーク上での動作を制御します。ホスト・アプリケーションは Anybus CompactCom モジュールからアクセスされるため、ホスト・アプリケーションはこれらの要求を適切に処理するように設計されている必要があります。ただし、この処理は、モジュールがどの産業用ネットワーク専用であってもおおよそ影響はありません。理想的には、モジュールは Anybus CompactCom ファミリの他のモジュールと交換でき、ホスト・アプリケーション・ファームウェアは、変更しなくてもアプリケーションを動作させる必要があります。

Anybus CompactCom を使用することによって、開発コストを最大 70% 節約できます。全ての共通のフィールドバスおよびイーサネット・ネットワークにアクセスすることによって、最大限のフレキシビリティを実現します。これによって、競争力のあるネットワーク接続ソリューションを持つ市場にすばやくアクセスできます。

3. チュートリアル

3.1 はじめに

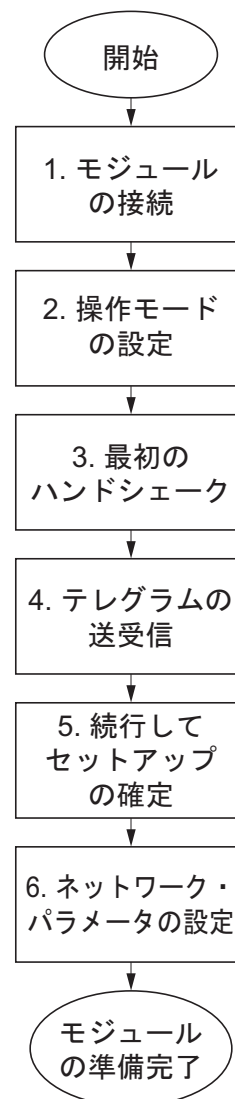
アプリケーションを実行するには、次の手順に従います。

1. Anybus CompactCom モジュールを接続します (6 ページの “Anybus CompactCom モジュールの接続” を参照)。
2. 操作モード (パラレルまたはシリアル、ボーレート) を設定します (8 ページの “操作モードの設定” を参照)。
3. 最初のハンドシェーキング機能を確認します (10 ページの “最初のハンドシェーク” (パラレル・モード) または 18 ページ (シリアル・モード) を参照)。
4. テレグラムの送受信を確認します (13 ページの “テレグラムの送信：パラレル通信” または 21 ページの “テレグラムの送信：シリアル通信” を参照)。
5. セットアップを確定します (28 ページの “セットアップの続き” を参照)。
6. ネットワーク特有の必要なパラメータを設定します (各 Network Interface Appendix を参照)。

ステップ 1 ~ 5 については、以下に説明します。Anybus CompactCom のコンセプトおよびモジュールの機能と特徴について、例を示して説明しています。

ステップ6については、各Network Interface Appendix で説明します。

手順を説明するために、通信の 2 つの追跡例を使用しました。アペンディックス 38 ページの “トレース、DeviceNet” および 44 ページの “トレース、Profibus DP-V1” を参照してください。



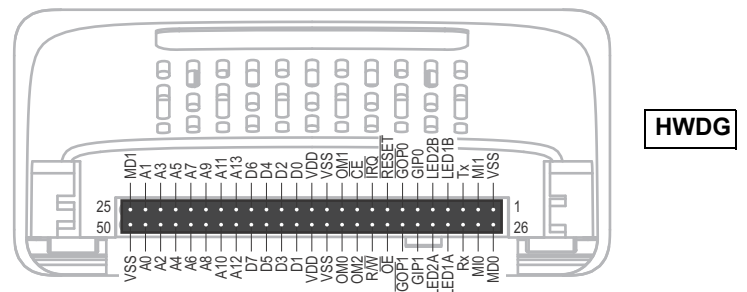
3.2 Anybus CompactCom モジュールの接続

Anybus CompactCom スタータ・キットを使用する場合、すぐに 18 ページの“シリアル・インターフェース・モード”に進むことができます。スタータ・キットではパラレル・モードを使用できません。



3.2.1 ホスト・インターフェース信号

Anybus CompactCom ホスト・インターフェースは、50 ピンの CompactFlash™ 型コネクタを使用します。



注意 1: ホスト・インターフェースは、CompactFlash™ 標準とは互換性のないピンです。

注意 2: インターフェースはホット・スワップに対応していません。モジュールの取り付けまたは取り外しの前に、電源を切る必要があります。電源を切らない場合、ホスト製品や Anybus CompactCom モジュールを損傷する可能性があります。

次の表に示す各信号の詳細については、"Anybus CompactCom Hardware Design Guide" で説明しています。

位置	信号	タイプ	機能	チュートリアルの例での接続方法
36, 11, 35	OM[0...2]	I	操作モード	8 ページの "操作モードの設定" を参照
27, 2	MI[0...1]	O	モジュールの識別	接続しない
8	RESET	I	リセット入力 (アクティブ・ロー)	ホスト・アプリケーションの制御可能な出力に接続
26, 25	MD[0...1]	O	モジュールの検出	接続しない
14, 39, 15, 40, 16, 41, 17, 42	D[0...7]	BI	パラレル・インターフェース (操作モードのピンで設定された場合はアクティブ)	未使用の場合、VSS に接続
49, 24, 48, 23, 47, 22, 46, 21, 45, 20, 44	A[0...10]	I		未使用の場合、VDD に接続
19, 43, 18	A[11...13]	I		未使用の場合、VSS に接続
10	CE	I		未使用の場合、VDD に接続
33	OE	I		未使用の場合、VDD に接続
34	R/W	I		未使用の場合、VDD に接続
9	IRQ	O		未使用の場合、接続しない
28	Rx	I		シリアル・インターフェース (操作モードのピンで設定された場合はアクティブ)
3	Tx	O	未使用の場合、接続しない	
30	LED2A	O	ネットワーク・ステータス LED 出力	接続しない
29	LED1A	O		
5	LED2B	O		
4	LED1B	O		
6, 31	GIP[0...1]	I	汎用 I/O	VSS に接続
7, 32	GOP[0...1]	O		接続しない
13, 38	VDD	PWR	電源供給	3.3V
1, 12, 37, 50	VSS	PWR	接地	接地

- I = 入力、CMOS (3.3V)
- O = 出力、CMOS (3.3V)
- BI = 双方向、トライステート
- P = 電源供給入力

注意 1: 機械的特性、測定値などについては、"Hardware Design Guide" のアペンディックス B "メカニカル仕様" を参照してください。

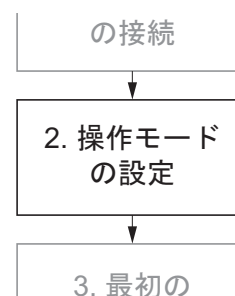
注意 2: ホスト・アプリケーションには、パラレル・インターフェースまたはシリアル・インターフェースのいずれかを接続します。使用されていないインターフェースは、表に示したとおりに接続する必要があります。

注意 3: ホスト・インターフェース信号は、5V に対する耐性がありません。

3.3 操作モードの設定

5 ページのフローチャートに従って、以下の表を参照してパラレル・モードまたはシリアル・モードのいずれかを選択します。

入力 OM2、OM1、および OM0 は、データの交換のために使用するインターフェース（パラレルまたはシリアル）および動作ボーレート（シリアル・インターフェースを使用する場合）を選択します。これらの信号の状態は起動中に 1 回サンプリングされます。何らかの変更を行う際は、一度リセットする必要があります。



SWDG

操作モード		設定		
パラレル・インターフェース 状態	シリアル・インターフェース 状態	OM2	OM1	OM0
有効	(無効)	ロー	ロー	ロー
(無効)	有効、ボーレート : 19.2kbps	ロー	ロー	ハイ
	有効、ボーレート : 57.6kbps	ロー	ハイ	ロー
	有効、ボーレート : 115.2kbps	ロー	ハイ	ハイ
	有効、ボーレート : 625kbps	ハイ	ロー	ロー

ロー = V_{IL}

ハイ = V_{IH}

注意 1 : RESET 信号を解除する前に、VDD、VSS、および操作モード入力の状態は安定している必要があります。これに従わない場合、障害または予期せぬ挙動が発生する可能性があります。

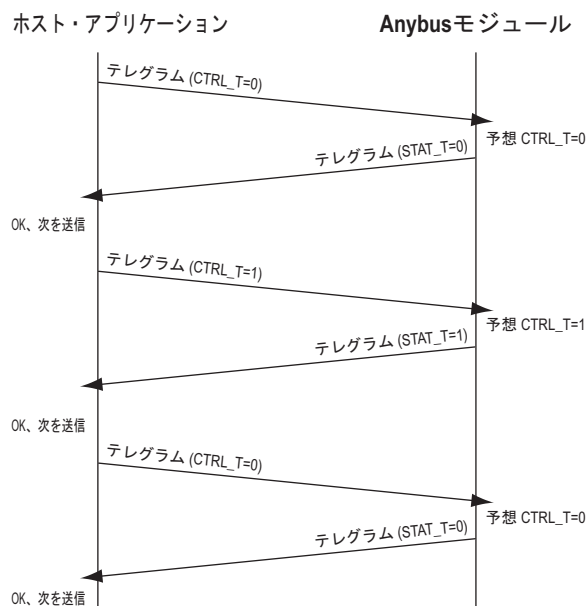
注意 2 : Anybus CompactCom スタータ・キット（Anybus CompactCom シリアル・アダプタを含む）を使用する場合、パラレル・モードは使用できません。

3.4 テレグラム

モジュールのハードウェアのセットアップはこれで完了です。以降のチュートリアルでは、ホスト・アプリケーションと Anybus CompactCom モジュールの通信のセットアップおよびモジュールのコンフィグレーションについて説明します。

Anybus CompactCom モジュールとの通信は、テレグラムによって処理されます。これらのテレグラムには明確なフォーマットがあり、各テレグラムには常に少なくとも制御またはステータス情報が含まれています。これらはピンポン・プロトコルで送信されます。つまりアプリケーションは少なくとも制御情報を含むテレグラムをモジュールに送信し、次のテレグラムを送信する前にテレグラムの返信を待ちます。返信されるテレグラムには、常に少なくともステータス情報が含まれます。

制御およびステータス情報はレジスタ内に構成されており、専用ビット（制御レジスタの CTRL_T とステータス・レジスタの STAT_T）は、再送信の必要性を示し、ポーリング・アプリケーションを実行可能にするために使用されます。¹



パラレル・モードであるかシリアル・モードであるかにかかわらず、全てのテレグラムには制御 / ステータス情報およびメッセージ・データ用の専用セクションがあります。モジュールが起動して動作している場合、プロセス・データのセクションもあります。テレグラム、レジスタ、およびメッセージの詳細については、この章の後半で説明します。

ユーザは予想されるレスポンスに対する最大時間を定義し、レスポンス時間が長すぎる場合にタイムアウトの例外を発生させることができます。詳細については、"Anybus CompactCom Software Design Guide" の第 3 章の "Anybus ウォッチドッグ" および "アプリケーション・ウォッチドッグ" のセクションを参照してください。

SWDG

1. これらのビットおよびレジスタの詳細については、12 ページ（パラレル・モード）と 19 ページ（シリアル・モード）を参照してください。

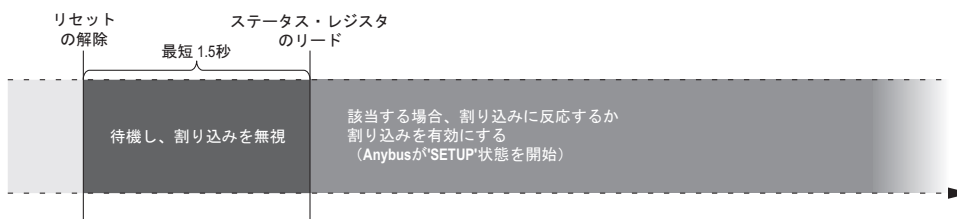
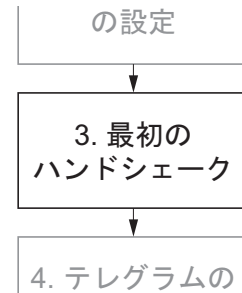
3.5 パラレル・インターフェース・モード

シリアル・インターフェース・モードについては、18 ページのセクション 3.6 に進んでください。

3.5.1 最初のハンドシェーク

最初のハンドシェークの目的は、Anybus Compact-Com モジュールの通信の準備を整えることです。準備が完了すると、モジュールは SETUP 状態になります。

内部ステータス・レジスタ（12 ページを参照）は起動時にクリアされます。これはセットアップを正しく行うために必要です。ステータス・レジスタを正しくリードすることができるように、ホスト・アプリケーションは、リセットまたは電源投入後少なくとも 1.5 秒間待つ必要があります。この間に検出される割り込みは無視する必要があります。この最初のハンドシェーク後、モジュールは SETUP 状態になり、設定できるようになります。ホスト・アプリケーションは、割り込みに反応するか、情報が入手可能であることを示す徴候に対して内部ステータス・レジスタをポーリングすることができます。



3.5.2 DPRAM

最初のハンドシェイクを行った後、ホスト・アプリケーションはテレグラムを使用して Anybus CompactCom モジュールとの通信を開始できます。このセクションでは、使用する通信手段について簡単に説明します。

アプリケーションと Anybus CompactCom モジュールの平行モードでの通信は、内部 DPRAM (デュアル・ポート・メモリ) で行われます。ホスト・アプリケーションは、組み込みシステムでのメモリと同様に、このメモリにアクセスし、処理できます。ホスト・アプリケーションからテレグラムが送信されると、ホスト・アプリケーションによって、内容が DPRAM 内の特定の領域に書き込まれます。ホスト・アプリケーションがテレグラムを受信すると、メモリの特定の領域は、ホスト・アプリケーションがリードするテレグラムの内容を保持します。

DPRAM メモリには、制御およびステータス・レジスタの領域 (ホスト・アプリケーションとモジュールの通信を制御するために使用される)、メッセージのリード/ライト領域、およびプロセス・データのリード/ライト領域が含まれます。異なる領域のアドレスは固定されています。ホスト・アプリケーションによって Anybus CompactCom に送信されるテレグラムには、常に DPRAM の制御レジスタにライトするデータが含まれています。逆方向に (Anybus CompactCom からホスト・アプリケーションに) 送信されるテレグラムは、常に DPRAM のステータス・レジスタを複製します。これらのレジスタの内容によっては、テレグラムにメッセージやプロセス・データが含まれる場合があります。プロセス・データ・リード/ライト領域は、セットアップおよびコンフィグレーション中には使用されません。

メモリ・マップ

下記に示すアドレス・オフセットは、モジュールのベース・アドレスを基準としています (平行・インターフェースが実装されたホスト・アプリケーションのメモリ・スペース内の領域の始まりから)。

アドレス・オフセット :	領域 :	アクセス :	注意 :
0000h... 37FFh	(予約)	-	(将来の使用のために予約)
3800h... 38FFh	プロセス・データ・ライト領域	ライト・オンリー	セットアップおよびコンフィグレーション中に使用されません。SWDG の "プロセス・データ・サブフィールド" を参照。
3900h... 39FFh	プロセス・データ・リード領域	リード・オンリー	セットアップおよびコンフィグレーション中に使用されません。SWDG の "プロセス・データ・サブフィールド" を参照。
3A00h... 3AFFh	(予約)	-	
3B00h... 3C06h	メッセージ・ライト領域	ライト・オンリー	ホスト・アプリケーションからモジュールへのメッセージ用に使用されます。SWDG の "オブジェクト・メッセージング" を参照。
3C07h... 3CFFh	(予約)	-	
3D00h... 3E06h	メッセージ・リード領域	リード・オンリー	モジュールからホスト・アプリケーションへのメッセージ用に使用されます。SWDG の "オブジェクト・メッセージング" を参照。
3E07h... 3FFDh	(予約)	-	
3FFEh	制御レジスタ	ライト・オンリー	12 ページの "制御レジスタ" を参照。
3FFFh	ステータス・レジスタ	リード・オンリー	12 ページの "ステータス・レジスタ" を参照。

重要 : C プログラマは必ず共有メモリ領域全体を *volatile* として宣言してください。

制御レジスタ（リード/ライト）

制御レジスタは、Anybus CompactCom モジュールとの通信を制御します。このチュートリアルでは、ビット 5 (CTRL_R)、6 (CTRL_M)、および 7 (CTRL_T) のみを使用されます。他のビットは一時的に無視できます。詳細については、"Anybus CompactCom Software Design Guide" を参照してください。

SWDG

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、新しいテレグラムを送信する場合にこのビットをトグルする必要があります。CTRL_T は、アプリケーションによってモジュールに送信される最初のテレグラムでは "1" に設定する必要があります。
CTRL_M	テレグラムにメッセージ・データが含まれる場合に設定します。
CTRL_R	設定されている場合、ホスト・アプリケーションは新しいコマンドを受信する準備ができています。
CTRL_AUX	予備ビット
-	(予約、ゼロに設定)

ステータス・レジスタ（リード・オンリー）

このレジスタは、Anybus CompactCom モジュールの現在のステータスを保持します。チュートリアルのこの段階では、ビット 5 (STAT_R)、ビット 6 (STAT_M)、およびビット 7 (STAT_T) のみを使用します。他のビットは一時的に無視できます。詳細については、"Anybus CompactCom Software Design Guide" を参照してください。

SWDG

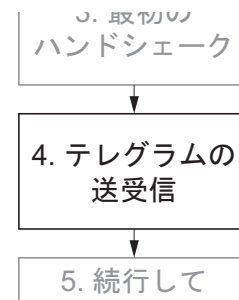
b7 (MSB)	b6	b5	b4	b3	b2	b1	b0 (LSB)
STAT_T	STAT_M	STAT_R	STAT_AUX	SUP	S2	S1	S0

ビット	説明
STAT_T	Anybus CompactCom モジュールが新しいテレグラムを発行すると、このビットは、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で、CTRL_T と同じ値に設定されます。
STAT_M	テレグラムにメッセージ・データが含まれる場合に設定します。
STAT_R	設定されている場合、Anybus CompactCom モジュールは新しいコマンドを受信する準備ができています。
STAT_AUX	予備ビット
SUP	値：意味： ^a 0：Anybus CompactCom モジュールは監視されません。 1：Anybus CompactCom モジュールは別のネットワーク・デバイスによって監視されます。
S _[0...2]	これらのビットは Anybus CompactCom モジュールの現在の状態を示します。
	S2 S1 S0 Anybus CompactCom 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

- a. このビットが全てのネットワークで使用されるわけではありません。詳細については、適切なアペンディックスを参照してください。

3.5.3 テレグラムの送信：パラレル通信

このセクションの例では、ホスト・アプリケーションが DPRAM 内の異なる領域に何をライトし、これらの領域から何をリードするかを示し、アプリケーションがテレグラムを送受信する際の手順を説明します。例えば、"W Control Reg 0x80" は、ホスト・アプリケーションが DPRAM の制御レジスタ領域に 16 進値 80 をライトすることを意味します。レジスタのビット 7 は 1 に設定され、レジスタの他のビットは全てゼロになります。



テレグラムの送信（パラレル）

最初のハンドシェークが行われると（10 ページを参照）、ホスト・アプリケーションは、モジュールへのテレグラムの送信を開始できるようになります。つまり、ホスト・アプリケーションは、DPRAM 内の制御レジスタ領域へのライトを開始します。

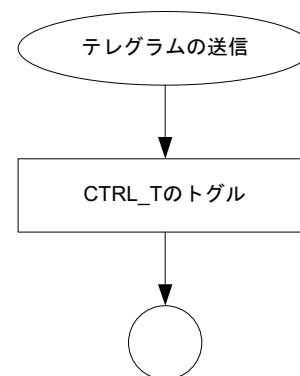
この例では、基盤となるテレグラムピンポン・プロトコルを示します。ホスト・アプリケーションからモジュールへの各テレグラムは、モジュールからホスト・アプリケーションへのレスポンス・テレグラムを生成します。

例：

```

W Control Reg 0x80
R Status Reg 0x80
W Control Reg 0x00
R Status Reg 0x00
W Control Reg 0x80
R Status Reg 0x80
W Control Reg 0x00
R Status Reg 0x00
etc.
  
```

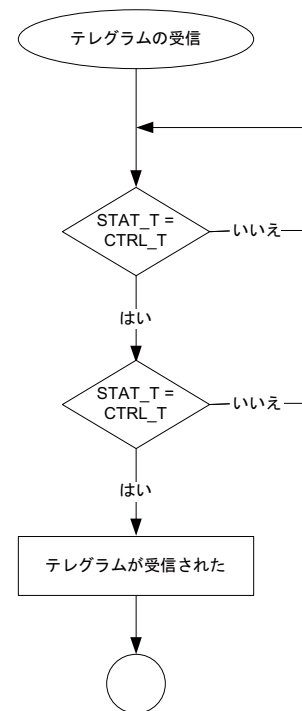
パラレル通信中、ホスト・アプリケーションは、DPRAM 内の制御レジスタの CTRL_T をトグルすることによって、テレグラムを送信します。このレジスタがライトされると、Anybus CompactCom モジュールに割り込みが生成され、モジュールは CTRL_T ビットをチェックして、新しいテレグラムが使用できるかどうか確認します。



Anybus CompactCom モジュールはそのステータスをチェックし、その時点で可能な全てのアクションを行ったことに対して満足した場合、ステータス・レジスタの STAT_T ビットを CTRL_T と等しくなるようにトグルします。有効なレスポンス・テレグラムがホスト・アプリケーションで使用できます。

ホスト・アプリケーションは STAT_T ビットをチェックし続けます。このビットが2回の連続したリードで CTRL_T と等しくなると、レスポンス・テレグラムを受信します。2回の連続したリードは、STAT_T ビットの値を安定させるために行われます。

Anybus CompactCom モジュールが正常に通信しているかぎり、テレグラムはこのようにピンポン・プロトコルで送受信されます。



テレグラムの連続的な交換が確立されます。ホスト・アプリケーションは、信号がモジュールにメッセージを送信するのを待ちます。

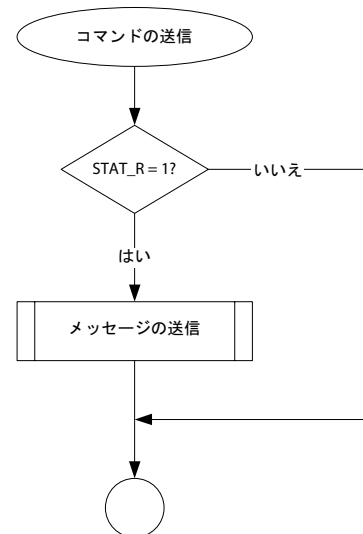
例：

```

W Control Reg 0x80
R Status Reg 0x80
W Control Reg 0x00
R Status Reg 0x00
W Control Reg 0x80
R Status Reg 0xa0
  
```

上記の例の最後の行では、ステータス・レジスタで、その STAT_T ビットが CTRL_T と等しくなるようにトグルされただけでなく、STAT_R ビットが 1 に設定されています。これは、モジュールがアプリケーションからコマンドを受け入れる準備ができたことを示しています。アプリケーションが次に送信するテレグラムで、テレグラムにメッセージ・データを追加できます。この段階でメッセージを送信することは必要でも必須でもありませんが、送信する場合、コマンドを送信する必要があります。

モジュールがメッセージを受け入れる準備ができる前に交換されるテレグラムの数は変動する可能性があります。



Anybus CompactCom モジュールへのメッセージの送信 (パラレル)

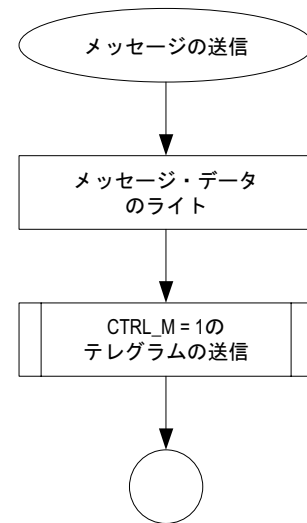
通信が機能していることを確認するために、コマンドを含むメッセージをモジュールに送信できます。例えば、この場合のように、メッセージはそのモジュール・タイプを返すためにモジュールに送信されます。

例：

```
W Control Reg 0x80
R Status Reg 0xa0
W Message Write Area 0x01, 0x01, 0x01,
0x00, 0x41,0x00, 0x01, 0x00
W Control Reg 0x40
R Status Reg 0x60
R Message Read Area 0x01, 0x01, 0x01,
0x00, 0x01,0x02, 0x01, 0x00, 0x01, 0x04
```

注意：DPRAM 内のメッセージ・ライト / リード領域のフォーマットと内容の詳細については、16 ページの表を参照してください。

アプリケーションは DPRAM 内のメッセージ・ライト領域にメッセージをライトします。11 ページのメモリ・マップを参照してください。これが完了すると、制御レジスタにライトし、CTRL_T ビットのトグルと CTRL_M ビットの設定を同時に行います。このレジスタに何かはライトされるたびに、Anybus モジュールに割り込みが生成されるため、レジスタ内の全てのビットを同時にライトする必要があります。同様に、アプリケーションが制御レジスタにライトする前に、メッセージ・ライト領域へのライトを終了している必要があります。



Anybus CompactCom モジュールからのメッセージの受信 (パラレル)

モジュールはホスト・アプリケーションが送信したテレグラムをリードし、モジュール・タイプを返信します。

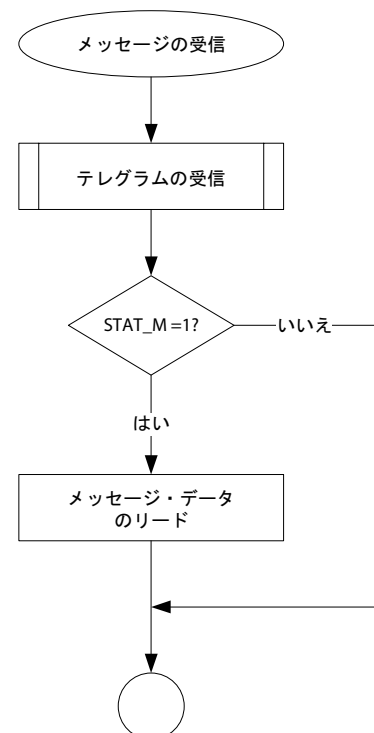
例 (繰り返し)：

```
W Message Write Area 0x01, 0x01, 0x01,
0x00, 0x41, 0x00, 0x01, 0x00
W Control Reg 0x40
R Status Reg 0x60
R Message Read Area 0x01, 0x01, 0x01,
0x00, 0x01, 0x02, 0x01, 0x00, 0x01, 0x04
```

アプリケーションが制御レジスタへのライトにより、そのテレグラムを送信し終わると、割り込みが生成され、モジュールはメッセージを含むテレグラムの受信を検出します。モジュールは DPRAM 内のメッセージ・ライト領域をリードし、コマンドを処理します (メッセージ・ライト / リード領域の内容の詳細については、16 ページの表を参照)。

モジュールはデータを処理してから、そのレスポンスをメッセージ・リード領域にライトします。これが終了すると、ステータス・レジスタにライトし、STAT_T ビットを CTRL_T ビットと等しくなるようにトグルし、STAT_M を 1 に設定して、メッセージがメッセージ・リード領域で使用できることを示します。

アプリケーションは、STAT_T が変化し、STAT_M が 1 に設定されていることを検出します。次のテレグラムが送信される前に、アプリケーションは DPRAM メッセージ・リード領域にあるメッセージを取り出す必要があります。



制御レジスタへのライト後、Anybus CompactCom モジュールがステータス・レジスタのアップデートを終了するまで、アプリケーションは DPRAM の内容をリードすることができません（ステータス・レジスタを除く）。これは、STAT_T¹ が CTRL_T と等しくなるまでステータス・レジスタをポーリングするか、Anybus CompactCom モジュールから割り込み信号（IRQ）を使用することにより、アプリケーションによって検出できます。

テレグラム内のメッセージの内容については、次の表に示します。メッセージ・レイアウトの詳細については、"Software Design Guide" の第 5 章を参照してください。

コマンド：

SWDG

領域 オフセット	ライト・メッセージ領域 内の要求の内容 ^a	説明（要求） ^b
0	00h	送信元 ID
1	01h	Anybus CompactCom モジュール内の Anybus オブジェクト
2	01h (lsb)	インスタンス 1
3	00h (msb)	
4	41h	メッセージ・タイプ：コマンド（モジュール・タイプを返す要求）
5	00h	メッセージ・データのサイズ（0 バイト）
6	01h	アトリビュート 1（モジュール・タイプ）
7	00h	（空き）

a. 全てのデータはリトル・エンディアンです。

b. オブジェクト・メッセージングは、Anybus CompactCom モジュールとホスト・アプリケーションのオブジェクトをアドレス指定する際に使用されます。詳細については、29 ページの“Anybus CompactCom へのアクセス”を参照してください。

レスポンス：

領域 オフセット	リード・メッセージ領域 内のレスポンスの内容 ^a	説明（レスポンス）
0	00h	送信元 ID ⁰
1	01h	Anybus CompactCom モジュール内の Anybus オブジェクト
2	01h (lsb)	インスタンス 1
3	00h (msb)	
4	01h	メッセージ・タイプ：レスポンス（コマンドがコピーされ、C ビットは 0 に設定される）
5	02h	メッセージ・データのサイズ（2 バイト）
6	01h	アトリビュート 1（モジュール・タイプ）
7	00h	（空き）
8-9	01h, 04h	メッセージ・データ：モジュール・タイプ（0401h = Anybus CompactCom）

a. 全てのデータはリトル・エンディアンです。

b. どのレスポンスがどの要求に属するかについてのつながりを保持するために、各メッセージには送信元 ID のタグが付けられます。コマンドを発行する場合、ホスト・アプリケーションは送信元 ID を任意に選択することができ、モジュールからのレスポンスは同じ送信元 ID を持つこととなります。レスポンスをモジュールからコマンドに送信する場合、ホスト・アプリケーションは、常に元のコマンドを使用してメッセージの送信元 ID、オブジェクト番号、およびインスタンス番号をコピーする必要があります。コマンドもコピーされますが、C ビットは 0 に設定されます。

注意：上記のとおり、テレグラムの送信は制御レジスタを経由してトリガされ、新しいテレグラムの受信はステータス・レジスタで示されます。これは、制御レジスタにアクセスする前に、プロセス・データおよびメッセージのサブフィールドをライトする必要があることを意味します。モジュールはこれを複数のアクセスとして解釈する可能性があるため、このレジスタではビット操作または他のリード / 修正 / ライト命令を直接使用しないでください。代わりに、全てのビット操作などを一時レジスタで行い、操作後にライトしなおす必要があります。また、テレグラムの受信を待っている間、ステータス・レジスタのポーリング以外でのパラレル・インターフェースへのアクセスは回避する必要があります。

1. ステータス・レジスタをポーリングする際に有効な結果を保証するには、2 回の連続したリードが一致するまでリードする手順（2 回の連続したリードで同じ値がリードされる）を使用する必要があります。

パラレル・テレグラム処理、要約

パラレル・インターフェースでは、送信と受信はハンドシェイク・レジスタ（ステータス・レジスタと制御レジスタ）によって管理されます。

テレグラムを送信するには、次の手順を行います。

1. 該当する場合、ライト・プロセス・データをプロセス・データ・ライト領域（3800h...38FFh）にライトします。
2. 該当する場合、メッセージをメッセージ・ライト領域（3B00h...3C06h）にライトします。
3. 制御レジスタをアップデートし、送信をトリガします。

テレグラムの受信は、ステータス・レジスタの STAT_T ビットによって合図されます。ホスト・アプリケーションは、ステータス・レジスタをサイクリックにポーリングして新しいテレグラムを検出するか、割り込み操作に依存することができます。

重要：

- ステータス・レジスタをポーリングする際に有効な結果を保証するには、2回の連続したリードが一致するまでリードする（2回の連続したリードで同じ値がリードされる）手順を使用する必要があります。
- ABCC モジュールがコマンドも送信できるように、ホスト・アプリケーションは継続的なピンポン通信を保証する必要があります。
- 1つのコマンドは常に1つのレスポンスのみを発生させます。
- メッセージ・レスポンスは、CTRL_R ビットまたは STAT_R ビットの状態にかかわらず、常に送信できます。
- レスポンスが送信される前に、いくつかのテレグラムが発生する場合があります。
- いくつかのコマンドが両方向でレスポンスを待っている可能性があります。CTRL_R ビットまたは STAT_R ビットが設定されているかぎり、それぞれの方向で新しいコマンドが可能です。
- いくつかのコマンドがレスポンスを待たずに送信された場合、レスポンスの順序はコマンドの順序と異なる可能性があります。

28 ページの“セットアップの続き”に進んでください。

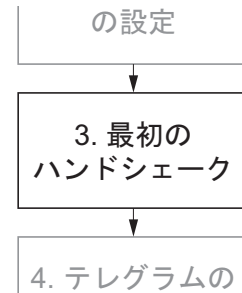
3.6 シリアル・インターフェース・モード

このセクションではセクション 3.5 と同じ手順を説明しますが、これはシリアル・インターフェース・モードの手順についてです。

3.6.1 最初のハンドシェーク

最初のハンドシェークの目的は、Anybus CompactCom モジュールの通信の準備を整えることです。準備が完了すると、モジュールは SETUP 状態になります。

内部ステータス・レジスタ (20 ページを参照) は起動時にクリアされます。これはセットアップを正しく行うために必要です。従って、モジュールがシリアル・データを受信する準備ができるように、ホスト・アプリケーションは、最初のテレグラムを送信する前に、リセットまたは電源投入後少なくとも 1.5 秒待つ必要があります。



3.6.2 シリアル・テレグラム・フレーム

シリアル・モードで送信されるテレグラムは全て、次に示すフレームを使用します。SETUP 中、プロセス・データ・サブフィールドはテレグラムには存在せず、送信する 19 バイトを各テレグラムに残します。

1 バイト	16 バイト	最大 256 バイト	2 バイト
ハンド シェーク・ レジスタ・ フィールド	メッセージ・ サブフィールド	プロセス・データ・サブフィールド (SETUP 中は存在しない)	CRC16
(最初の バイト)	このフィールドの詳細 については、"Software Design Guide" を参照して ください。		(最後の バイト)

- ハンドシェーク・レジスタ・フィールド

このフィールドには、Anybus CompactCom モジュールに送信されるテレグラムの制御レジスタ (19 ページを参照) と、Anybus CompactCom モジュールから受信されるテレグラムのステータス・レジスタ (20 ページを参照) が含まれます。

- メッセージ・サブフィールド

このフィールドは、送信するメッセージ (またはメッセージ・フラグメント) を保持します。プロセス・データのスループットを維持するために、メッセージ・サブフィールドは、シリアル・インターフェースを使用する場合、16 バイトに制限されます。より長いメッセージは、いくつかのより小さいフラグメントとして交換されます ("Software Design Guide" の "フラグメンテーション" を参照)。

SWDG

- プロセス・データ・サブフィールド

このフィールドには、Anybus CompactCom モジュールに送信されるテレグラムのライト・プロセス・データと、Anybus CompactCom モジュールから受信するリード・プロセス・データが含まれます。Anybus CompactCom モジュールが 'SETUP' 状態で動作している場合、このフィールドは存在しません。

- CRC16

このフィールドは、16 ビット・サイクリック冗長検査を保持します ("Software Design Guide" の "CRC 計算" を参照)。CRC は、CRC 自体を除いたテレグラム全体を対象としています。

SWDG

制御レジスタ (送信)

制御レジスタは、Anybus CompactCom モジュールとの通信を制御します。このチュートリアルでは、ビット 5 (CTRL_R)、6 (CTRL_M)、および 7 (CTRL_T) のみを使用します。他のビットは一時的に無視できます。詳細については、"Anybus CompactCom Software Design Guide" を参照してください。

SWDG

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、新しいテレグラムを送信する場合にこのビットをトグルする必要があります。CTRL_T は、アプリケーションによってモジュールに送信される最初のテレグラムでは "1" に設定する必要があります。
CTRL_M	テレグラムにメッセージ・データが含まれる場合に設定します。
CTRL_R	設定されている場合、ホスト・アプリケーションは新しいコマンドを受信する準備ができています。
CTRL_AUX	予備ビット
-	(予約、ゼロに設定)

ステータス・レジスタ（受信）

このレジスタは、Anybus CompactCom モジュールの現在のステータスを保持します。チュートリアルはこの段階では、ビット 5 (STAT_R)、ビット 6 (STAT_M)、およびビット 7 (STAT_T) のみを使用します。他のビットは一時的に無視できます。詳細については、"Anybus CompactCom Software Design Guide" を参照してください。

SWDG

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

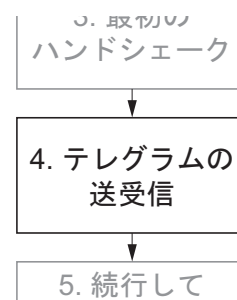
ビット	説明																																				
STAT_T	Anybus CompactCom モジュールが新しいテレグラムを発行すると、このビットは、ホスト・アプリケーションから最後に受信したテレグラムで、CTRL_T と同じ値に設定されます。																																				
STAT_M	テレグラムにメッセージ・データが含まれる場合に設定します。																																				
STAT_R	設定されている場合、Anybus CompactCom モジュールは新しいコマンドを受信する準備ができています。																																				
STAT_AUX	予備ビット																																				
SUP	値：意味 ^a 0：Anybus CompactCom モジュールは監視されません。 1：Anybus CompactCom モジュールは別のネットワーク・デバイスによって監視されます。																																				
S _[0...2]	これらのビットは Anybus CompactCom モジュールの現在の状態を示します。																																				
	<table border="1"> <thead> <tr> <th>S2</th> <th>S1</th> <th>S0</th> <th>Anybus CompactCom の状態</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> <td>SETUP</td> </tr> <tr> <td>0</td> <td>0</td> <td>1</td> <td>NW_INIT</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> <td>WAIT_PROCESS</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> <td>IDLE</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> <td>PROCESS_ACTIVE</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> <td>ERROR</td> </tr> <tr> <td>1</td> <td>1</td> <td>0</td> <td>(予約)</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> <td>EXCEPTION</td> </tr> </tbody> </table>	S2	S1	S0	Anybus CompactCom 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
S2	S1	S0	Anybus CompactCom 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																																		

- a. このビットが全てのネットワークで使用されるわけではありません。詳細については、適切なアペンディックスを参照してください。

3.6.3 テレグラムの送信：シリアル通信

このセクションでの例は、ホスト・アプリケーションから Anybus CompactCom モジュールに送信されるもの、およびアプリケーションによってモジュールから受信されるものを示し、シリアル通信を使用する際の手順について説明します。

シリアル・モードで送信される全てのテレグラムには、全てのフィールドが必要ではない場合でも、19 バイト含まれます。使用されないフィールドはゼロで埋められます。従って、これらの例は各テレグラムの内容全体を示します。



テレグラムの送信（シリアル）

最初のハンドシェイクが行われると（18 ページを参照）、ホスト・アプリケーションは、モジュールへのテレグラムの送信を開始できるようになります。

例：

```
Send:{0x80, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
bCrccHi, bCrccLo}
```

```
Receive:{0x80, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, bCrccHi, bCrccLo}
```

```
Send:{0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
bCrccHi, bCrccLo}
```

```
Receive:{0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, bCrccHi, bCrccLo}
```

```
Send:{0x80, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
bCrccHi, bCrccLo}
```

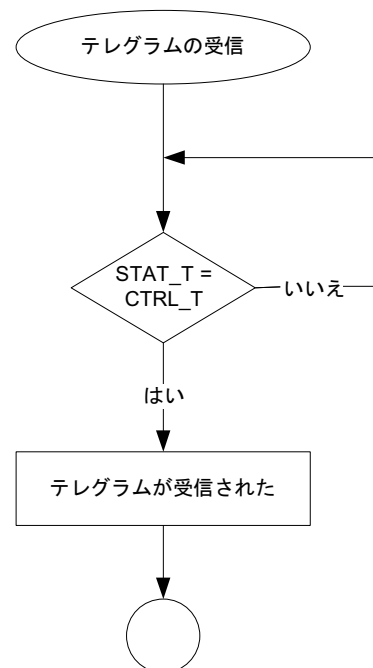
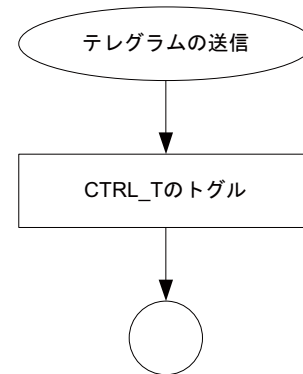
```
Receive:{0x80, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, bCrccHi, bCrccLo}
```

etc.

アプリケーションは、制御レジスタの CTRL_T ビットが 1 に設定された状態で、テレグラムを送信します。モジュールはテレグラムを返信し、STAT_T ビットを CTRL_T と等しくなるようにトグルします。データはまだ送信されておらず、テレグラム・フレームの残りは、各フレームを終了する CRC (bCrccHi, bCrccLo) 以外は空です。

ほとんどのテレグラム・フレームに情報がない場合でも、毎回完全なフレームを送信する必要があります。使用されない部分はゼロで埋められます。

Anybus CompactCom モジュールが正常に通信しているかぎり、テレグラムはこのようにピンポン・プロトコルで送受信されます。



テレグラムの連続的な交換が確立されます。ホスト・アプリケーションは、信号がモジュールにメッセージを送信するのを待ちます。

例：

```
Send:{0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, bCrcHi, bCrcLo}
```

```
Receive:{0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, bCrcHi,
bCrcLo}
```

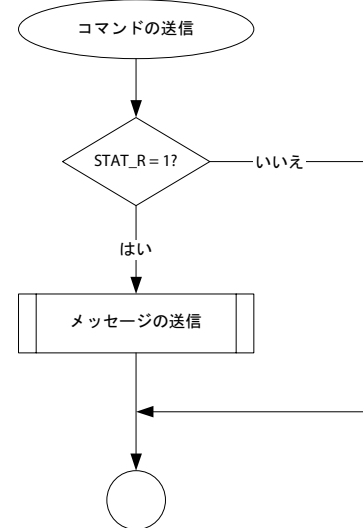
```
Send:{0x80, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, bCrcHi, bCrcLo}
```

```
Receive:{0xa0, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, bCrcHi,
bCrcLo}
```

上記の例の最後のテレグラムでは、ステータス・レジスタで、その STAT_T ビットが CTRL_T と等しくなるようにトグルされただけでなく、STAT_R ビットが 1 に設定されています。これは、モジュールがアプリケーションからコマンドを受け入れる準備ができたことを示しています。

アプリケーションが次に送信するテレグラムで、テレグラムにメッセージ・データを追加できます。この段階でメッセージを送信することは必要でも必須でもありませんが、送信する場合、コマンドを送信する必要があります。

モジュールがメッセージを受け入れる準備ができる前に交換されるテレグラムの数は変動する可能性があります。



Anybus CompactCom モジュールへのメッセージの送信 (シリアル)

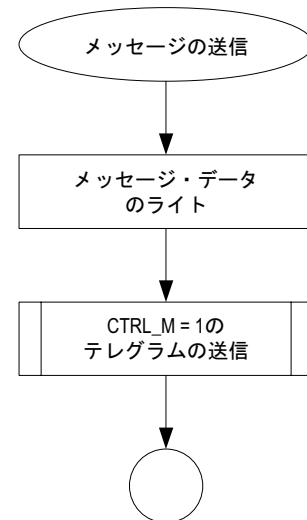
通信が機能していることを確認するために、コマンドを含むメッセージのあるテレグラムをホスト・アプリケーションからモジュールに送信できます。例えば、この場合のように、メッセージはそのモジュール・タイプを返すためにモジュールに送信できます。

例：

```
Send:{0x40, 0x00, 0x00, 0x01, 0x01, 0x00,
0x41, 0x00, 0x01, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, bCrcHi, bCrcLo}
```

```
Receive:{0x20, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, bCrcHi,
bCrcLo}
```

アプリケーションは、テレグラム・フレームのメッセージ・データ・サブフィールドのコマンドを送信します (次の表を参照)。モジュールはテレグラムを承認しますが、メッセージ内の全てのデータを受信しているかどうかはわからないため、まだ処理を行いません。



最初に送信されたテレグラムの内容の説明：

バイト番号	アプリケーションからモジュールへ ^a	説明 (モジュールへ) ^b
1	40h	制御レジスタ、CTRL_M = 1
2	00h	送信元 ID
3	01h	Anybus CompactCom モジュール内の Anybus オブジェクト インスタンス 1
4	01h (lsb)	
5	00h (msb)	
6	41h	メッセージ・タイプ：コマンド (モジュール・タイプを返す要求)
7	00h	メッセージ・データのサイズ (0 バイト)
8	01h	アトリビュート 1 (モジュール・タイプ)
9	00h	(空き)
10-17	00h、00h、00h、00h、00h、00h、00h、00h	(メッセージ・データなし)
18-19	bCrcHi、bCrcLo	CRC

- 全てのデータはリトル・エンディアンです。
- オブジェクト・メッセージングは、Anybus CompactCom モジュールとホスト・アプリケーションのオブジェクトをアドレス指定する際に使用されます。詳細については、29 ページの“Anybus CompactCom へのアクセス”を参照してください。

メッセージ・サブフィールド (バイト 2 ~ 17) のレイアウトについては、SWDG の第 5 章で説明しています。

この例でアプリケーションがモジュールから受信するテレグラムは、最初のバイト (20h) のステータス・レジスタと最後の 2 つのバイトの CRC 以外は空です。STAT_T ビットは、CTRL_T と同じ値にトグルされます。STAT_R が設定され、これは新しいコマンドを受け入れることができることを示します。テレグラム・フレームには常に 19 バイトあるため、空のメッセージ・サブフィールドはゼロで埋められます。

SWDG

メッセージの受信 (シリアル)

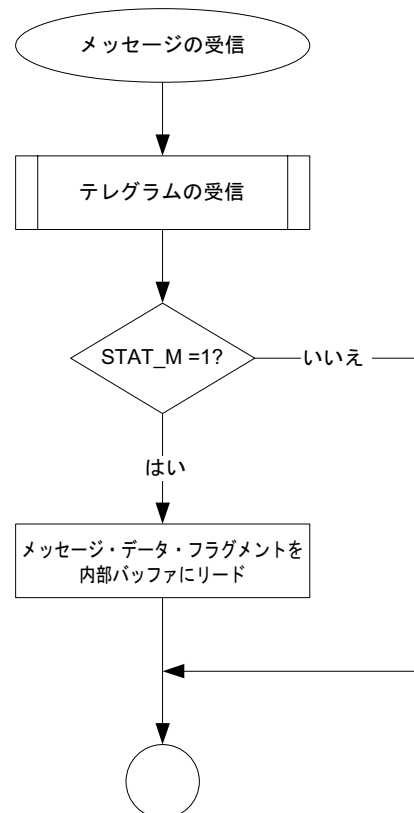
ホスト・アプリケーションは必要な全てのメッセージ・データを Anybus CompactCom モジュールに送信しましたが、このことをモジュールに通知する必要があります。通知後、モジュールは、ホスト・アプリケーションによって送信されたコマンドに対するレスポンスを含むテレグラムを送信します。

例：

```
Send:{0x80, 0x00, 0x00, 0x00, 0x00,
      0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
      0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
      bCrcHi, bCrcLo}
```

```
Receive:{0xE0, 0x00, 0x01, 0x01, 0x00,
          0x01, 0x02, 0x01, 0x00, 0x01, 0x04,
          0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
          bCrcHi, bCrcLo}
```

アプリケーションによって送信されたテレグラムには、メッセージ・データが含まれていません。このテレグラムは全てのメッセージ・データを受信したことをモジュールに通知し、モジュールは以前に受信したコマンドの処理を開始できるようになります。この例では、モジュールからのレスポンスは、アプリケーションがモジュールから受信する次のテレグラムに含まれます (次の表を参照)。これは必ずしも実情ではない可能性があります。モジュールがコマンドの処理を終了し、レスポンスを送信する前に、いくつかのテレグラムを交換できます。



アプリケーションによって 2 番目に受信されるテレグラムの内容の説明：

バイト番号	モジュールからアプリケーションへ ^a	説明 (モジュールへ / モジュールから)
1	E0h	ステータス・レジスタ
2	00h	送信元 ID ^b
3	01h	Anybus CompactCom モジュール内の Anybus オブジェクト
4	01h (lsb)	インスタンス 1
5	00h (msb)	
6	01h	メッセージ・タイプ：レスポンス (コマンドがコピーされ、C ビットは 0 に設定される)
7	02h	メッセージ・データのサイズ (0/2 バイト)
8	01h	アトリビュート 1 (モジュール・タイプ)
9	00h	(空き)
10-11	01h, 04h	メッセージ・データ：モジュール・タイプ (0401h = Anybus CompactCom)
12-17	00h, 00h, 00h, 00h, 00h, 00h	(空き)
18-19	bCrcHi, bCrcLo	CRC

- 全てのデータはリトル・エンディアンです。
- どのレスポンスがどの要求に属するかについてのつながりを保持するために、各メッセージには送信元 ID のタグが付けられます。コマンドを発行する場合、ホスト・アプリケーションは送信元 ID を任意に選択することができ、モジュールからのレスポンスは同じ送信元 ID を持つこととなります。レスポンスをモジュールからコマンドに送信する場合、ホスト・アプリケーションは、常に元のコマンドを使用してメッセージの送信元 ID、オブジェクト番号、およびインスタンス番号をコピーする必要があります。コマンドもコピーされますが、C ビットは 0 に設定されます。

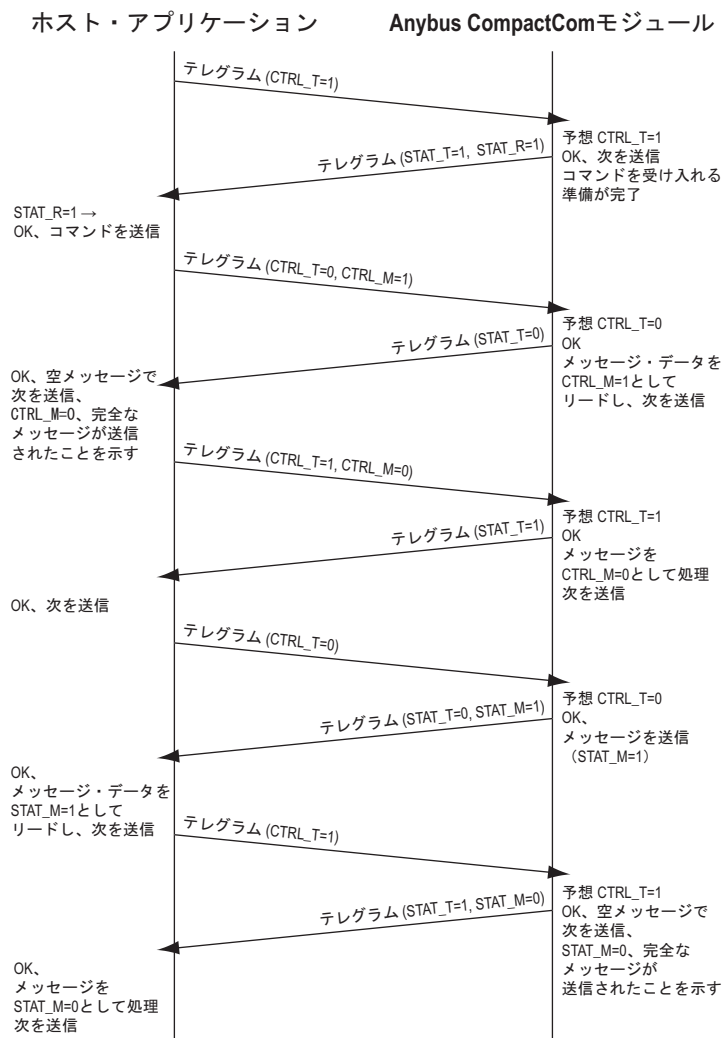
メッセージのレイアウトの詳細については、SWDG の第 5 章を参照してください。

SWDG

シリアル通信でのテレグラムの数

シリアル通信では、CTRL_M = 0（アプリケーションから）または STAT_M = 0（Anybus CompactCom モジュールから）があるテレグラムは、メッセージに属している全てのデータが送信されたことを通知します。これは、どんなメッセージでも少なくとも2つのテレグラムをモジュールに伝送する必要があり、その逆も同様であることを意味します。

下の図は、ホスト・アプリケーションと Anybus CompactCom モジュールの間での一連のテレグラムの例を示します。アプリケーションはテレグラムを送信し、モジュールがコマンドを受け入れる準備ができたことを確認します。モジュールは STAT_R が 1 に設定されたテレグラム（コマンドを受け入れる準備ができていることを示す）を返信します。アプリケーションは、コマンドを含むメッセージがあるテレグラムを送信できるようになります。Anybus CompactCom モジュールはこのテレグラムを受信しますが、空メッセージ・フィールドのあるテレグラムを受信して、全てのメッセージが送信されたことを確認するまで、メッセージを処理しません。Anybus CompactCom はメッセージを処理してレスポンスを返信し、空メッセージを含むテレグラムによって終了します。



このピンポン通信の例の各矢印は、アプリケーションから Anybus CompactCom モジュールへ、またはその逆の方向に送信されるテレグラムを示します。

シリアルメッセージ処理、要約

シリアル・インターフェースでは、メッセージの送受信は、ハンドシェイク・レジスタ（その内容は常にテレグラムの最初のバイトで使用可能）によって管理されます。

メッセージを送信するには、次の手順を行います。

1. テレグラム・フレームのメッセージ・サブフィールドにメッセージがある Anybus CompactCom モジュールにテレグラムを送信し、CTRL_M = 1 を設定します。
2. STAT_T が CTRL_T と等しいモジュールからのテレグラムを待ちます。
3. メッセージがテレグラム・フレーム内の定義済みフィールドより大きい場合、メッセージを断片化する必要があります。全てのメッセージ・データが送信されるまで、メッセージ・データと CTRL_M が設定されたテレグラムを送信し続けます。
4. Anybus CompactCom モジュールに CTRL_M = 0 のテレグラムを送信し、メッセージの送信が終了したことを示します。

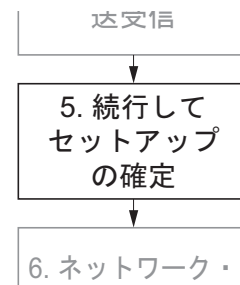
SWDG

重要：

- ABCC モジュールがコマンドも送信できるように、ホスト・アプリケーションは継続的なピンポン通信を保証する必要があります。
- 1 つのコマンドは常に 1 つのレスポンスのみを発生させます。
- メッセージ・レスポンスは、CTRL_R ビットまたは STAT_R ビットの状態にかかわらず、常に送信できます。
- レスポンスが送信される前に、いくつかのテレグラムが発生する場合があります。
- いくつかのコマンドが両方向でレスポンスを待っている可能性があります。CTRL_R ビットまたは STAT_R ビットが設定されているかぎり、それぞれの方向で新しいコマンドが可能です。
- いくつかのコマンドがレスポンスを待たずに送信された場合、レスポンスの順序はコマンドの順序と異なる可能性があります。

3.7 セットアップの続き

このチュートリアルの残りの例では、制御レジスタとステータス・レジスタおよびメッセージ・リード/ライト領域の内容を示します。テレグラムは前のセクションで示したプロトコルに従ってメッセージとデータの伝送手段としてのみ機能するため、シリアルまたはパラレル通信をこのレベルの通信で使用することは適切ではありません。

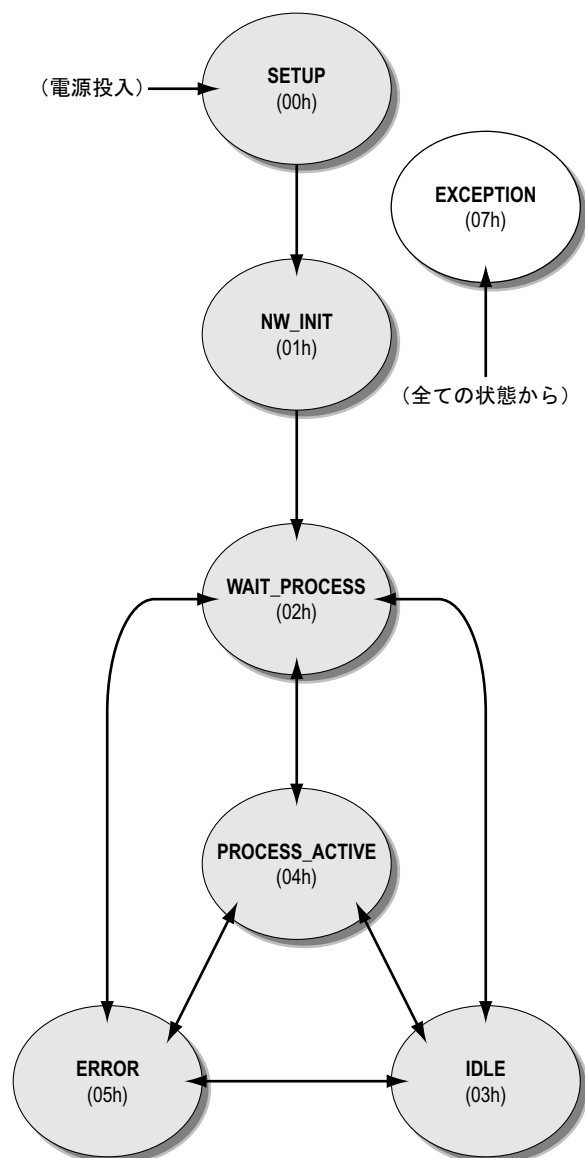


3.7.1 Anybus CompactCom ステート・マシン

最初のハンドシェイク後、Anybus CompactCom は SETUP 状態になります。

ホスト・アプリケーションはモジュールにテレグラムを送信でき、レスポンスのリード / 受信を行います。モジュールとの全ての通信は、メッセージとデータの伝送手段として機能するテレグラム内で定義されます。¹ このセクションでは、Anybus CompactCom のセットアップの続きについて説明し、モジュールにプロセス・データの送受信の準備をさせる方法について説明します。

セットアップが終了すると、モジュールはネットワーク関連の初期化タスクを行う NW_INIT 状態になります。これらのタスクは、使用するネットワーク・インターフェースによって異なります。これが終了すると、モジュールは WAIT_PROCESS 状態になり、選択されたネットワーク・インターフェースの通信に参加する準備が完了します。CompactCom モジュールの状態は常にステータス・レジスタのビット 0 ~ 2 をリードすることによって決定できますが (12 ページの “ステータス・レジスタ (リード・オンリー)” を参照)、状態間の移行が速すぎるために全ての移行を検出できない場合があります。



1. SETUP状態ではプロセス・データを送信できません。メッセージ・データのみを送信できます。

3.7.2 Anybus CompactCom へのアクセス

モジュールに送信されるメッセージの最初の部分には、コマンド / 要求をモジュール・ファームウェア内のどこに送信するかを識別する情報が含まれています。

モジュール内の情報とサービスはオブジェクトに分類され、各オブジェクトは、相互に属しているか関連している情報およびサービスを保持します。これによって、コンフィグレーションのためにモジュールにアクセスする簡単で効率的な方法（オブジェクト・メッセージング）が提供されます。各メッセージには、メッセージに関連付けられたデータまたは設定の位置を指定するオブジェクト番号とインスタンス番号がタグ付けされます。

モジュールからの要求は、アプリケーションを同じように設計するために同じフォーマットを使用します。構造が類似したオブジェクトを、Anybus CompactCom ファームウェアのオブジェクトに対して実装します。

ホスト・アプリケーションは、要求されたオブジェクトが存在しない場合でも、モジュールから受信する全ての要求を処理できる必要があります。モジュールがメッセージを送信する場合、アプリケーションで実装されていないオブジェクトをアドレス指定するので、アプリケーションはエラー・メッセージでレスポンスする必要があります。詳細については、"Anybus CompactCom Software Design Guide" の "オブジェクト・モデル" を参照してください。

アプリケーションが実装する必要があるオブジェクトが 1 つだけあります。それはアプリケーション・データ・オブジェクトで、いわゆる ADI（アプリケーション・データ・インスタンス、次のセクションを参照）を通してデータを交換するために使用されます。ほとんどのアプリケーションでは、認証のために必要になる可能性があるため、より多くのオブジェクトを実装することをお奨めします。"Network Interface Appendix" を参照してください。

SWDG

3.7.3 ADI のマッピング

データはホスト・アプリケーションにあるアプリケーション・データ・オブジェクトの ADI (アプリケーション・データ・インスタンス) を通してネットワーク上で交換されます。各 ADI はネットワーク・データのブロックを表します。ADI は、通常では、ネットワーク側のアサイクリック・パラメータに関連しています。プロセス・データとしてマッピングされる場合もあります。プロセス・データは Anybus CompactCom ホスト・プロトコル内の専用データ・チャネルを通して交換されますが、通常は高速のサイクリック・ネットワーク I/O に関連付けられています。

各 ADI には、名前、データ・タイプ、範囲、およびデフォルト値をタグ付けすることができます。これにより、ネットワークから簡単にアクセスできるようになります。

ADI の正確な表示はネットワークに特有であるため、この例では、単純かつ比較的一般的な ADI マッピングのみを示します。最初の 2 バイト (データ・タイプ UINT16) と次の 1 バイト (データ・タイプ UINT8) は、プロセス・リード領域にマッピングされます。これはその領域のセットアップにすぎないことにご注意ください。領域自体はセットアップが終了するまでアクセスできません。メッセージの内容のみを例に示します。

例：

```
First message to module:
{0x02, 0x03, 0x01, 0x00, 0x51, 0x04, 0x01, 0x00, 0x05, 0x01, 0x01, 0x00}

The module responds with:
{0x02, 0x03, 0x01, 0x00, 0x11, 0x01, 0x01, 0x00, 0x00}

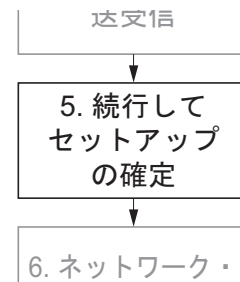
Second message to module:
{0x03, 0x03, 0x01, 0x00, 0x51, 0x04, 0x02, 0x00, 0x04, 0x01, 0x02, 0x00}

The module responds with:
{0x03, 0x03, 0x01, 0x00, 0x11, 0x01, 0x02, 0x00, 0x02}
```

例の詳細については、下記および次のページの表を参照してください。

最初のコマンド：

領域 オフセット	要求の内容	説明 (要求)
0	02h	送信元 ID
1	03h	Anybus CompactCom モジュール内のネットワーク・オブジェクト
2-3	01h 00h	インスタンス 1
4	51h	メッセージ・タイプ : コマンド (Map_ADI_Read_Area)
5	04h	メッセージ・データのサイズ (4)
6-7	01h 00h	ADI インスタンス番号 (1)
8	05h	データ・タイプ UINT16
9	01h	ADI 内のエレメントの数 (1)
10-11	01h 00h	ADI のオーダー番号 (1)



最初のレスポンス :

領域 オフセット	レスポンスの内容	説明 (レスポンス)
0	02h	送信元 ID
1	03h	Anybus CompactCom モジュール内のネットワーク・オブジェクト
2-3	01h 00h	インスタンス 1
4	11h	メッセージ・タイプ : レスポンス
5	01h	メッセージ・データのサイズ (1 バイト)
6-7	01h 00h	ADI インスタンス番号 (1)
8	00h	成功したレスポンス、マッピングされた ADI のオフセットを与える

2 番目のコマンド :

領域 オフセット	要求の内容	説明 (要求)
0	03h	送信元 ID
1	03h	Anybus CompactCom モジュール内のネットワーク・オブジェクト
2-3	01h 00h	インスタンス 1
4	51h	メッセージ・タイプ : コマンド (Map_ADI_Read_Area)
5	04h	メッセージ・データのサイズ (4)
6-7	02h 00h	ADI インスタンス番号 (2)
8	04h	データ・タイプ UINT8
9	01h	ADI 内のエレメントの数 (1)
10-11	02h 00h	ADI のオーダー番号 (2)

2 番目のレスポンス :

領域 オフセット	レスポンスの内容	説明 (レスポンス)
0	03h	送信元 ID
1	03h	Anybus CompactCom モジュール内のネットワーク・オブジェクト
2-3	01h 00h	インスタンス 1
4	11h	メッセージ・タイプ : レスポンス
5	01h	メッセージ・データのサイズ (1 バイト)
6-7	02h 00h	ADI インスタンス番号 (2)
8	02h	成功したレスポンス、マッピングされた ADI のオフセットを与える

注意 :

- マッピングされた最初の ADI のデータ・タイプは UINT8 であるため、マッピングされた ADI のオフセットは、2 番目のレスポンスで 2 です。

詳細については、"Anybus CompactCom Software Design Guide" の "ネットワーク・オブジェクト" と "メッセージ・レイアウト" を参照してください。

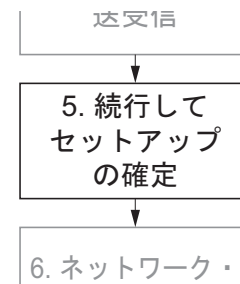
SWDG

3.7.4 セットアップの完了

セットアップを終了するために、アプリケーションはモジュールにこのメッセージを送信します。

例：

```
The following message is sent to the module:
{0x04, 0x01, 0x01, 0x00, 0x42, 0x01, 0x05,
0x00, 0x01}
The module responds with:
{0x04, 0x01, 0x01, 0x00, 0x02, 0x00, 0x05,
0x00}
```



コマンド：

領域 オフセット	要求の内容	説明（要求）
0	04h	送信元 ID
1	01h	Anybus CompactCom モジュール内の Anybus オブジェクト
2	01h (lsb)	インスタンス 1
3	00h (msb)	
4	42h	メッセージ・タイプ：コマンド (Set_attribute)
5	01h	メッセージ・データのサイズ (1 バイト)
6	05h	アトリビュート 5 (セットアップ完了)
7	00h	(空き)
8	01h	セットアップ完了アトリビュートを 1 に設定

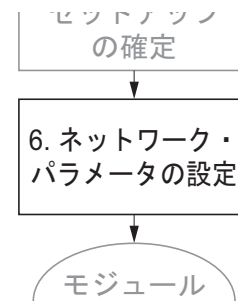
レスポンス：

領域 オフセット	レスポンスの内容	説明（レスポンス）
0	04h	送信元 ID
1	01h	Anybus CompactCom モジュール内の Anybus オブジェクト
2	01h (lsb)	インスタンス 1
3	00h (msb)	
4	02h	メッセージ・タイプ：レスポンス (コマンドがコピーされ、C ビットは 0 に設定される)
5	00h	メッセージ・データのサイズ (0 バイト)
6	05h	アトリビュート 5 (セットアップ完了)
7	00h	(空き)

セットアップが終了し、1 に設定されたセットアップ完了アトリビュートの合図によって、Anybus CompactCom は状態を SETUP から NW_INIT に変更します。モジュールの状態は、常にステータス・レジスタのビット b2、b1、および b0 で示されます (12 ページを参照)。

3.8 ネットワークの初期化

Anybus CompactCom は NW_INIT 状態になりました。モジュールはアプリケーションにコマンドを送信し始め、ネットワーク通信を初期化します。正確にどんなコマンドを送信するかは、産業用ネットワークの選択によって異なります。例については、アペンディックス（38 ページの DeviceNet と 44 ページの PROFIBUS DP-V1）を参照してください。受信した各コマンドに対して、アプリケーションは、要求されたオブジェクトがアプリケーションで実装されない場合でも、モジュールに有効なレスポンスを与える必要があります。この場合、ベンダ ID（DeviceNet オブジェクト内のパラメータ）を返送する要求は、モジュールから送信されます。このオブジェクトはアプリケーションで実装されないため、アプリケーションは "サポートされていないオブジェクト" というエラー・メッセージを返信します。



例：

The following message is sent from the module to the application:
 {0x06, 0xfc, 0x01, 0x00, 0x41, 0x00, 0x01, 0x00}

The application responds with:
 {0x06, 0xfc, 0x01, 0x00, 0x81, 0x01, 0x01, 0x00, 0x03}

コマンド（モジュールからホスト・アプリケーションへ）：

領域 オフセット	要求の内容	説明（要求）
0	06h	送信元 ID
1	FCh	ホスト・アプリケーションの DeviceNet オブジェクト
2	01h (lsb)	インスタンス 1
3	00h (msb)	
4	41h	メッセージ・タイプ：コマンド (Get_attribute)
5	00h	メッセージ・データのサイズ (0 バイト)
6	01h	アトリビュート 1 (ベンダ ID)
7	00h	(空き)

レスポンス（ホスト・アプリケーションからモジュールへ）：

領域 オフセット	レスポンスの内容	説明（レスポンス）
0	06h	送信元 ID
1	FCh	ホスト・アプリケーションの DeviceNet オブジェクト
2	01h (lsb)	インスタンス 1
3	00h (msb)	
4	81h	メッセージ・タイプ：エラー・メッセージ
5	01h	メッセージ・データのサイズ (1 バイト)
6	01h	アトリビュート 1 (ベンダ ID)
7	00h	(空き)
8	03h	エラー・メッセージ：サポートされていないオブジェクト

上記のとおり、コマンドは Anybus CompactCom によってサポートされた産業用ネットワークによって異なります。また、コマンドの数も異なります。モジュールによって要求される全てのアトリビュートのデフォルト値があるので、例に示すような有効なレスポンスが与えられるかぎり、モジュールはネットワークの初期化を継続します。終了すると、

WAIT_PROCESS 状態になります (28 ページの “Anybus CompactCom ステート・マシン” を参照)。

3.9 さらになるコンフィグレーションと認証

上記の例は非常に簡単な実装を示しています。ホスト・アプリケーションには様々な種類のさらに多くのオブジェクトが含まれており、これらの一部の実装が推奨される場合があります (最終生成物が問題になっているフィールドバス機関の認証を受ける場合など)。

必要な設定や初期化の詳細について説明している各アペンディックスを参照してください。

4. リソース

前の章では、Anybus CompactCom モジュールでアプリケーションを実装する方法の比較的簡単な例について説明しました。モジュールは非常に多機能で、このチュートリアルで使用される機能よりもかなり多くの機能が含まれています。この章では、独自のアプリケーションを作成する場合に使用できる他の資料を示します。

4.1 機能のカテゴリ化

Anybus CompactCom とアプリケーションのオブジェクト（アトリビュートとサービスを含む）は、3つのカテゴリ（基本、拡張、および高度）に分類されます。

4.1.1 基本

このカテゴリには、実装または使用するために必要なオブジェクト、アトリビュート、およびサービスが含まれます。これらは Anybus CompactCom を起動し、選択されたネットワーク・プロトコルでデータを送受信するには十分です。産業用ネットワークの基本的な機能が使用されます。

これらのオブジェクトについては、このチュートリアル（および適切な産業用ネットワークのアペンディックス）で説明しています。製品を認証できるようにする追加オブジェクトもこのカテゴリに属しています。

4.1.2 拡張

このカテゴリにあるオブジェクトを使用すると、アプリケーションの機能が拡張されます。ネットワークとの間でのデータの基本的な移動だけでなく、産業用ネットワークのより具体的な特徴にアクセスします。アプリケーションには余分の値が与えられます。

4.1.3 高度

このグループに属するオブジェクト、アトリビュート、およびサービスは、特殊な、または、ほとんど使用されない機能を提供します。使用できるネットワーク機能のほとんどは有効で、アクセス可能です。産業用ネットワークの仕様へのアクセスは通常では必要です。

4.2 設計ガイド

4.2.1 Anybus CompactCom Software Design Guide

このマニュアルは、ソフトウェア・インターフェースの完全なマニュアルです。全ての Anybus CompactCom 通信モジュールに共通のソフトウェア・インターフェースの、全ての部分に関する情報を提供します。

4.2.2 Anybus CompactCom Hardware Design Guide

このマニュアルは、Anybus CompactCom プラットフォームの機械的および電気的特性を十分に理解できるようにすることを意図しています。

4.3 Network Interface Appendix

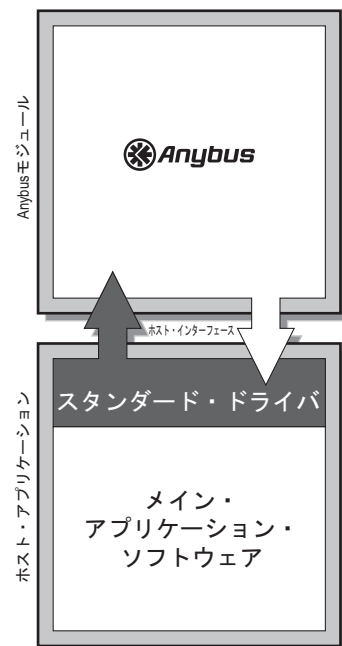
各モジュールには、ネットワーク・インターフェース特有の問題（認証されるようにアプリケーションで実装するには何がかなど）について説明するアペンディックスが付いています。

4.4 ドライバ

HMS は 2 つのフリー・ソース・レベル (C 言語) のソフトウェア・ドライバ (スタンダード・ドライバとライト・ドライバ) を供給しています。これらのドライバは、Anybus モジュールとホスト・アプリケーションの間で "接着剤" として作用することによって、開発プロセスをスピードアップするように設計されています。これらは低レベルの通信タスクをホスト・ソフトウェア環境から分離します。スタンダード・ドライバは、高レベルの抽出とフレキシビリティを維持しつつ、Anybus-CompactCom コンセプトの多機能性を活用するように設計されています。これは、より小さいアプリケーションでは非現実的になる可能性があります。ある程度のメモリと処理能力自体が必要であることを意味します。このギャップを埋めるために、HMS は、アプリケーションに適した最小限度のソリューションに厳しいメモリや性能の要求を提供する代替ドライバ ('ライト・ドライバ' として知られる) を供給しています。

スタンダード・ドライバは完全に自立型です。機能するためにオペレーティング・システムを必要とせず、割り込みによって駆動するサービスとして動作するか、またはホスト・ファームウェアによってサイクリックにポーリングされます。

ドライバ (マニュアルを含む) は、www.anybus.com からダウンロードできます。



4.5 スタータ・キット

スタータ・キットは Anybus CompactCom プラットフォームに使用できます。スタータ・キットには、シリアル・ホスト・インターフェース・チャンネル経由でネットワークング・アプリケーションを開発するために使用できる評価ボードが含まれています。

スタータ・キットには、Software Development Kit (SDK) も含まれています。このキットの目的は、Anybus CompactCom との通信に Anybus CompactCom ドライバを使用してソフトウェア・アプリケーションを実装する方法を示すことです。

SDK は win32 PC 上のコンソール・アプリケーションとして動作し、シリアル・ポートを使用して Anybus CompactCom と通信するように設計されています。

詳細については、www.anybus.com を参照してください。

4.6 ドライブ・プロファイル

Anybus-CompactCom Drive Profile 製品系列では、Anybus-CompactCom のコンセプトを追加のドライブ・プロファイル機能で拡張します。この拡張されたソフトウェア機能によって、製造業者はドライブの最新通信規格に適合する製品を製造しやすくなります。このチュートリアルではドライブ・プロファイルについては触れませんが、ここで扱っている問題は、当然それらの製品にも有効です。www.anybus.com では、一般的なドライブ・プロファイルとドライブ・プロファイルを組み込んだモジュールの両方の資料を入手できます。

アペンディックス A

A. トレース、DeviceNet

この付録では、ホスト・アプリケーションと Anybus CompactCom DeviceNet モジュールの間のパラレル・モードでの通信例のトランスクリプションを示します。"データ" 欄には、ホスト・アプリケーションによって指定された DPRAM 領域にライトされたデータ、もしくはその領域からリードされたデータが含まれます。DeviceNet については、"Anybus CompactCom Software Design Guide" および "Network Interface Appendix" も参照してください。

R/W	DPRAM 領域	データ	宛先オブジェクト	インスタンス	コマンド/レスポンス	情報
W	制御レジスタ	0x80				最初のハンドシェーク
R	ステータス・レジスタ	0x80				
W	制御レジスタ	0x00				
R	ステータス・レジスタ	0x00				
W	制御レジスタ	0x80				
R	ステータス・レジスタ	0xa0				
モジュール・タイプの要求						
W	メッセージ・ライト領域	0x01、0x01、0x01、0x00、0x41、0x00、 0x01、0x00	Anybus (0x01)	1	Get_Attribute	Attr = MODULE_TYPE (0x01)
W	制御レジスタ	0xc0				
R	ステータス・レジスタ	0xa0				
R	メッセージ・リード領域	0x01、0x01、0x01、0x00、0x01、0x02、 0x01、0x00、0x01、0x04			ACK	ModuleType = ABCC (0x0401)
1バイトをリード・プロセス・データにマッピング						
W	メッセージ・ライト領域	0x02、0x03、0x01、0x00、0x51、0x04、 0x01、0x00、0x04、0x01、0x01、0x00	ネットワーク (0x03)	1	Map_ADI_Read_Area	ADI = 0x0001、DataType = UINT8 (0x04)、 NumElements = 0x01、OrderNum = 0x0001
W	制御レジスタ	0xc0				
R	ステータス・レジスタ	0xa0				
W	制御レジスタ	0x00				
R	ステータス・レジスタ	0x60				
R	メッセージ・リード領域	0x02、0x03、0x01、0x00、0x11、0x01、0x01、 0x00、0x00			ACK	AreaOffset = 0x00
セットアップの完了						
W	メッセージ・ライト領域	0x03、0x01、0x01、0x00、0x42、0x01、 0x05、0x00、0x01	Anybus (0x01)	1	Set_Attribute	Attr = SETUP_COMPLETE (0x05)、 Value = TRUE (≠0x00)
W	制御レジスタ	0xc0				
R	ステータス・レジスタ	0xe0				
R	メッセージ・リード領域	0x03、0x01、0x01、0x00、0x02、0x00、 0x05、0x00			ACK	
W	制御レジスタ	0x00				
R	ステータス・レジスタ	0x21				(Anybus State = NW_INIT)
W	制御レジスタ	0x80				

R/W	DPRAM 領域	データ	宛先オブジェクト	インスタンス	コマンド / レスポンス	情報
R	ステータス・レジスタ	0xa1				
W	制御レジスタ	0x00				
R	ステータス・レジスタ	0x21				
モジュールからコマンドを受け入れ、エラー・レスポンスを返信 (CTRL_R=1)						
W	制御レジスタ	0xa0				
R	ステータス・レジスタ	0xe1				
R	メッセージ・リード領域	0x06、0xfc、0x01、0x00、0x41、0x00、0x01、0x00	DeviceNet (0xfc)	1	Get_Attribute	Attr = VENDOR_ID (0x01)
W	メッセージ・ライト領域	0x06、0xfc、0x01、0x00、0x81、0x01、0x01、0x00、0x03			NAK	Error = UNSUP_OBJ (0x03)
W	制御レジスタ	0x60				
R	ステータス・レジスタ	0x21				
W	制御レジスタ	0xa0				
R	ステータス・レジスタ	0xe1				
R	メッセージ・リード領域	0x06、0xfc、0x01、0x00、0x41、0x00、0x02、0x00	DeviceNet (0xfc)	1	Get_Attribute	Attr = DEVICE_TYPE (0x02)
W	メッセージ・ライト領域	0x06、0xfc、0x01、0x00、0x81、0x01、0x02、0x00、0x03			NAK	Error = UNSUP_OBJ (0x03)
W	制御レジスタ	0x60				
R	ステータス・レジスタ	0x21				
W	制御レジスタ	0xa0				
R	ステータス・レジスタ	0xe1				
R	メッセージ・リード領域	0x06、0xfc、0x01、0x00、0x41、0x00、0x03、0x00	DeviceNet (0xfc)	1	Get_Attribute	Attr = PRODUCT_CODE (0x03)
W	メッセージ・ライト領域	0x06、0xfc、0x01、0x00、0x81、0x01、0x03、0x00、0x03			NAK	Error = UNSUP_OBJ (0x03)
W	制御レジスタ	0x60				
R	ステータス・レジスタ	0x21				
W	制御レジスタ	0xa0				
R	ステータス・レジスタ	0xe1				
R	メッセージ・リード領域	0x06、0xfc、0x01、0x00、0x41、0x00、0x04、0x00	DeviceNet (0xfc)	1	Get_Attribute	Attr = REVISION (0x04)
W	メッセージ・ライト領域	0x06、0xfc、0x01、0x00、0x81、0x01、0x04、0x00、0x03			NAK	Error = UNSUP_OBJ (0x03)

R/W	DPRAM 領域	データ	宛先オブジェクト	インスタンス	コマンド / レスポンス	情報
W	制御レジスタ	0x60				
R	ステータス・レジスタ	0x21				
W	制御レジスタ	0xa0				
R	ステータス・レジスタ	0xe1				
R	メッセージ・リード領域	0x06、0xfc、0x01、0x00、0x41、0x00、0x05、0x00	DeviceNet (0xfc)	1	Get_Attribute	Attr = SERIAL_NUMBER (0x05)
W	メッセージ・ライト領域	0x06、0xfc、0x01、0x00、0x81、0x01、0x05、0x00、0x03			NAK	Error = UNSUP_OBJ (0x03)
W	制御レジスタ	0x60				
R	ステータス・レジスタ	0x21				
W	制御レジスタ	0xa0				
R	ステータス・レジスタ	0xe1				
R	メッセージ・リード領域	0x06、0xfc、0x01、0x00、0x41、0x00、0x06、0x00	DeviceNet (0xfc)	1	Get_Attribute	Attr = PRODUCT_NAME (0x06)
W	メッセージ・ライト領域	0x06、0xfc、0x01、0x00、0x81、0x01、0x06、0x00、0x03			NAK	Error = UNSUP_OBJ (0x03)
W	制御レジスタ	0x60				
R	ステータス・レジスタ	0x21				
W	制御レジスタ	0xa0				
R	ステータス・レジスタ	0xe1				
R	メッセージ・リード領域	0x06、0xfc、0x01、0x00、0x41、0x00、0x08、0x00	DeviceNet (0xfc)	1	Get_Attribute	Attr = CONS_INSTANCE (0x08)
W	メッセージ・ライト領域	0x06、0xfc、0x01、0x00、0x81、0x01、0x08、0x00、0x03			NAK	Error = UNSUP_OBJ (0x03)
W	制御レジスタ	0x60				
R	ステータス・レジスタ	0x21				
W	制御レジスタ	0xa0				
R	ステータス・レジスタ	0xe1				
R	メッセージ・リード領域	0x06、0xfc、0x01、0x00、0x41、0x00、0x07、0x00	DeviceNet (0xfc)	1	Get_Attribute	Attr = PROD_INSTANCE (0x07)
W	メッセージ・ライト領域	0x06、0xfc、0x01、0x00、0x81、0x01、0x07、0x00、0x03			NAK	Error = UNSUP_OBJ (0x03)
W	制御レジスタ	0x60				
R	ステータス・レジスタ	0x21				

R/W	DPRAM 領域	データ	宛先オブジェクト	インスタンス	コマンド / レスポンス	情報
W	制御レジスタ	0xa0				
R	ステータス・レジスタ	0xe1				
R	メッセージ・リード領域	0x06, 0xfc, 0x01, 0x00, 0x41, 0x00, 0x09, 0x00	DeviceNet (0xfc)	1	Get_Attribute	Attr = ADDRESS_FROM_NET (0x09)
W	メッセージ・ライト領域	0x06, 0xfc, 0x01, 0x00, 0x81, 0x01, 0x09, 0x00, 0x03			NAK	Error = UNSUP_OBJ (0x03)
W	制御レジスタ	0x60				
R	ステータス・レジスタ	0x21				
W	制御レジスタ	0xa0				
R	ステータス・レジスタ	0xe1				
R	メッセージ・リード領域	0x06, 0xfc, 0x01, 0x00, 0x41, 0x00, 0x0a, 0x00	DeviceNet (0xfc)	1	Get_Attribute	Attr = BAUD_RATE_FROM_NET (0x0a)
W	メッセージ・ライト領域	0x06, 0xfc, 0x01, 0x00, 0x81, 0x01, 0x0a, 0x00, 0x03			NAK	Error = UNSUP_OBJ (0x03)
W	制御レジスタ	0x60				
R	ステータス・レジスタ	0x21				
W	制御レジスタ	0xa0				
R	ステータス・レジスタ	0xe1				
R	メッセージ・リード領域	0x06, 0xfc, 0x01, 0x00, 0x41, 0x00, 0x0b, 0x00	DeviceNet (0xfc)	1	Get_Attribute	Attr = ENABLE_APP_CIP_OBJECTS (0x0b)
W	メッセージ・ライト領域	0x06, 0xfc, 0x01, 0x00, 0x81, 0x01, 0x0b, 0x00, 0x03			NAK	Error = UNSUP_OBJ (0x03)
W	制御レジスタ	0x60				
R	ステータス・レジスタ	0x21				
W	制御レジスタ	0xa0				
R	ステータス・レジスタ	0xe1				
R	メッセージ・リード領域	0x06, 0xfc, 0x01, 0x00, 0x41, 0x00, 0x0c, 0x00	DeviceNet (0xfc)	1	Get_Attribute	Attr = ENABLE_PARAM_OBJECT (0x0c)
W	メッセージ・ライト領域	0x06, 0xfc, 0x01, 0x00, 0x81, 0x01, 0x0c, 0x00, 0x03			NAK	Error = UNSUP_OBJ (0x03)
W	制御レジスタ	0x60				
R	ステータス・レジスタ	0x21				
W	制御レジスタ	0xa0				
R	ステータス・レジスタ	0xe1				

R/W	DPRAM 領域	データ	宛先オブジェクト	インスタンス	コマンド / レスポンス	情報
R	メッセージ・リード領域	0x06、0xfc、0x01、0x00、0x41、0x00、0x0d、0x00	DeviceNet (0xfc)	1	Get_Attribute	Attr = ENABLE_QUICK_CONNECT (0x0d)
W	メッセージ・ライト領域	0x06、0xfc、0x01、0x00、0x81、0x01、0x0d、0x00、0x03			NAK	Error = UNSUP_OBJ (0x03)
W	制御レジスタ	0x60				
R	ステータス・レジスタ	0x21				
W	制御レジスタ	0xa0				
R	ステータス・レジスタ	0xe1				
R	メッセージ・リード領域	0x06、0xff、0x01、0x00、0x41、0x00、0x02、0x00	アプリケーション (0xff)	1	Get_Attribute	Attr = SUP_LANG (0x02)
W	メッセージ・ライト領域	0x06、0xff、0x01、0x00、0x81、0x01、0x02、0x00、0x03			NAK	Error = UNSUP_OBJ (0x03)
W	制御レジスタ	0x60				
R	ステータス・レジスタ	0x21				
W	制御レジスタ	0xa0				
R	ステータス・レジスタ	0xa1				
W	制御レジスタ	0x20				
R	ステータス・レジスタ	0x22				(Anybus State = WAIT_PROCESS)

アペンディックス B

B. トレース、Profibus DP-V1

この付録では、ホスト・アプリケーションと Anybus CompactCom Profibus DP-V1 モジュールの間のパラレル・モードでの通信例のトランスクリプションを示します。"データ" 欄には、ホスト・アプリケーションによって指定された DPRAM 領域にライトされたデータ、もしくはその領域からリードされたデータが含まれます。Profibus DP-V1 については、"Anybus CompactCom Software Design Guide" および "Network Interface Appendix" も参照してください。

R/W	DPRAM 領域	データ	宛先オブジェクト	インスタンス	コマンド/ レスポンス	情報
W	制御レジスタ	0x80				最初のハンドシェーク
R	ステータス・レジスタ	0x80				
W	制御レジスタ	0x00				
R	ステータス・レジスタ	0x00				
W	制御レジスタ	0x80				
R	ステータス・レジスタ	0xa0				
モジュール・タイプの要求						
W	メッセージ・ライト領域	0x01、0x01、0x01、0x00、0x41、0x00、 0x01、0x00	Anybus (0x01)	1	Get_Attribute	Attr = MODULE_TYPE (0x01)
W	制御レジスタ	0x40				
R	ステータス・レジスタ	0x60				
R	メッセージ・リード領域	0x01、0x01、0x01、0x00、0x01、0x02、 0x01、0x00、0x01、0x04			ACK	ModuleType = ABCC (0x0401)
1バイトをリード・プロセス・データにマッピング						
W	メッセージ・ライト領域	0x02、0x03、0x01、0x00、0x51、0x04、 0x01、0x00、0x04、0x01、0x01、0x00	ネットワーク (0x03)	1	Map_ADI_Read_Area	ADI = 0x0001、DataType = UINT8 (0x04)、 NumElements = 0x01、OrderNum = 0x0001
W	制御レジスタ	0xc0				
R	ステータス・レジスタ	0xa0				
W	制御レジスタ	0x00				
R	ステータス・レジスタ	0x60				
R	メッセージ・リード領域	0x02、0x03、0x01、0x00、0x11、0x01、 0x01、0x00、0x00			ACK	AreaOffset = 0x00
セットアップの完了						
W	メッセージ・ライト領域	0x03、0x01、0x01、0x00、0x42、0x01、 0x05、0x00、0x01	Anybus (0x01)	1	Set_Attribute	Attr = SETUP_COMPLETE (0x05)、 Value = TRUE (≠0x00)
W	制御レジスタ	0xc0				
R	ステータス・レジスタ	0xe0				
R	メッセージ・リード領域	0x03、0x01、0x01、0x00、0x02、0x00、 0x05、0x00			ACK	
W	制御レジスタ	0x00				
R	ステータス・レジスタ	0x21				(Anybus State = NW_INIT)
W	制御レジスタ	0x80				

R/W	DPRAM 領域	データ	宛先オブジェクト	インスタンス	コマンド/レスポンス	情報
R	ステータス・レジスタ	0xa1				
W	制御レジスタ	0x00				
R	ステータス・レジスタ	0x21				
モジュールからコマンドを受け入れ、エラー・レスポンスを返信 (CTRL_R=1)						
W	制御レジスタ	0xa0				
R	ステータス・レジスタ	0xe1				
R	メッセージ・リード領域	0x00、0xfd、0x01、0x00、0x41、0x00、 0x01、0x00	PROFIBUS DP-V1 (0xfd)	1	Get_Attribute	Attr = IDENT_NUMBER (0x01)
W	メッセージ・ライト領域	0x00、0xfd、0x01、0x00、0x81、0x01、 0x01、0x00、0x03			NAK	Error = UNSUP_OBJ (0x03)
W	制御レジスタ	0x60				
R	ステータス・レジスタ	0x21				
W	制御レジスタ	0xa0				
R	ステータス・レジスタ	0xe1				
R	メッセージ・リード領域	0x01、0xfd、0x01、0x00、0x41、0x00、 0x06、0x00	PROFIBUS DP-V1 (0xfd)	1	Get_Attribute	Attr = BUFFER_MODE (0x06)
W	メッセージ・ライト領域	0x01、0xfd、0x01、0x00、0x81、0x01、 0x06、0x00、0x03			NAK	Error = UNSUP_OBJ (0x03)
W	制御レジスタ	0x60				
R	ステータス・レジスタ	0x21				
W	制御レジスタ	0xa0				
R	ステータス・レジスタ	0xe1				
R	メッセージ・リード領域	0x00、0xfd、0x01、0x00、0x41、0x00、 0x03、0x00	PROFIBUS DP-V1 (0xfd)	1	Get_Attribute	Attr = EXPECTED_CFG_DATA (0x03)
W	メッセージ・ライト領域	0x00、0xfd、0x01、0x00、0x81、0x01、 0x03、0x00、0x03			NAK	Error = UNSUP_OBJ (0x03)
W	制御レジスタ	0x60				
R	ステータス・レジスタ	0x21				
W	制御レジスタ	0xa0				
R	ステータス・レジスタ	0xe1				
R	メッセージ・リード領域	0x01、0xfd、0x01、0x00、0x41、0x00、 0x05、0x00	PROFIBUS DP-V1 (0xfd)	1	Get_Attribute	Attr = SIZEOF_ID_REL_DIAG (0x05)
W	メッセージ・ライト領域	0x01、0xfd、0x01、0x00、0x81、0x01、 0x05、0x00、0x03			NAK	Error = UNSUP_OBJ (0x03)

R/W	DPRAM 領域	データ	宛先オブジェクト	インスタンス	コマンド/レスポンス	情報
W	制御レジスタ	0x60				
R	ステータス・レジスタ	0x21				
W	制御レジスタ	0xa0				
R	ステータス・レジスタ	0xe1				
R	メッセージ・リード領域	0x00、0xfd、0x01、0x00、0x41、0x00、 0x07、0x00	PROFIBUS DP-V1 (0xfd)	1	Get_Attribute	Attr = ALARM_SETTINGS (0x07)
W	メッセージ・ライト領域	0x00、0xfd、0x01、0x00、0x81、0x01、 0x07、0x00、0x03			NAK	Error = UNSUP_OBJ (0x03)
W	制御レジスタ	0x60				
R	ステータス・レジスタ	0x21				
W	制御レジスタ	0xa0				
R	ステータス・レジスタ	0xe1				
R	メッセージ・リード領域	0x01、0xfd、0x01、0x00、0x41、0x00、 0x08、0x00	PROFIBUS DP-V1 (0xfd)	1	Get_Attribute	Attr = MANUFACTURER_ID (0x08)
W	メッセージ・ライト領域	0x01、0xfd、0x01、0x00、0x81、0x01、 0x08、0x00、0x03			NAK	Error = UNSUP_OBJ (0x03)
W	制御レジスタ	0x60				
R	ステータス・レジスタ	0x21				
W	制御レジスタ	0xa0				
R	ステータス・レジスタ	0xe1				
R	メッセージ・リード領域	0x00、0xfd、0x01、0x00、0x41、0x00、 0x09、0x00	PROFIBUS DP-V1 (0xfd)	1	Get_Attribute	Attr = ORDER_ID (0x09)
W	メッセージ・ライト領域	0x00、0xfd、0x01、0x00、0x81、0x01、 0x09、0x00、0x03			NAK	Error = UNSUP_OBJ (0x03)
W	制御レジスタ	0x60				
R	ステータス・レジスタ	0x21				
W	制御レジスタ	0xa0				
R	ステータス・レジスタ	0xe1				
R	メッセージ・リード領域	0x01、0xfd、0x01、0x00、0x41、0x00、 0x0a、0x00	PROFIBUS DP-V1 (0xfd)	1	Get_Attribute	Attr = SERIAL_NO (0x0a)
W	メッセージ・ライト領域	0x01、0xfd、0x01、0x00、0x81、0x01、 0x0a、0x00、0x03			NAK	Error = UNSUP_OBJ (0x03)
W	制御レジスタ	0x60				
R	ステータス・レジスタ	0x21				

R/W	DPRAM 領域	データ	宛先オブジェクト	インスタンス	コマンド/レスポンス	情報
W	制御レジスタ	0xa0				
R	ステータス・レジスタ	0xe1				
R	メッセージ・リード領域	0x00、0xfd、0x01、0x00、0x41、0x00、 0x0b、0x00	PROFIBUS DP-V1 (0xfd)	1	Get_Attribute	Attr = HW_REV (0x0b)
W	メッセージ・ライト領域	0x00、0xfd、0x01、0x00、0x81、0x01、 0x0b、0x00、0x03			NAK	Error = UNSUP_OBJ (0x03)
W	制御レジスタ	0x60				
R	ステータス・レジスタ	0x21				
W	制御レジスタ	0xa0				
R	ステータス・レジスタ	0xe1				
R	メッセージ・リード領域	0x01、0xfd、0x01、0x00、0x41、0x00、 0x0c、0x00	PROFIBUS DP-V1 (0xfd)	1	Get_Attribute	Attr = SW_REV (0x0c)
W	メッセージ・ライト領域	0x01、0xfd、0x01、0x00、0x81、0x01、 0x0c、0x00、0x03			NAK	Error = UNSUP_OBJ (0x03)
W	制御レジスタ	0x60				
R	ステータス・レジスタ	0x21				
W	制御レジスタ	0xa0				
R	ステータス・レジスタ	0xe1				
R	メッセージ・リード領域	0x00、0xfd、0x01、0x00、0x41、0x00、 0x0e、0x00	PROFIBUS DP-V1 (0xfd)	1	Get_Attribute	Attr = PROFILE_ID (0x0e)
W	メッセージ・ライト領域	0x00、0xfd、0x01、0x00、0x81、0x01、 0x0e、0x00、0x03			NAK	Error = UNSUP_OBJ (0x03)
W	制御レジスタ	0x60				
R	ステータス・レジスタ	0x21				
W	制御レジスタ	0xa0				
R	ステータス・レジスタ	0xe1				
R	メッセージ・リード領域	0x01、0xfd、0x01、0x00、0x41、0x00、0x0f、 0x00	PROFIBUS DP-V1 (0xfd)	1	Get_Attribute	Attr = PROFILE_SPEC_TYPE (0x0f)
W	メッセージ・ライト領域	0x01、0xfd、0x01、0x00、0x81、0x01、0x0f、 0x00、0x03			NAK	Error = UNSUP_OBJ (0x03)
W	制御レジスタ	0x60				
R	ステータス・レジスタ	0x21				
W	制御レジスタ	0xa0				
R	ステータス・レジスタ	0xe1				

R/W	DPRAM 領域	データ	宛先オブジェクト	インスタンス	コマンド/レスポンス	情報
R	メッセージ・リード領域	0x00、0xfd、0x01、0x00、0x41、0x00、 0x10、0x00	PROFIBUS DP-V1 (0xfd)	1	Get_Attribute	Attr = IM_VERSION (0x10)
W	メッセージ・ライト領域	0x00、0xfd、0x01、0x00、0x81、0x01、 0x10、0x00、0x03			NAK	Error = UNSUP_OBJ (0x03)
W	制御レジスタ	0x60				
R	ステータス・レジスタ	0x21				
W	制御レジスタ	0xa0				
R	ステータス・レジスタ	0xe1				
R	メッセージ・リード領域	0x01、0xfd、0x01、0x00、0x41、0x00、 0x11、0x00	PROFIBUS DP-V1 (0xfd)	1	Get_Attribute	Attr = IM_SUPPORTED (0x11)
W	メッセージ・ライト領域	0x01、0xfd、0x01、0x00、0x81、0x01、 0x11、0x00、0x03			NAK	Error = UNSUP_OBJ (0x03)
W	制御レジスタ	0x60				
R	ステータス・レジスタ	0x21				
W	制御レジスタ	0xa0				
R	ステータス・レジスタ	0xe1				
R	メッセージ・リード領域	0x00、0xfd、0x01、0x00、0x41、0x00、 0x12、0x00	PROFIBUS DP-V1 (0xfd)	1	Get_Attribute	Attr = IM_HEADER (0x12)
W	メッセージ・ライト領域	0x00、0xfd、0x01、0x00、0x81、0x01、 0x12、0x00、0x03			NAK	Error = UNSUP_OBJ (0x03)
W	制御レジスタ	0x60				
R	ステータス・レジスタ	0x21				
W	制御レジスタ	0xa0				
R	ステータス・レジスタ	0xa2				(Anybus State = WAIT_PROCESS)

サポート

Sales		Support	
HMS Sweden (Head Office)			
E-mail:	sales@hms.se	E-mail:	support@hms-networks.com
Phone:	+46 (0) 35 - 17 29 56	Phone:	+46 (0) 35 - 17 29 20
Fax:	+46 (0) 35 - 17 29 09	Fax:	+46 (0) 35 - 17 29 09
Online:	www.anybus.com	Online:	www.anybus.com
HMS North America			
E-mail:	us-sales@hms-networks.com	E-mail:	us-support@hms-networks.com
Phone:	+1-312 - 829 - 0601	Phone:	+1-312-829-0601
Toll Free:	+1-888-8-Anybus	Toll Free:	+1-888-8-Anybus
Fax:	+1-312-629-2869	Fax:	+1-312-629-2869
Online:	www.anybus.com	Online:	www.anybus.com
HMS Germany			
E-mail:	ge-sales@hms-networks.com	E-mail:	ge-support@hms-networks.com
Phone:	+49 (0) 721-96472-0	Phone:	+49 (0) 721-96472-0
Fax:	+49 (0) 721-96472-10	Fax:	+49 (0) 721-96472-10
Online:	www.anybus.de	Online:	www.anybus.de
HMS Japan			
E-mail:	jp-sales@hms-networks.com	E-mail:	jp-support@hms-networks.com
Phone:	+81 (0) 45-478-5340	Phone:	+81 (0) 45-478-5340
Fax:	+81 (0) 45-476-0315	Fax:	+81 (0) 45-476-0315
Online:	www.anybus.jp	Online:	www.anybus.jp
HMS China			
E-mail:	cn-sales@hms-networks.com	E-mail:	cn-support@hms-networks.com
Phone:	+86 (0) 10-8532-3183	Phone:	+86 (0) 10-8532-3023
Fax:	+86 (0) 10-8532-3209	Fax:	+86 (0) 10-8532-3209
Online:	www.anybus.cn	Online:	www.anybus.cn
HMS Italy			
E-mail:	it-sales@hms-networks.com	E-mail:	it-support@hms-networks.com
Phone:	+39 039 59662 27	Phone:	+39 039 59662 27
Fax:	+39 039 59662 31	Fax:	+39 039 59662 31
Online:	www.anybus.it	Online:	www.anybus.it
HMS France			
E-mail:	fr-sales@hms-networks.com	E-mail:	fr-support@hms-networks.com
Phone:	+33 (0) 3 68 368 034	Phone:	+33 (0) 3 68 368 033
Fax:	+33 (0) 3 68 368 031	Fax:	+33 (0) 3 68 368 031
Online:	www.anybus.fr	Online:	www.anybus.fr
HMS UK & Eire			
E-mail:	uk-sales@anybus.co.uk	E-mail:	support@hms-networks.com
Phone:	+44 (0) 1926 405599	Phone:	+46 (0) 35 - 17 29 20
Fax:	+44 (0) 1926 405522	Fax:	+46 (0) 35 - 17 29 09
Online:	www.anybus.co.uk	Online:	www.anybus.com
HMS Denmark			
E-mail:	info@anybus.dk	E-mail:	support@hms-networks.com
Phone:	+45 (0) 22 30 08 01	Phone:	+46 (0) 35 - 17 29 20
Fax:	+46 (0) 35 17 29 09	Fax:	+46 (0) 35 - 17 29 09
Online:	www.anybus.com	Online:	www.anybus.com
HMS India			
E-mail:	in-sales@anybus.com	E-mail:	in-support@hms-networks.com
Phone:	+91 (0) 20 40111201	Phone:	+46 (0) 35 - 17 29 20
Fax:	+91 (0) 20 40111105	Fax:	+46 (0) 35 - 17 29 09
Online:	www.anybus.com	Online:	www.anybus.com