



このマークは、オムロンの
CompoBus/D対応商品
を表しています。



OMRON

DeviceNet™ PCIボードスキャナ

形3G8F7-DRM21

ユーザーズマニュアル

おことわり

- (1) 本マニュアルの内容の一部または全部を無断で複写、複製、転載することを禁じます。
- (2) 本マニュアルの内容に関しては、改良のため予告なしに仕様などを変更することがありますので、予めご了承ください。
- (3) 本マニュアルの内容に関しては万全を期しておりますが、万一ご不審な点や誤りなどお気づきのことがありましたら、お手数ですが巻末記載の弊社支店または営業所までご連絡ください。その際、巻末記載のマニュアル No.も併せてお知らせください。

著作権・商標について

ODVA、CIP、CompoNet、DeviceNet、EtherNet/IP は ODVA の商標です。

Pentium は、Intel Corporation の商標です。

Microsoft、Windows、Windows 95、Windows 98、Windows Me、Windows NT、Windows 2000、Windows XP、Windows 7、Visual C++、Visual Basic は、米国マイクロソフト社の登録商標です。
Borland C++Builder は、米国ボーランドソフトウェア社の登録商標です。

その他、本文中に記載されている商品名および会社名は、それぞれ各社の商標または登録商標です。

本商品の著作権は、オムロン株式会社が有します。

はじめに

このたびは、DeviceNet PCI ボードをお買い上げいただきまして、ありがとうございます。

DeviceNet PCI ボードは、弊社の高度な制御技術と豊富な経験を元に開発された、DeviceNet のマスタの機能を持つ、PCI バス対応のボードです。

DeviceNet PCI ボードの機能や性能などを十分ご理解の上、正しくお使いください。

●対象となる読者の方々

本マニュアルは、次の方を対象に記述しています。

電気の知識（電気工事士あるいは同等の知識）を有する方で

- ・FA 機器の導入を担当される方
- ・FA システムを設計される方
- ・FA 現場を管理される方

●お願い

本マニュアルは、DeviceNet PCI ボードを使用する上で、必要な情報を記載しています。お使いになる前に本マニュアルをよく読んで、十分に理解してください。また、お読みになった後も本マニュアルは大切に保管して、いつも手元においてお使いください。

●ご承諾事項

「当社商品」について特別の合意がない場合には、お客様のご購入先にかかわらず、本ご承諾事項記載の条件を適用いたします。

●定義

本ご承諾事項中の用語の定義は次のとおりです。

- ・「当社商品」：「当社」の FA システム機器、汎用制御機器、センシング機器、電子・機構部品
- ・「カタログ等」：「当社商品」に関する、ベスト制御機器オムロン、電子・機構部品総合カタログ、その他のカタログ、仕様書、取扱説明書、マニュアル等であって電磁的方法で提供されるものも含みます。
- ・「利用条件等」：「カタログ等」に記載の、「当社商品」の利用条件、定格、性能、動作環境、取り扱い方法、利用上の注意、禁止事項その他
- ・「お客様用途」：「当社商品」のお客様におけるご利用方法であって、お客様が製造する部品、電子基板、機器、設備またはシステム等への「当社商品」の組み込み、または利用を含みます。
- ・「適合性等」：「お客様用途」での「当社商品」の(a)適合性、(b)動作、(c)第三者の知的財産の非侵害、(d)法令の遵守および(e)各種規格の遵守

●記載事項のご注意

「カタログ等」の記載内容については次の点をご理解ください。

- ・ 定格値および性能値は、単独試験における各条件のもとで得られた値であり、各定格値および性能値の複合条件のもとで得られる値を保証するものではありません。
- ・ 参考データはご参考として提供するもので、その範囲で常に正常に動作することを保証するものではありません。
- ・ 利用事例はご参考ですので、「当社」は「適合性等」について保証いたしかねます。
- ・ 「当社」は、改善や当社都合等により、「当社商品」の生産を中止し、または「当社商品」の仕様を変更することがあります。

●ご利用にあたってのご注意

ご採用およびご利用に際しては次の点をご理解ください。

- ・ 定格・性能ほか「利用条件等」を遵守しご利用ください。
- ・ お客様ご自身にて「適合性等」をご確認いただき、「当社商品」のご利用の可否をご判断ください。
「当社」は「適合性等」を一切保証いたしかねます。
- ・ 「当社商品」がお客様のシステム全体の中で意図した用途に対して、適切に配電・設置されていることをお客様ご自身で、必ず事前に確認してください。
- ・ 「当社商品」をご使用の際には、(i)定格および性能に対し余裕のある「当社商品」のご利用、冗長設計などの安全設計、(ii)「当社商品」が故障しても、「お客様用途」の危険を最小にする安全設計、(iii)利用者に危険を知らせるための、安全対策のシステム全体としての構築、(iv)「当社商品」および「お客様用途」の定期的な保守、の各事項を実施してください。
- ・ 「当社商品」は、一般工業製品向けの汎用品として設計製造されています。従いまして、次に掲げる用途での使用は意図しておらず、お客様が「当社商品」をこれらの用途に使用される際には、「当社」は「当社商品」に対して一切保証をいたしません。ただし、次に掲げる用途であっても「当社」の意図した商品用途の場合や特別の合意がある場合は除きます。
 - (a) 高い安全性が必要とされる用途（例：原子力制御設備、燃焼設備、航空・宇宙設備、鉄道設備、昇降設備、娯楽設備、医用機器、安全装置、その他生命・身体に危険が及ぶ用途）
 - (b) 高い信頼性が必要な用途（例：ガス・水道・電気等の供給システム、24 時間連続運転システム、決済システムほか権利・財産を取扱う用途など）
 - (c) 厳しい条件または環境での用途（例：屋外に設置する設備、化学的汚染を被る設備、電磁的妨害を被る設備、振動・衝撃を受ける設備など）
 - (d) 「カタログ等」に記載のない条件や環境での用途
- ・ 上記の(a)から(d)に記載されている他、「本カタログ等記載の商品」は自動車（二輪車含む。以下同じ）向けではありません。自動車に搭載する用途には利用しないでください。自動車搭載用商品については当社営業担当者にご相談ください。

●保証条件

「当社商品」の保証条件は次のとおりです。

- ・保証期間 ご購入後 1 年間といたします。
(ただし「カタログ等」に別途記載がある場合を除きます。)
- ・保証内容 故障した「当社商品」について、以下のいずれかを「当社」の任意の判断で実施します。
 - (a) 当社保守サービス拠点における故障した「当社商品」の無償修理
(ただし、電子・機構部品については、修理対応は行いません。)
 - (b) 故障した「当社商品」と同数の代替品の無償提供
- ・保証対象外 故障の原因が次のいずれかに該当する場合は、保証いたしません。
 - (a) 「当社商品」本来の使い方以外のご利用
 - (b) 「利用条件等」から外れたご利用
 - (c) 本ご承諾事項「ご利用にあたってのご注意」に反するご利用
 - (d) 「当社」以外による改造、修理による場合
 - (e) 「当社」以外の者によるソフトウェアプログラムによる場合
 - (f) 「当社」からの出荷時の科学・技術の水準では予見できなかった原因
 - (g) 上記のほか「当社」または「当社商品」以外の原因（天災等の不可抗力を含む）

●責任の制限

本ご承諾事項に記載の保証が、「当社商品」に関する保証のすべてです。

「当社商品」に関連して生じた損害について、「当社」および「当社商品」の販売店は責任を負いません。

●輸出管理

「当社商品」または技術資料を、輸出または非居住者に提供する場合は、安全保障貿易管理に関する日本および関係各国の法令・規制を遵守ください。お客様が法令・規則に違反する場合には、「当社商品」または技術資料をご提供できない場合があります。

安全上のご注意

●安全に使用していただくための表示と意味について

本マニュアルでは、DeviceNet PCI ボードを安全に使用していただくために、注意事項を次のような表示と図記号で示しています。ここで示した注意事項は、安全に関する重大な内容を記載しています。必ず守ってください。

表示と意味は次のとおりです。



正しい取扱いをしなければ、この危険のために、
軽傷・中程度の傷害を負ったり、万一の場合には
重傷や死亡に至る恐れがあります。
また、同様に重大な物的損害をもたらす恐れが
あります。



正しい取扱いをしなければ、この危険のために、
時に軽傷・中程度の傷害を負ったり、あるいは
物的損害を受ける恐れがあります。

安全上の要点

製品を安全に使用するために実施または回避すべきことを示します。

使用上の注意

製品が動作不能、誤動作、または性能・機能への悪影響を予防するために実施または回避すべきことを示します。

お願い

本文中の「お願い」は安全上の要点、使用上の注意と同等の内容を示します。

●図記号の説明



⊘ 記号は、禁止を意味しています。
具体的な内容は、⊘ の中と文章で示します。
左図の場合は、「分解禁止」を表します。



△ 記号は、注意（警告を含む）を意味しています。
具体的な内容は、△ の中と文章で示します。
左図の場合は、「一般的な注意」を表します。



● 記号は、強制を意味しています。
具体的な内容は、● の中と文章で示します。
左図の場合は、「一般的な強制事項」を表します。

警告

通電中は、ボードを分解したり内部に触れたりしないでください。
感電の恐れがあります。



プログラマブルコントローラ (PLC) の故障や外部要因による異常が発生した場合も、システム全体が安全側に働くように、PLC の外部で安全対策を施してください。

異常動作により、重大な事故につながる恐れがあります。

- ① 非常停止回路、インターロック回路、リミット回路など、安全保護に関する回路は、必ず PLC 外部の制御回路で構成してください。
- ② PLC は、自己診断機能で異常を検出したときや、運転停止故障診断 (FALS) 命令を実行したとき、運転を停止して全出力を OFF にします。ただし、PLC の自己診断機能では検出できない、入出力制御部や I/O メモリなどの異常時は、意図しない出力をすることがあります。これらいずれのときでも、システムが安全側に動作するよう、PLC 外部で対策を施してください。
- ③ 出力リレーの溶着や焼損、出力トランジスタの破壊などによって、PLC の出力が ON または OFF になったままになることがあります。このとき、システムが安全側に動作するよう、PLC 外部で対策を施してください。
- ④ PLC の DC24V 出力 (サービス電源) が過負荷の状態または短絡されると、電圧が低下し、出力は OFF となることがあります。このとき、システムが安全側に動作するよう、PLC 外部で対策を施してください。



運転を停止している状態 (「プログラム」モード) においても、CPU ユニットは、I/O リフレッシュを行っています。したがって、以下のいずれかの操作によって、出力ユニットに割り付けられた出力リレーエリアのデータ、または高機能 I/O ユニット / CPU 高機能ユニットに割り付けられた各リレーエリアのデータを変更する場合、十分に安全を確認してから行ってください。出力ユニットまたは高機能 I/O ユニット / CPU 高機能ユニットに接続された負荷が思いがけない動作をする恐れがあります。

- ・周辺ツール (パソコンツール) による、I/O メモリの CPU ユニットへの転送操作
- ・周辺ツールによる、現在値変更操作
- ・周辺ツールによる、強制セット / リセット操作
- ・メモ리카ードまたは EM ファイルメモリからの、I/O メモリファイルの CPU ユニットへの転送操作
- ・ネットワーク上の他の PLC または上位コンピュータからの、I/O メモリの転送操作



注意

他ノードへプログラムを転送するときや、I/O メモリを変更するときは、変更先ノードの安全を確認してから行ってください。
けがをする恐れがあります。



安全上の要点

DeviceNet PCI ボードを使用する際には、次のことに気を付けてください。

- 信号線の断線、瞬時停電による異常信号などに備えて、ご使用者側でフェールセーフ対策を施してください。
- 運転時には、必ずスキャンリスト有効モードで使用してください。
- 動作中のネットワークに新たなノードを追加する場合は、通信速度が一致していることを確認してから行ってください。
- 通信ケーブルの接続には、必ず指定ケーブルをご使用ください。
- 接続距離は、仕様の範囲内でご使用ください。
- 次のことを行うときは、パソコン本体の電源やスレーブの電源、通信用の電源をOFFにしてください。
 - ・ DeviceNet PCIボードを着脱するとき
 - ・ ロータリスイッチを設定するとき
 - ・ 装置を組み立てるとき
 - ・ ケーブルの接続、配線するとき
 - ・ コネクタを取り付けたり、取り外したりするとき
- 通信ケーブルコネクタなどロック機構のあるものは、必ずロックしていることを確認してから使用してください。
- コンピュータへのボード取付ネジ、コネクタのネジ、ケーブルのネジは、本マニュアルで指定した規定トルクで締めてください。
- 本マニュアルで指定した電源電圧で使用してください。
- 端子台、コネクタを十分確認してから、装着してください。
- 配線には、圧着端子を付けてください。撚り合わせただけの電線を直接、接続しないでください。
- 正しく配線してください。
- ケーブルの配線時には、以下の注意をお守りください。
 - ・ 通信ケーブルは、動力線、高圧線から離してください。
 - ・ 通信ケーブルを折り曲げないでください。
 - ・ 通信ケーブルを引っ張らないでください。
 - ・ 通信ケーブルに物を乗せないでください。
 - ・ 通信ケーブルは、必ずダクト内に配線してください。
- 接地された金属に触るなどして人体の静電気を放電されてから、PCI ボードに触れてください。
- 電源事情が悪い場所では、特に定格の電圧や周波数を供給できるようにしてご使用ください。
- 据え付け工事の際には、必ず D 種接地（第 3 種接地）をしてください。
- 外部配線の短絡に備えて、ブレーカなどの安全対策を施してください。

- 配線、スイッチなどの設定を十分に確認してから通電してください。
- 作成したユーザプログラム（ラダープログラムなど）は、十分な動作確認を行った後、本運転に移行してください。
- 運転再開に必要なデータメモリや保持リレーの内容、パラメータおよびデータは、交換した CPU ユニット・高機能 I/O ユニットに転送してから運転を再開してください。
- 運搬や保存時は、LSI、IC などの静電気による破壊防止のため、回路基板を導電性のもので覆い、保存温度範囲に保ってください。
- 回路基板には、電気部品のリードなど鋭利な部分がありますので、部品実装部や基盤の裏面に直接手を触れないでください。
- PCI ボードを輸送するときは、専用の梱包箱を使用してください。また、輸送中に過度な振動や衝撃が加えられないように注意してください。
- 本製品を、分解して修理や改造をしないでください。
- DeviceNet PCI ボードスキャナを使用中は、パソコンのスタンバイ／スリープ／休止機能は使用しないでください。本製品を使用中にパソコンがスタンバイ／スリープ／休止状態になった場合は、通信の切断やその他、予期しないエラーが発生することがあります。
- DeviceNet PCI ボードスキャナは、パソコンのスタンバイ／スリープ／休止機能には対応していません。DeviceNet PCI ボードスキャナを使用中はパソコンのスタンバイ／スリープ／休止機能を使用しないでください。

使用上の注意

- 本マニュアルに示すとおり、正しく設置してください。
- 次のような環境には設置しないでください。
 - ・ 日光が直接当る場所
 - ・ 周囲温度や相対湿度が仕様値の範囲を超える場所
 - ・ 温度の変化が急激で結露するような場所
 - ・ 腐蝕性ガス、可燃性ガスのある場所
 - ・ ちり、ほこり、塩分、鉄粉が多い場所
 - ・ 水、油、薬品などの飛沫がかかる場所
 - ・ 本体に直接振動や衝撃が伝わる場所
- 次のような場所で使用する際は、遮蔽対策を十分に行ってください。
 - ・ 静電気などによるノイズが発生する場所
 - ・ 強い電界や磁界が生じる場所
 - ・ 放射能を被曝する恐れのある場所
 - ・ 電源線が近くを通る場所

EC 指令への適合について

●対応指令

EMC 指令

●対応の考え方

EMC 指令：オムロンの EC 指令適応商品は、各種機械、製造装置に組み込まれ使用される電気機器であるため、組み込んだ機械・装置がより容易に EMC 規格に適合できるように、関連する EMC 規格（注1）の適合を図っています。

よって、商品自身についての EMC 規格への適合性については確認できませんが、お客様の使用状態での適合性確認はできません。EMC の性能は EC 指令適応商品を組み込んだ機器・制御盤の構成、配線状態、配置状態などにより変化しますので、機械・装置全体での最終的な EMC 適合性の確認を、お客様自身で実施していただくようお願いいたします。

注1：EMC(Electro-Magnetic Compatibility：電磁環境両立性)関連規格のうち、EMS(Electro-Magnetic Susceptibility：電磁感受性)に関しては、EN61000-6-2、EMI(Electro-Magnetic Interference：電磁妨害)に関しては EN61000-6-4 によります。

●EC 指令への適合について

DeviceNet の EC 指令適合品は、設置するにあたり、以下の注意が必要です。

1. 通信電源、内部電源、I/O 電源として使用する DC 電源は、強化絶縁、または二重絶縁されたものを使用してください。
2. DeviceNet の EC 指令適合品は、共通エミッション規格（EN61000-6-4）に適合していますが、特に Radiated emission（10m 法）に関しては、ご使用になる制御盤の構成、接続される他の機器との関係、配線等により変化することがあります。したがって、EC 指令適合品である DeviceNet をご使用の場合でも、お客様にて機械・装置全体で EC 指令適合性を確認・対応していただく必要があります。

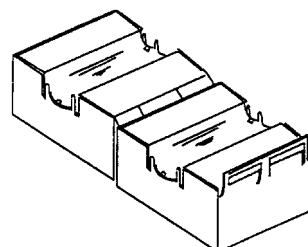
下記対策例は、ノイズを減衰するための一例です。

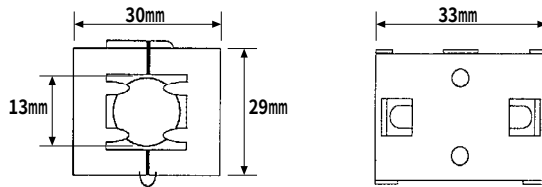
[対策例]

- (1) DeviceNet PCI ボードから 10cm 以内の通信ケーブルにフェライトコアを装着すると、通信ケーブルからの放射ノイズを減衰させることができます。

フェライトコア（データラインフィルタ）：0443-164151（日辰電機製作所製）

インピーダンス仕様	
25MHz	100MHz
156 Ω	250 Ω





連絡先

株式会社 日辰電機製作所 第三営業部

TEL 04-2934-4151

FAX 04-2934-4155

- (2) DeviceNet 通信ケーブルは、できるだけ短い電線で、確実に D 種接地（第 3 種接地）してください。

構成品の確認

ご購入いただいた商品の内容物をご確認ください。

- ・ 専用 PCI ボード（通信コネクタ 1 個付属） : 1 枚
- ・ Scanner SDK のインストールディスク（CD-ROM） : 1 枚
- ・ オペレーションマニュアル（本書） : 1 冊
- ・ ユーザ登録はがき（ソフトウェア使用許諾契約書と兼用） : 1 枚

ソフトウェアの使用および保証条件

弊社では、下記の「使用および保証条件」に従って、ソフトウェア製品のご使用をお願い申し上げております。同封の「ソフトウェアユーザ登録票」に基づいて、お客様へのアフターサービスを実施させていただきます。

●使用および保証条件

(詳細については別紙ソフトウェア使用許諾契約書をご覧ください)

1 著作権について

このソフトウェア（同封の記録媒体、マニュアル等の文書に記録、記載された内容を意味します）の著作権はオムロン株式会社にあります。

2 コピーおよび変更、改良について

- (1) ソフトウェアの保存（バックアップ）および次項(2)による変更、改造の目的以外のためにこのソフトウェアの全部または一部をコピーしないでください。
- (2) お客様ご自身にご利用になる場合のみ、このソフトウェアの変更、改造を行うことができます。ただし、お客様の変更、改造により何らかの欠陥が生じた場合、および、このような変更、改造の結果、お客様に損害が生じた場合につきましては、弊社は責任を負いかねます。

3 保証およびアフターサービスについて

- (1) 万一、弊社の責任によりこのソフトウェアが正常に機能しない場合は、無償で修理、または交換いたします。
- (2) 前項(1)の保証は、この製品のご購入日より1年間とさせていただきます。
- (3) 「ソフトウェアユーザ登録票」をご返送していただいたお客様には、弊社が知り得たプログラム上の誤り（バグ）や製品改良（バージョンアップ）など、必要な情報をお知らせします。
- (4) このソフトウェアのご使用により、万一、お客様に損害が生じた場合におきましても、本書に定める以外の責任は負いかねます。

4 第三者による使用について

このソフトウェアまたはそのコピーを貸与、譲渡その他の方法による第三者の使用に関しての権利は許諾しないものとします。

マニュアル改訂履歴

マニュアル改訂記号は、表表紙および裏表紙の左下に記載されている Man.No.の語尾に付記されています。

Man.No. SBCD-315F

↑
改訂記号

改訂記号	改訂日	改ページ・内容
—	2000 年 8 月	初版印刷
B	2004 年 5 月	安全に関するご注意の追加および変更、構成や表現の改善、内容追加に伴う全面改訂
C	2004 年 9 月	付録を削除、説明追加、誤り修正に伴う改訂
D	2005 年 7 月	説明追加、誤り修正に伴う改訂
E	2006 年 12 月	説明追加、誤り修正に伴う改訂
F	2013 年 9 月	・ Windows 7 対応に関する記述の追加 ・ 他マニュアルとの情報整合に伴う改訂

マニュアル構成

DeviceNet PCI ボードの関連マニュアルは、下表のように構成されています。併せてご覧ください。

Man.No.	形式	マニュアル名称	用途
SBCD-315 (本書)	形 3G8F7-DRM21	DeviceNet™ PCI ボードスキャナ ユーザーズマニュアル	PCI ボードについて知りたいとき
SCCC-308		DeviceNet™ ユーザーズマニュアル	DeviceNet 通信ネットワーク共通の通信仕様、配線方法について知りたいとき
SBCD-316	形 WS02-CFDC1-J 形 3G8F5-DRM21 形 3G8E2-DRM21	DeviceNet™ コンフィグレータ オペレーションマニュアル	コンフィグレータを使用するとき
SBCD-314	形 CS1W-DRM21 形 CJ1W-DRM21	CS/CJ シリーズ DeviceNet™ ユニットの ユーザーズマニュアル	CS/CJ シリーズ DeviceNet ユニットの について知りたいとき
SBCD-312	形 C200HW-DRM21-V1 形 CVM1-DRM21-V1	DeviceNet™ マスタユニット ユーザーズマニュアル	C200H 用、CVM1/CV 用 DeviceNet マスタユニットについて 知りたいとき
SBCD-324	形 DRT2 シリーズ	DeviceNet™ スレーブマニュアル	DRT2 シリーズのスレーブについて 知りたいとき
SBCD-305	形 C200HW-DRT21 形 CQM1-DRT21 形 DRT1 シリーズ	DeviceNet™ スレーブマニュアル	C200H 用、CQM1 用 I/O リンク ユニットと、DRT1 シリーズの スレーブについて知りたいとき
SBCD-306	形 DRT1-COM 形 GT1 シリーズ	DeviceNet™ マルチプル I/O ターミナル マニュアル	スレーブの一種であるマルチプル I/O ターミナルについて知りたい とき

上記のマニュアルの他に、ご使用のパソコンのマニュアルも併せてご覧ください。

また、DeviceNet ネットワーク関連の製品として、DeviceNet コンフィグレータもあります。

本書の構成

第1章 概要

1

DeviceNet PCI ボード（以下、「本ボード」）の機能の概要や、システム構成、仕様、各部の機能などについて説明しています。初めてご使用になるときは、必ずお読みください。

第2章 インストール

2

本ボードのハードウェアの設定・装着方法や、本ボードを使用するためのドライバ・Scanner SDK のインストール方法、通信ケーブルの接続方法について説明しています。
「DeviceNet ユーザーズマニュアル」（SCCC-308）と併せてご覧ください。

第3章 API 関数の使用手順

3

本ボードの制御に使用する API 関数の使用手順を、機能ごとに説明しています。API 関数を使用する上での注意点についても記載しています。

第4章 API 関数リファレンス

4

本ボードを制御する API 関数の使用方法について説明しています。本ボードを制御するアプリケーションソフトは、API 関数について十分ご理解の上で作成してください。

第5章 サンプルプログラム

5

本ボードを使用する、サンプルプログラムを記載しています。アプリケーションソフトウェア開発の参考としてください。

第6章 通信タイミング

6

リモート I/O 通信機能やメッセージ通信機能使用時の、応答時間や伝送遅延時間など、通信の詳細について説明しています。厳密な入出力タイミングが必要な場合の参考としてください。

第7章 異常時の処理

7

本ボード使用中に発生する様々な異常と、それに対する処置について説明しています。異常発生時にご活用ください。異常の予防のためにも、初めに一度ご覧になることをお勧めします。

目次

はじめに.....	1
安全上のご注意.....	4
安全上の要点.....	7
使用上の注意.....	9
EC 指令への適合について	10
構成品の確認.....	12
ソフトウェアの使用および保証条件	13
マニュアル改訂履歴	14
マニュアル構成.....	15
本書の構成.....	17

第1章 概要

1-1 製品構成	1-1
1-2 DeviceNet PCI ボードについて.....	1-1
1-3 Scanner SDK の特長.....	1-3
1-4 Scanner SDK の機能.....	1-5
■I/O 通信機能	1-5
■メッセージ通信機能	1-6
■保守機能.....	1-7
1-5 システム構成.....	1-8
■通信速度と通信距離	1-8
■スレーブ機器の接続方法	1-8
1-6 仕様	1-9
■DeviceNet PCI ボード一般仕様.....	1-9
■DeviceNet 通信仕様.....	1-9
■Scanner SDK 通信仕様.....	1-10
■動作環境.....	1-10
■開発環境.....	1-11
■外形寸法.....	1-11
1-7 各部の名称と機能.....	1-12
■LED 表示の意味.....	1-12
■ボード ID	1-13
1-8 使用までの手順.....	1-14

第2章 インストール

2-1 インストールの手順.....	2-1
2-2 パソコンへの装着.....	2-2
■作業前の確認.....	2-2
■ボード ID の設定.....	2-3
■パソコンへの装着.....	2-3

2-3	ドライバのインストール	2-5
	■Windows 98 の場合.....	2-5
	■Windows XP の場合.....	2-8
	■Windows 7 の場合.....	2-10
2-4	Scanner SDK のインストール	2-14
	■インストール方法	2-14
	■インストールファイル	2-18
	■アンインストールについて.....	2-19
2-5	DeviceNet への接続.....	2-21
	■通信ケーブルの加工とコネクタへの取り付け.....	2-21
	■通信ケーブルの接続	2-23

第 3 章 API 関数の使用手順

3-1	アプリケーション開発環境.....	3-1
	■推奨する開発環境	3-1
	■他の開発環境での注意事項.....	3-1
3-2	API 関数の種類	3-2
	■ボード管理用 API.....	3-2
	■マスタ機能用 API.....	3-2
	■スレーブ機能用 API.....	3-3
	■Explicit メッセージ用 API	3-3
	■保守用 API	3-4
3-3	イベントのチェック方法	3-5
	■イベントの種類	3-5
	■チェック方法.....	3-5
3-4	エラーのチェック方法	3-6
	■関数の戻り値によるエラーのチェック	3-6
3-5	パラメータについて	3-7
	■パラメータの種類	3-7
	■パラメータ操作 API 関数について.....	3-8
3-6	I/O 通信機能の使用手順.....	3-9
	■マスタ機能の使用手順	3-9
	■スレーブ機能の使用手順.....	3-10
3-7	Explicit メッセージクライアント機能の使用手順	3-11
	■Windows メッセージによるイベント通知を使用する場合.....	3-11
	■ポーリングによるイベントチェックを使用する場合	3-12
3-8	Explicit メッセージサーバ機能の使用手順	3-13
	■Windows メッセージによるイベント通知を使用する場合.....	3-13
	■ポーリングによるイベントチェックを使用する場合	3-14
3-9	リセット機能の使い方	3-15
3-10	異常履歴機能の使い方	3-16
3-11	PC ウォッチドッグタイマ管理機能の使い方	3-17

第4章 API 関数リファレンス

4-1	関数一覧	4-1
	■ボード管理用 API	4-1
	■マスタ機能用 API	4-2
	■スレーブ機能用 API.....	4-3
	■Explicit メッセージ用 API.....	4-4
	■保守用 API	4-5
4-2	ボードの状態.....	4-6
4-3	ボード管理用 API	4-7
	■ボードサービス API.....	4-7
	■PC ボード割り込みサービス API	4-11
4-4	マスタ機能用 API	4-14
	■スキャンリスト操作 API.....	4-14
	■I/O 通信サービス API	4-21
	■I/O データアクセスサービス API.....	4-26
4-5	スレーブ機能用 API	4-29
	■スレーブスキャンリスト操作 API	4-29
	■I/O 通信サービス API	4-32
	■I/O データアクセスサービス API.....	4-33
4-6	Explicit メッセージ用 API.....	4-36
	■メッセージ監視タイマサービス API	4-36
	■クライアントメッセージサービス API	4-38
	■サーバメッセージサービス API	4-42
4-7	保守用 API.....	4-47
	■ステータスサービス API.....	4-47
	■異常履歴アクセスサービス API	4-55
	■PC ウォッチドッグタイマサービス API.....	4-57

第5章 サンプルプログラム

5-1	サンプルプログラムについて	5-1
5-2	DeviceNet Scanner Demo の使い方	5-2
	■DeviceNet Scanner Demo の機能	5-2
	■I/O 通信を行う場合の使用例	5-2
	■Explicit メッセージ通信を行う場合の使用例	5-10

第6章 通信タイミング

6-1	リモート I/O 通信の性能.....	6-1
	■通信サイクル時間.....	6-1
	■1 ネットワーク内にマスタが複数台の場合	6-4

第7章 異常時の処置

7-1	LED 表示と異常時の処置	7-1
	■正常動作中の LED 表示	7-1
	■異常発生時の LED 表示と処置.....	7-2

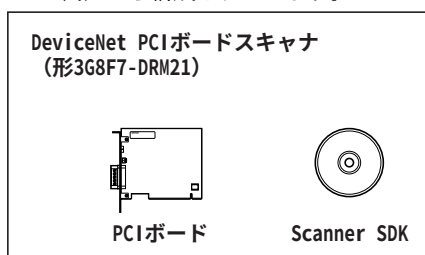
7-2	関数で検知される異常と処置.....	7-4
	■エラーコード	7-4
7-3	異常履歴機能.....	7-6
	■異常履歴データ	7-6
	■異常コード一覧	7-7

第1章 概要

この章では、DeviceNet Scanner SDK の機能や、仕様、機器の構成などについて説明します。

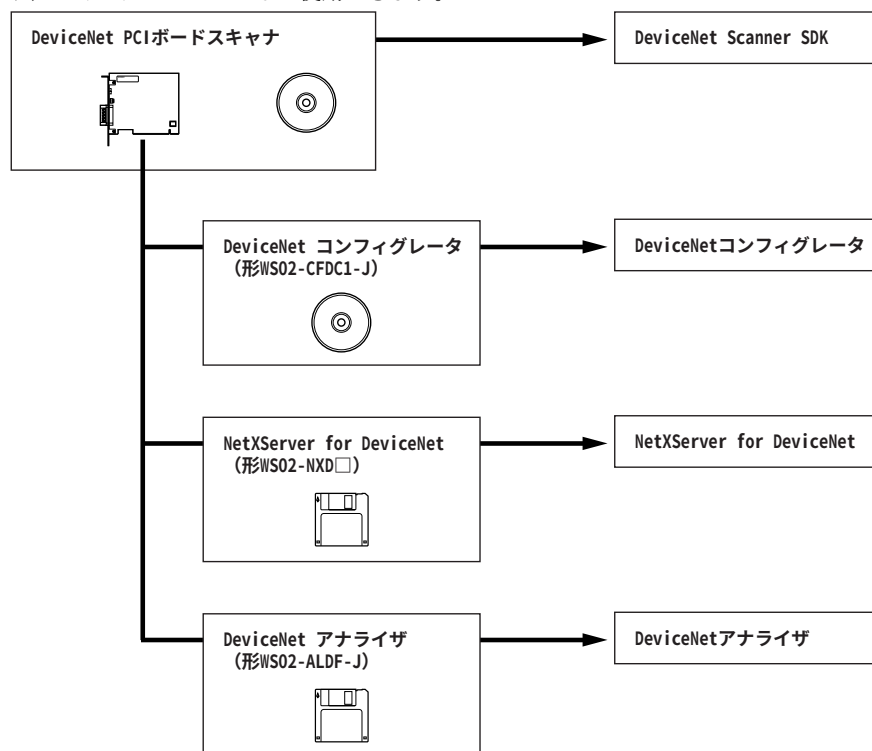
1-1 製品構成

本製品 形 3G8F7-DRM21 は、PCI ボード（ハード）と Scanner SDK（ソフト、添付 CD-ROM 内）から構成されています。



1-2 DeviceNet PCI ボードについて

PCI ボード自身は、他のソフトウェア（コンフィグレータ、ネット X サーバ、アナライザ）のインタフェースとして使用できます。



●DeviceNet Scanner SDK

DeviceNet Scanner SDK（本製品）は、DeviceNet のマスタやスレーブとして動作するアプリケーションを開発するための、Windows 環境でのライブラリです。DLL として提供されています。

Scanner SDK を使用すれば、業界トップクラスの性能と機能を持ったマスタ/スレーブのアプリケーションを開発することができます。

●DeviceNet コンフィグレータ

DeviceNet コンフィグレータは、DeviceNet の構築をサポートするための Windows アプリケーションです。オムロン製マスタ/スレーブのパラメータ設定やモニタリングだけでなく、EDS ファイルをインストールすることで他社製スレーブのパラメータ設定もできます。

設計からメンテナンスまで、ネットワークの管理を幅広くサポートします。

●NetXServer for DeviceNet

NetXServer は DeviceNet 上の I/O データを収集し、そのデータをモニタリングアプリケーションなどに提供する、Windows 環境でのミドルウェアです。DDE サーバーとして動作します。

NetXServer を使用することで、マスタとスレーブの通信に影響を与えることなく、I/O データをモニタできます。

NetXServer には、次の 2 種類があります。

DDE Edition：DDE クライアント（Microsoft Excel など）による、I/O データのモニタリングが可能なタイプ

SDK Edition：NetXServer の機能を使用したモニタリングアプリケーションを開発するためのライブラリタイプ

●DeviceNet アナライザ

DeviceNet アナライザは、DeviceNet 上のメッセージフレームを解析するための Windows アプリケーションです。

ネットワーク上を流れるメッセージフレームを表示したり、トラフィック状態を表示することができます。トラブル発生時の原因調査や、DeviceNet 対応機器の開発に使用できます。

1-3 Scanner SDK の特長

Scanner SDK は、通信機能として、以下の機能を持っています。

- DeviceNet 上のノードと I/O データを交換する I/O 通信機能
 - DeviceNet マスタ機能
 - DeviceNet スレーブ機能
- DeviceNet Explicit メッセージのクライアント機能・サーバ機能

Scanner SDK には、上記の機能の他に、ノード（マスタ・スレーブ）やネットワークの状態の読み出しや、異常の内容・発生日時を記録する異常履歴機能があります。

参 考

- DeviceNet ではケーブル 1 本で、遠隔地で分散しているスレーブ（子局）とデータの入出力を行うことができます。また、メッセージを送受信することで、スレーブや他のマスタを制御・モニタすることができます。詳細は、「DeviceNet ユーザーズマニュアル」（SCCC-308 □）を参照してください。
- このマニュアルでは、メッセージを発行することでサービスを要求する側を「クライアント」、メッセージを受け取り、メッセージに応じた処理や返信を行う側を「サーバ」と呼びます。

Scanner SDK には、次のような特長があります。

●DeviceNet スレーブのデータの入出力

DeviceNet 上のスレーブの入出力（接点）の状態が、本ボード内に反映されます。I/O データ読み書き用の関数を呼び出すことで、指定したスレーブと入出力を行えます。

●DeviceNet 準拠によるマルチベンダ対応

DeviceNet は、世界的に標準化が図られている DeviceNet（デバイスネット）に準拠しているため、国内外のさまざまなメーカーの DeviceNet 対応スレーブを接続することができます。

●最大 63 台のスレーブ・最大 37,800 バイトの I/O データ入出力

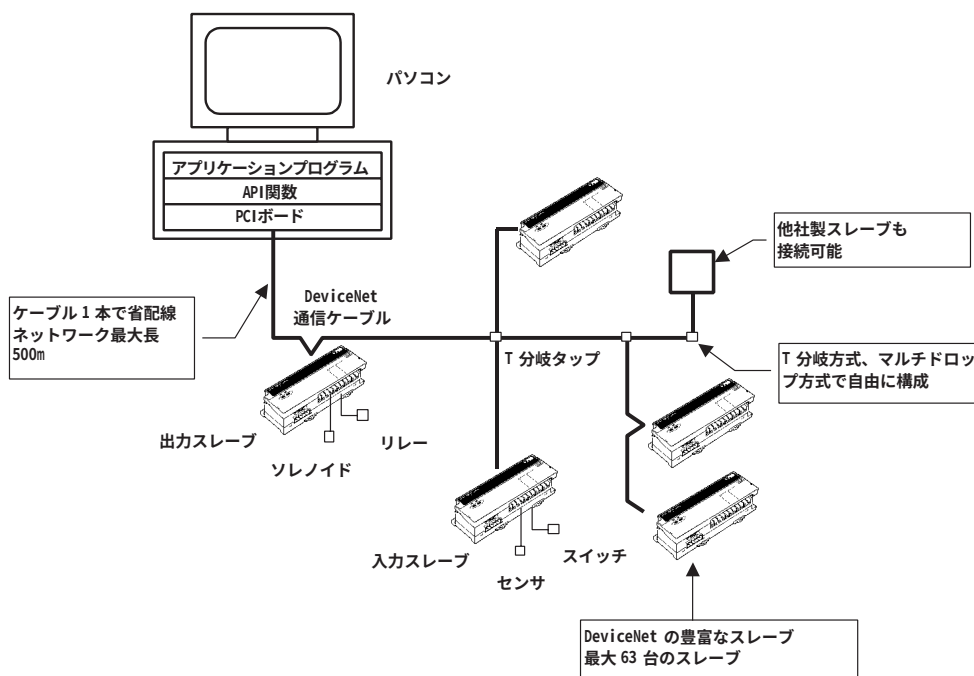
Scanner SDK を使用すれば、最大 63 台のスレーブによる、37,800 バイト（IN：25,200 バイト、OUT：12,600 バイト）の I/O データを入出力することができます。

●API 関数による制御

Scanner SDK の機能は、すべて API 関数で提供されます。この API 関数を使用して、ユーザアプリケーションを作成します。

●Windows メッセージまたはポーリングによるイベントチェック

イベントのチェックには、Windows メッセージによる自動的な通知と、ユーザアプリケーションによるポーリングの 2 種類を使用でき、用途に合わせた、柔軟なアプリケーションを作成できます。



1-4 Scanner SDK の機能

■I/O 通信機能

マスタ機能

Scanner SDK のマスタ機能では、IN 用のエリア 200 バイト（100CH=1600 点）×2 個と、OUT 用のエリア 200 バイト（100CH=1600 点）×1 個がスレーブ単位で用意されています。

I/O 通信は、Scanner SDK に登録したスキャンリストに従って実行されます。スキャンリストには、各スレーブの IN・OUT バイト数などの情報が登録されています。

●最大入出力点数・最大接続スレーブ数

Scanner SDK のマスタ機能で通信できる最大入出力点数、最大接続スレーブ数、最大 I/O コネクション数は、次のようになっています。

項目	内容
最大入出力点数	IN : 25200 バイト (12600CH=201600 点) OUT : 12600 バイト (6300CH=100800 点)
1 スレーブ当たりの最大入出力点数	IN : 200 バイト×2 (100CH×2=1600 点×2) OUT : 200 バイト (100CH=1600 点)
1 スレーブ当たりの最大 I/O コネクション数	最大 2
最大接続スレーブ数	63 台 (ノードアドレス 0~63)

参考

1 スレーブ当たり 2 個の IN エリアが用意されていますが、通常は 1 個目の IN エリアを使用します。同時に 2 つのコネクションを使用した場合、2 個目の IN エリアを使用することが可能です。

スレーブ機能

Scanner SDK のスレーブ機能では、IN 用のエリア 200 バイト（100CH=1600 点）×2 個と、OUT 用のエリア 200 バイト（100CH=1600 点）×1 個が用意されています。

スレーブスキャンリストへのマスタ登録は、以下のいずれかの方法で行います。

- (1) 関数によって個別／一括で登録する
- (2) オムロン製 DeviceNet コンフィグレータで作成したパラメータファイルを指定して、一括登録する

スレーブとして動作させるためには、Scanner SDK にスレーブスキャンリストを登録する必要があります。

●最大入出力点数・最大接続マスタ数

Scanner SDK のスレーブ機能で通信できる最大入出力点数は、次のようになっています。

項目	内容
最大入出力点数	IN : 200 バイト×2 (100CH×2=1600 点×2) OUT : 200 バイト (100CH=1600 点)
最大接続マスタ数	1 台

参 考

2 個の IN エリアが用意されていますが、通常は 1 個目の IN エリアを使用します。同時に 2 つのコネクションを使用した場合、2 個目の IN エリアを使用することが可能です。

■メッセージ通信機能

Explicit メッセージ通信機能

クライアントとして、必要に応じて Explicit メッセージを発行することで、他ノードの制御やモニタを行えます。

また、サーバとして、他ノードからの Explicit メッセージを受け取ることもできます（処理やレスポンスは、ユーザアプリケーションにて対応してください）。

Explicit メッセージ通信を使用することで、他社製の DeviceNet 対応製品との間で自由に通信を行うことができます。

●最大コネクション数

本ボードと接続できる最大コネクション数は、次のようになっています。

項目	内容
最大クライアントコネクション数	63 コネクション (1 サーバ当たり 1 コネクション)
最大サーバコネクション数	4 コネクション (1 クライアント当たり 1 コネクション)

■保守機能

ステータス読み出し機能

本ボードでは、ノード（マスタ・スレーブ）やネットワークの稼働状態や設定状態などの、次のような情報を読み出すことができます。

- ・ Scanner SDK の DLL のバージョン
- ・ 本ボード用ドライバのバージョン
- ・ 本ボードの有無
- ・ ネットワークステータス
- ・ ネットワークへの加入状態・リモート I/O 通信の稼働状態
- ・ 通信ステータス
- ・ スレーブごとのスキャンリストへの登録の有無
- ・ スレーブごとのデバイスステータス

リセット機能

本ボードを、パソコン側からの指示によってリセット（初期化）できます。

通信サイクルタイム管理機能

通信サイクルタイム（スレーブの入出力を読み書きする間隔）の設定や、最大値・最小値の読み出し・クリアを行うことができます。

異常履歴機能

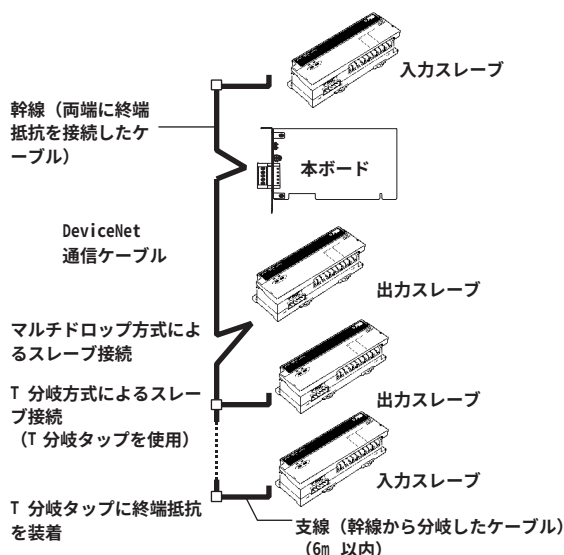
本ボードには、運転中に発生した異常の内容を記録する異常履歴機能があります。異常内容の詳細を調べることができますので、迅速な処置に有効です。

PC ウォッチドッグタイマ管理機能

本ボードを制御するアプリケーションが、何らかの理由により停止したときに、自動的にリモート I/O を停止させることができます。

パソコン（アプリケーション）側からは、定期的に本ボードの PC ウォッチドッグタイマを更新することで、アプリケーションが正常に動作していることを、本ボードに通知します。

1-5 システム構成



参考

スレーブについての詳細は、次のマニュアルを参照してください。

- ・「DeviceNet スレーブマニュアル」（形 DRT2 シリーズ）（SBCD-324□）
- ・「DeviceNet スレーブマニュアル」（形 C200HW-DRT21、形 CQM1-DRT21、形 DRT1 シリーズ）（SBCD-305□）
- ・「DeviceNet マルチプル I/O マニュアル」（形 DRT1-COM、形 GT1 シリーズ）（SBCD-306□）

通信速度と通信距離

DeviceNet では、通信速度によって通信距離が次のようになります。

通信速度	ネットワーク最大長		支線長	総支線長
	太いケーブル	細いケーブル		
500k ビット/s	100m 以下	100m 以下	6m 以下	39m 以下
250k ビット/s	250m 以下			78m 以下
125k ビット/s	500m 以下			156m 以下

スレーブ機器の接続方法

スレーブ機器は、次の 2 通りの方法で接続できます。

T 分岐方式：幹線または支線から T 分岐タップによって分岐させた支線にスレーブを接続する方式

マルチドロップ方式：幹線または支線に直接スレーブを接続する方式

T 分岐方式とマルチドロップ方式は、自由に混在させることができます。

お願い

接続方法や、接地のしかたについての詳細は、「DeviceNet ユーザーズマニュアル」（SCCC-308□）を参照してください。

1-6 仕様

■DeviceNet PCI ボード一般仕様

項目	仕様
外形寸法	119.9(W)×106.7(H)mm
動作電圧	DC5V±5% DC3.3V 未使用
消費電流	内部電源 DC5V 290mA 以下 通信電源 DC24V 30mA 以下
耐振動	JIS C0040 に準拠 10～57Hz 振幅 0.075mm、57～150Hz 加速度 9.8m/s ² X、Y、Z 各方向 80 分（掃引時間 8 分×掃引回数=合計 80 分）（DIN レール取付時 2～55Hz 2.94m/s ² X、Y、Z 各方向 20 分）
耐衝撃	JIS C0041 に準拠 147m/s ² X、Y、Z 各方向 3 回）
使用周囲温度	0～55℃
使用周囲湿度	10～90%RH（結露のしないこと）
使用周囲雰囲気	腐食性ガスのないこと
保存周囲温度	－20～60℃
質量	91g 以下
最大装着枚数	3 枚

DeviceNet PCI ボードは、PCI Local Bus Specification Revision 2.0 に対応しています。

■DeviceNet 通信仕様

項目	仕様			
通信方式	DeviceNet（デバイスネット）通信規約準拠			
接続形態	マルチドロップ方式、T 分岐方式の組み合わせが可能（幹線および支線に対して） （幹線の両端に終端抵抗が必要です。）			
通信速度	500k / 250k / 125k ビット/s（SCAN_Online()関数で指定）			
通信距離	通信速度	ネットワーク最大長	支線長	総支線長
	500k ビット/s	100m 以下	6m 以下	39m 以下
	250k ビット/s	250m 以下 *1	6m 以下	78m 以下
	125k ビット/s	500m 以下 *1	6m 以下	156m 以下
通信電源	DC11～24V 30mA（通信コネクタより供給）			
最大接続スレーブ数	63 台			
誤り制御	CRC エラー、ノードアドレス重複チェック、スキャンリストの照合			
ケーブル	5 線（信号系 2 本、電源系 2 本、シールド 1 本）			

*1：太いケーブルを使用した場合の距離です。細いケーブルを使用する場合は、ネットワーク最大長を 100m 以内にしてください。また、混在させる場合など、ネットワーク構成についての詳細は、「DeviceNet ユーザーズ マニュアル」（SCCC-308□）を参照してください。

■Scanner SDK 通信仕様

項目	仕様
サポートする I/O コネクション	• Bit Strobe • Polling • Cyclic • Change of State(COS) • Explicit Peer to Peer Messaging
通信サイクルタイム *1	2～500ms（API 関数で指定可能）
Explicit クライアントとして同時に 通信可能なサーバ数	63 ノード
Explicit サーバとして同時に 通信可能なクライアント数	4 ノード
Explicit メッセージの送受信データ長 （サービスデータ長）	クライアント - Explicit リクエストメッセージ : 552 バイト - Explicit レスポンスメッセージ : 552 バイト
	サーバ - Explicit リクエストメッセージ : 552 バイト - Explicit レスポンスメッセージ : 552 バイト
Explicit メッセージのレスポンス監視 時間（クライアント時）	2 秒（デフォルト）（API 関数で指定可能）
Explicit メッセージの再送	なし（ユーザアプリケーションによる対応が必要）

*1： マスタがあるスレーブに対してリモート I/O 通信を実行したときから、再び同じスレーブに対してリモート I/O 通信を行うまでの最大時間

■動作環境

ハードウェア

DOS/V 対応パソコン

- ・ 1 つ以上の PCI バススロット（PCI バス Rev.2.0 以降）
- ・ ハードディスク空き容量
5MB 以上（ユーザプログラムを除く）
- ・ CD-ROM ドライブ
ソフトウェアインストール時に、1 台必要
- ・ ディスプレイ
VGA 以上の表示機能

推奨 CPU と推奨メモリ容量、および上記以外の項目については、ご使用の OS に準じます。

動作 OS

Microsoft Windows 95 / 98 / Me / NT 4.0 / 2000 / XP / Windows 7
Windows 3.1、Windows NT 3.5 はサポートしていません。

■開発環境

推奨開発環境

- ・ Microsoft Visual C++ (Ver.6.0 以降)

他の開発環境

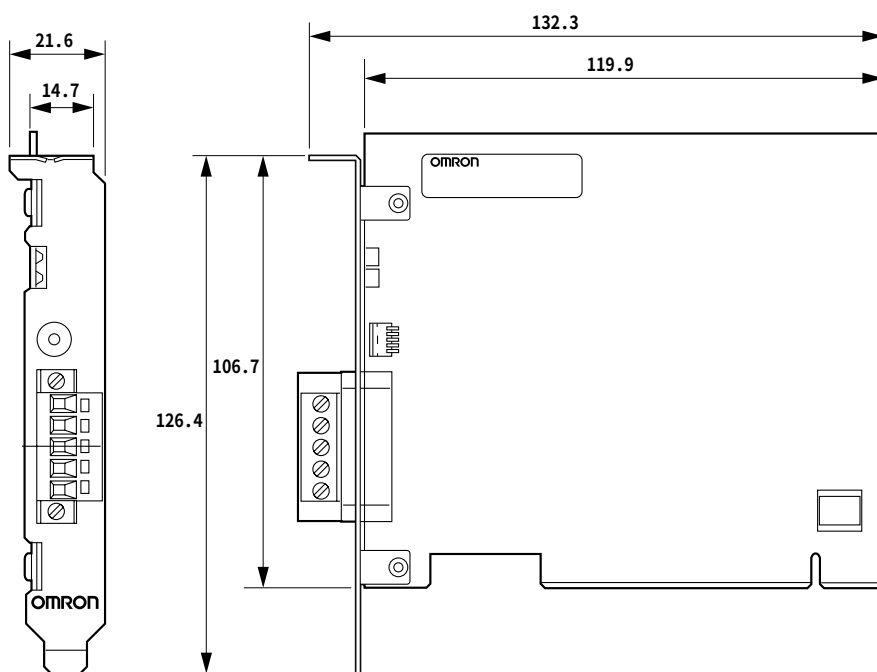
- ・ Microsoft Visual Basic

使用可能ですが、一部機能に制限があります。詳細については、「他の開発環境での注意事項」(P.3-1)を参照してください。

- ・ Borland C++ Builder

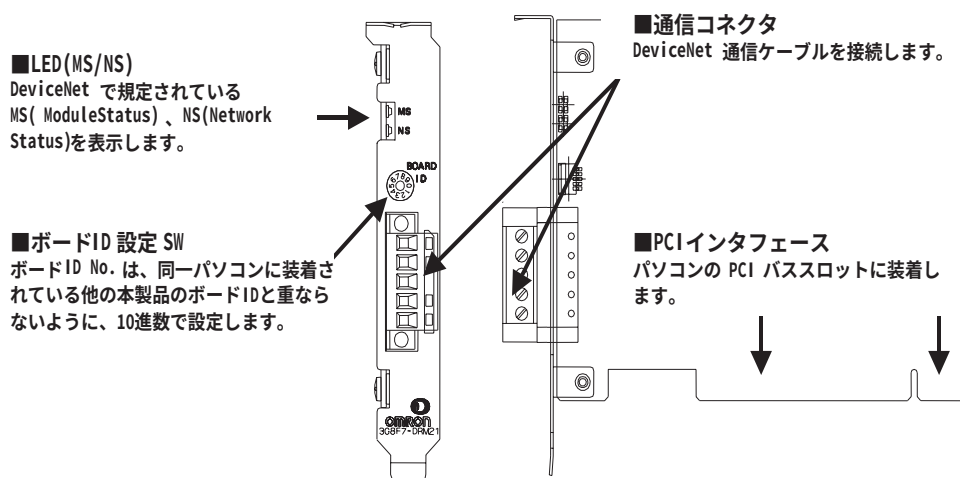
使用可能です。詳細については、「他の開発環境での注意事項」(P.3-1)を参照してください。

■外形寸法



(部品面の高さは、PCI バススロット 1 個分以内です。)

1-7 各部の名称と機能



■LED 表示の意味

LED 表示の意味は、次のようになっています。

MS	NS	意味
●	●	ブートプログラムの初期処理を実行中。
○ 緑	●	スキャナファームウェアの初期処理を実行中。
○ 緑	●	オンライン要求待ち。
○ 緑	○ 緑	コネクション開設後、I/O 通信中。 マスタからのコネクション待ち。
○ 緑	○ 緑	I/O 通信、あるいはメッセージ通信中。

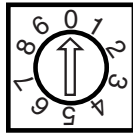
○
緑 : 点灯 ○
緑 : 点滅 ● : 消灯

■ボード ID

ボード ID は、複数の本ボードが装着されているときに、ボードの識別に使用される番号です。API 関数を使用する場合も、ボード ID を指定することで、対象のボードを指定します。

下図のように、10 進数で設定します。

有効な設定範囲は、0～7 です。



工場出荷時には、0 に設定されています。

本ボードは、1 台のパソコンに最大 3 枚まで装着できます。

参考

- 同じパソコンに装着されている本ボード間で番号が重複しない限り、0～7 の範囲内で自由に設定できます。
- 本ボード上では、0～9 の範囲で設定できますが、8～9 を設定すると、正常に使用できません。

1-8 使用までの手順

設定 (2-2 節参照)

項目	設定箇所	意味
ボード ID	ロータリスイッチ	本ボードが複数枚装着されているときの識別用の番号を設定します。

ボード ID の設定は、パソコンの電源投入前に行う必要があります。



パソコンへの装着 (2-2 節参照)

本ボードをパソコンに装着します。



ソフトウェアのインストール (2-3 節、2-4 節参照)

本ボードをパソコンから使用できるように、本ボード用のドライバや専用ソフトウェアをインストールします。



プログラム作成 (第 4 章～第 7 章参照)

本ボードを設定・制御するプログラム（ユーザアプリケーション）を作成します。

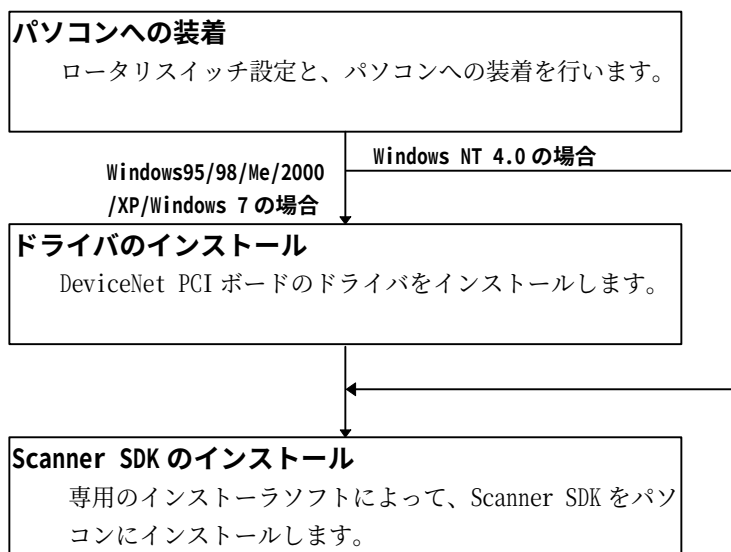
通信ケーブルの接続については、「2-5 DeviceNet への接続」を参照してください。
DeviceNet ネットワークの配線については、「DeviceNet ユーザーズマニュアル」(SCCC-308□)を参照してください。
スレーブについては、「DeviceNet スレーブマニュアル」(SB CD-324□、SB CD-305□)や、「DeviceNet マルチプル I/O ターミナルマニュアル」(SB CD-306□)を参照してください。

第2章 インストール

この章では、DeviceNet PCI ボードのパソコンへの装着方法や、ソフトウェアのインストール方法、通信ケーブルの接続方法について説明します。

2-1 インストールの手順

本ボードとソフトウェアのインストール手順を以下に示します。



参考

Windows 2000/XP/Windows 7 にドライバと Scanner SDK をインストールするには、Administrator 権限が必要です。

2-2 パソコンへの装着

ここでは、DeviceNet PCI ボードの設定方法と、パソコンへの装着方法について説明します。

■作業前の確認

DeviceNet PCI ボードは、Plug & Play に対応しています。
装着する前に、あらかじめ以下のことを確認しておいてください。

確認項目	確認内容
未使用拡張スロット	ご使用のパソコンの拡張スロット（PCI スロット）に空きがあることを確認してください。
IRQ の重複確認	本ボードは IRQ を使用します。 PCI バスでは IRQ が自動的に割り当てられますが、ISA バスを搭載しているパソコンでは、ISA バスで使用する IRQ と衝突し、パソコンが立ち上がらなくなる場合があります。 このような場合は、次のいずれかの方法をお試しください。 A) BIOS 設定メニューを呼び出し、Plug & Play OS を使用しない設定とする。 B) BIOS 設定メニューを呼び出し、PCI バスにおける IRQ の割り当て設定で、ISA バスで使用している IRQ を Reserved（自動割当禁止）とする。 注意）・BIOS 設定メニューの呼び出し方は、ご使用中のパソコンのマニュアルをご覧ください。 ・ISA バスで使用している IRQ は次の方法で確認ください。 （1）本ボードを装着しない状態でパソコンを立ち上げます。 （2）「コントロールパネル」→「システム」→「デバイスマネージャ」を起動します。 （3）該当するデバイスのプロパティを開き、リソースタブを押して使用している IRQ を確認します。

■ボード ID の設定

パソコンに装着する前に、DeviceNet PCI ボードのロータリスイッチで、ボード ID を設定します。

お願い

- DeviceNet PCI ボードのコネクタ部や部品面に直接触れないようにしてください。
- DeviceNet PCI ボードを取り扱うときには、静電気に注意してください。

ロータリスイッチで、0～7 のボード ID を設定します。1 台のパソコンに複数枚の DeviceNet PCI ボードを装着する場合は、ボード ID が重複しないように設定してください。工場出荷時設定は 0 です。

■パソコンへの装着

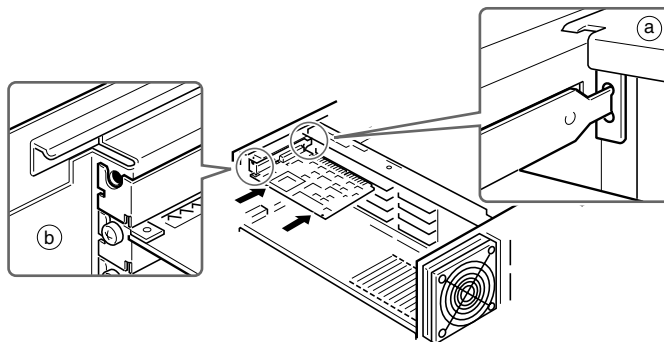
DeviceNet PCI ボードをパソコンの拡張スロット（PCI スロット）に装着します。

以下に装着手順を示しますが、実際のパソコンへの装着方法については、ご使用のパソコンの取り扱い説明書を参照してください。

お願い

- 装着作業を行う場合は、パソコンの電源を OFF にしてください。
- 静電気に注意してください。

- ①本ボードからすべてのケーブルを取り外してください。
- ②装着される側のパソコン本体の電源も OFF にして、電源コードを外してください。
- ③パソコン本体の取り扱い説明書を参照して、筐体を外し、PCI ボードの脱着を行える状態にしてください。
- ④パソコン側と本ボード側の、PCI バスコネクタの位置と向きを合わせて、奥に突き当たるまで押し込んでください。このとき、パソコン側のコネクタに、本ボード側のコネクタが均等に差し込まれるようにしてください。また、あまり無理に力を加えないようにご注意ください。



- ⑤本ボードを軽く引き、ボードが抜けないことを確認してください。
- ⑥前ページの図の b の箇所を、適正締め付けトルク $0.5\text{N} \cdot \text{m}$ でネジ止めし、ボードを固定してください。
- ⑦パソコン本体に筐体を取り付け、電源を ON にしてください。
- ⑧ご使用の OS（Windows の種類）によって、インストール方法が異なります。以下を参照して作業を行ってください。
- Windows 95/98/Me/2000/XP/Windows 7 の場合
パソコンが起動すると PCI ボードが新しいハードウェアとして認識され、自動的にウィザードが起動します。
以降の作業は、「2-3 ドライバのインストール」を参照してください。
 - Windows NT 4.0 の場合
パソコンを起動しても PCI ボードは認識されません。ドライバは Scanner SDK といっしょにインストールします。
以降の作業は、「2-4 Scanner SDK のインストール」を参照してください。

2-3 ドライバのインストール

PCI ボードをパソコンに装着後、パソコンを起動すると、本ボードが新しいハードウェアとして認識されます。本ボードを正常に動作させるために、専用のドライバをインストールしてください。

ドライバのインストール方法は、ご使用の OS（Windows の種類）によって、操作や画面表示が少し異なります。

ここでは Windows 98、Windows XP と Windows 7 の場合を例に説明します。

参考

Windows NT 4.0 では、PCI ボードは自動的に認識されません。ドライバは Scanner SDK といっしょにインストールします。「2-4 Scanner SDK のインストール」の手順を実行してください。

■Windows 98 の場合

- ①本ボードをパソコンに装着後、パソコンを起動すると、本ボードが新しいハードウェアとして認識され、自動的に「新しいハードウェアの追加ウィザード」が起動します（下図）。  をクリックします。



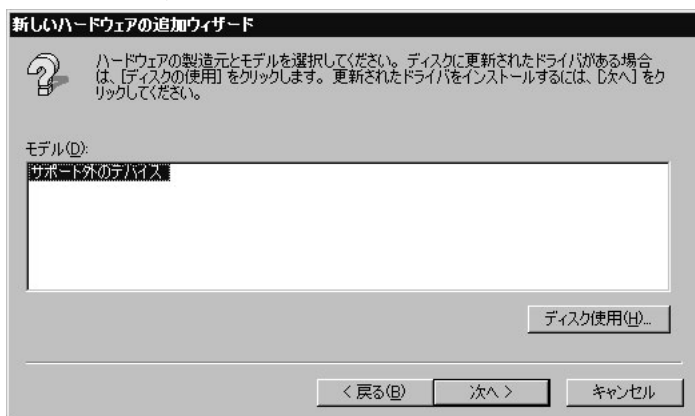
- ②ドライバの検索方法を指定するウィンドウが表示されたら、「特定の場所にあるすべてのドライバの一覧を作成し、インストールするドライバを選択する」を選択してから（下図）、  をクリックします。



- ③デバイスの種類として、「その他のデバイス」を選択してから（下図）、をクリックします。



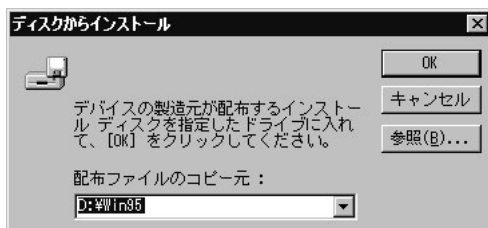
- ④ハードウェアのモデルを指定するウィンドウ（下図）では、「ディスク使用」ボタンをクリックします。



- ⑤ Scanner SDK の「CD-ROM ディスク」を CD-ROM ドライブにセットします。
- ⑥「配布ファイルのコピー元」を指定するウィンドウ（下図）では、CD-ROM ドライブの「Win95」フォルダを指定し、をクリックします。

[参照] ボタンをクリックすると、フォルダ構成を確認しながらフォルダを選択できるウィンドウが表示されます。

（※）下図は、CD-ROM ドライブがドライブ D の場合の例です。



- ⑦ CD-ROM からドライバの情報が読み出されたら、下図のようにモデルとして「OMRON 3G8F7-DRM21 PCI Adapter」を選択してから、**次へ(N) >** をクリックします。



- ⑧ 表示されている内容を確認してから、**次へ(N) >** をクリックします。
ドライバのインストールが開始されます。



- ⑨ インストール処理が終了すると、完了メッセージが表示されます(下図)。「完了」ボタンをクリックすると、ドライバのインストールが完了します。

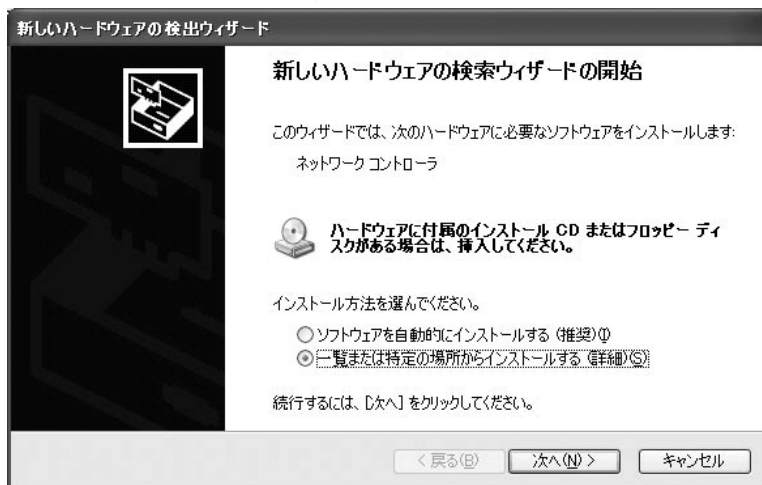


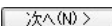
■Windows XP の場合

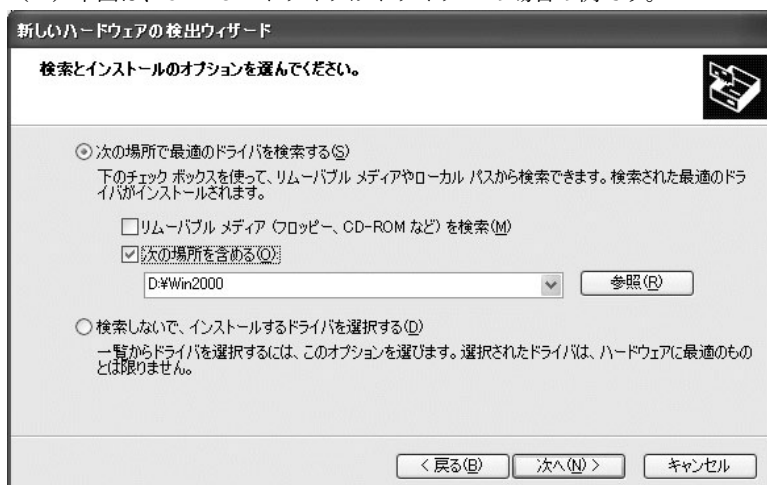
参考

Windows XP にドライバをインストールするには、Administrator 権限が必要です。

- ①本ボードをパソコンに装着後、パソコンを起動すると、本ボードが新しいハードウェアとして認識され、自動的に「新しいハードウェアの検出ウィザード」が起動します。「一覧または特定の場所からインストールする（詳細）」を選択してから（下図）、 をクリックします。



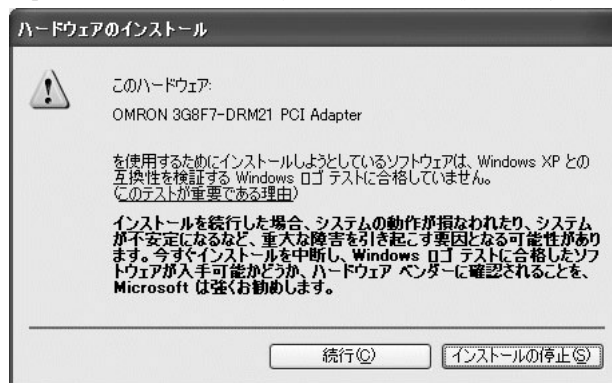
- ② Scanner SDK の「CD-ROM ディスク」を CD-ROM ドライブにセットします。
- ③検索とインストールのオプションとして、「次の場所で最適のドライバを検索する」を選択し、「次の場所を含める」にチェックを付けます。さらに、[参照] ボタンをクリックして、CD-ROM ドライブの Win2000 フォルダを指定してから（下図）、 をクリックします。
- （※）下図は、CD-ROM ドライブがドライブ D の場合の例です。



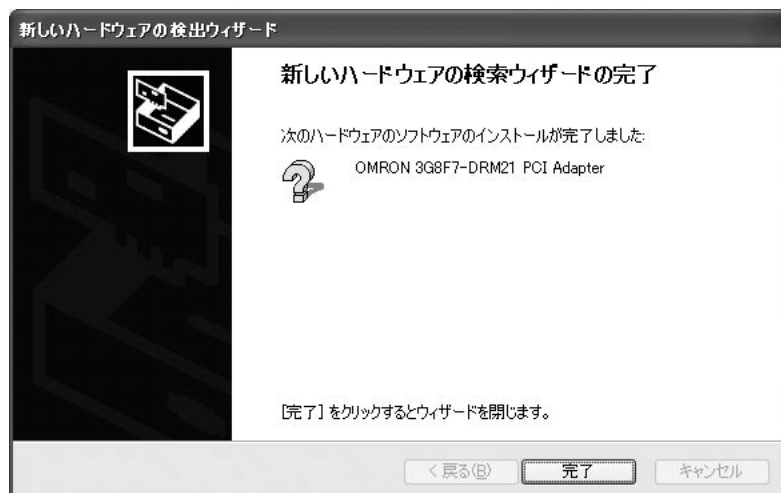
- ④ ドライバのインストールが開始されます。



- ⑤ インストールの途中で下図のウィンドウが表示されますが、問題ありませんので「続行」ボタンをクリックして、インストールを続けます。



- ⑥ インストール処理が終了すると、完了メッセージが表示されます（下図）。「完了」ボタンをクリックすると、ドライバのインストールが完了します。



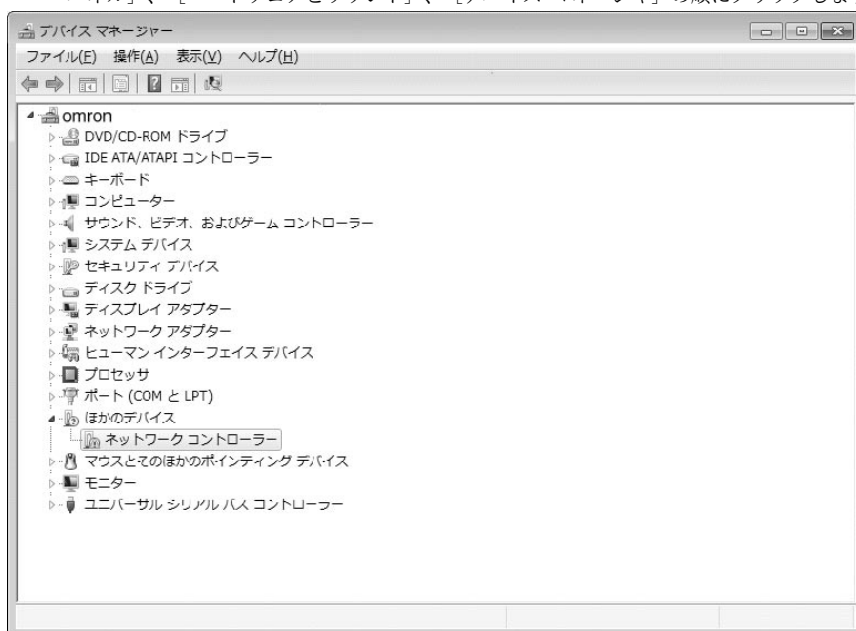
■Windows 7 の場合

参考

Windows 7 にドライバをインストールするには、Administrator 権限が必要です。

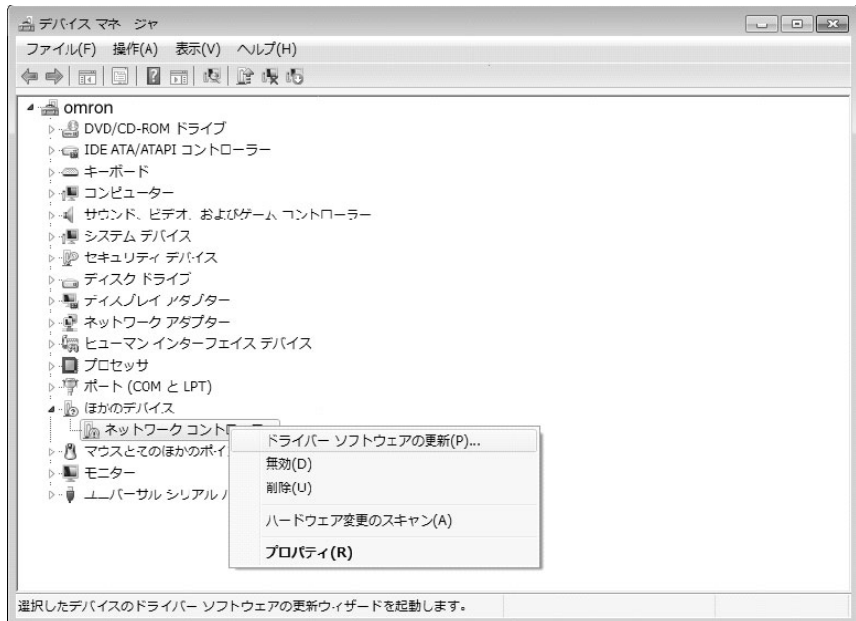
- ① ボードをパソコンに装着後、「デバイスマネージャ」を起動すると、新しいハードウェアを自動検出します。「デバイスマネージャ」※¹を開き、「ほかのデバイス」をダブルクリックしてください。

※1：[デバイスマネージャ]を開くには、[スタート] ボタン をクリックし、[コントロールパネル]、[ハードウェアとサウンド]、[デバイス マネージャ] の順にクリックします。



- ② 「ほかのデバイス」に「ネットワークコントローラー」が現れます。

「ネットワークコントローラー」を右クリックして、表示されるメニューの「ドライバソフトウェアの更新(P)」をクリックしてください。

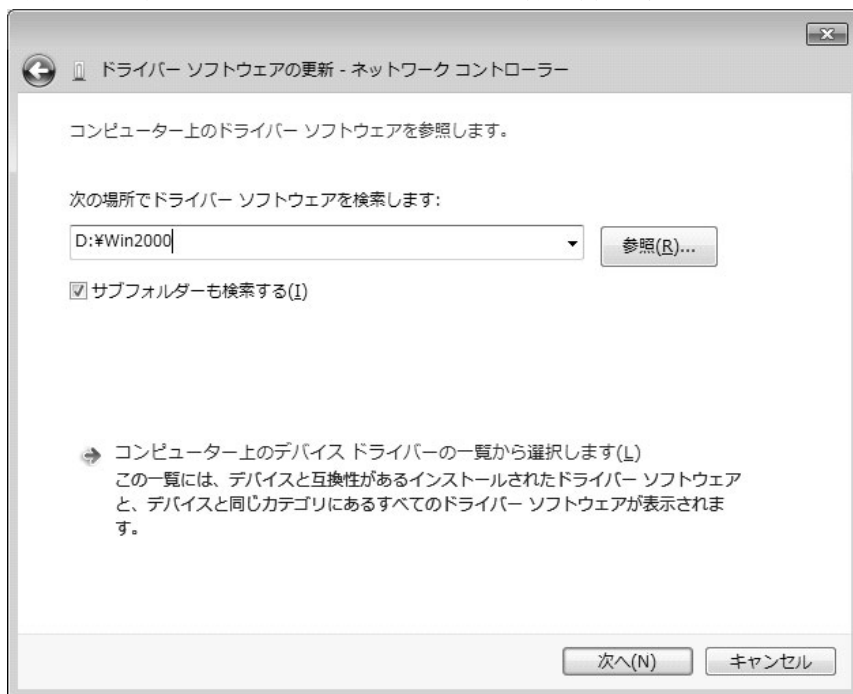


- ③ 「どのような方法でドライバーソフトウェアを検索しますか」が表示されます。

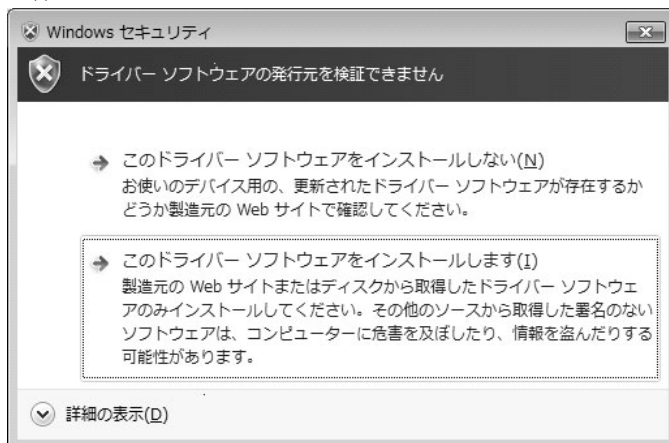
「コンピュータを参照してドライバーソフトウェアを検索します(R)」をクリックしてください。



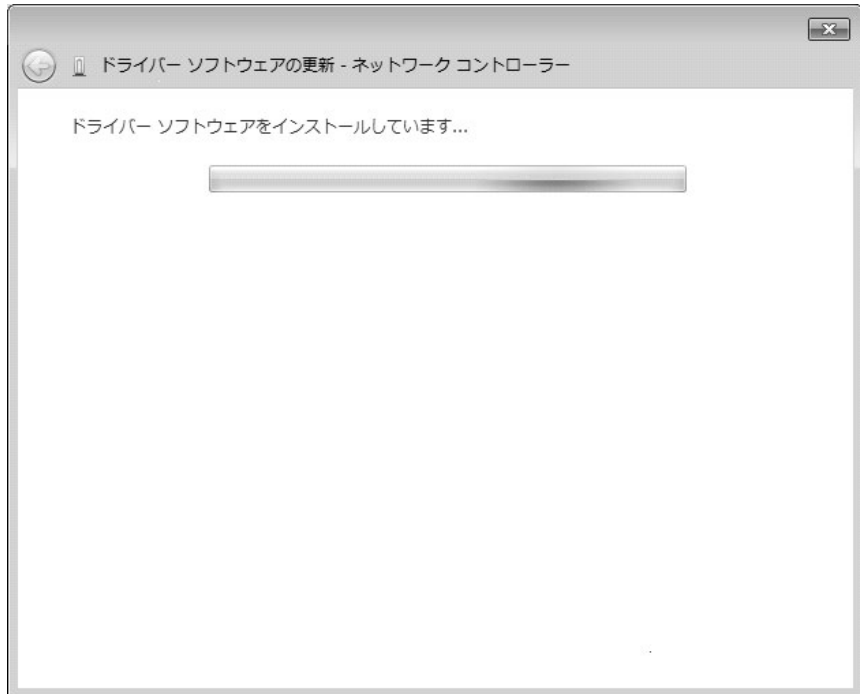
- ④ Scanner SDK の「CD-ROM ディスク」を CD-ROM ドライブにセットします。
- ⑤ 「コンピュータ上のドライバーソフトウェアを参照します」が表示されます。
[参照] ボタンをクリックして、CD-ROM ドライブの Win2000 フォルダを指定してから（下図）、[次へ(N)] ボタンをクリックします。
（※）下図は、CD-ROM ドライブがドライブ D の場合の例です。



- ⑥ 下図のウィンドウが表示されます。[このドライバーソフトウェアをインストールします(I)] ボタンをクリックしてインストールを開始します。



- ⑦ドライバのインストールが開始されます。



- ⑧インストール処理が終了すると、完了メッセージが表示されます（下図）。
[閉じる(C)] ボタンをクリックすると、ドライバのインストールが完了します。



2-4 Scanner SDK のインストール

ここでは、Scanner SDK のインストール方法について説明します。

参考

ご使用の Windows によって、操作や画面表示が少し異なります。ここでは、Windows 7 の場合を例に説明します。

お願い

Windows 7 に各ソフトウェア製品をインストールするには、Administrator 権限でログオンしなければなりません。

■インストール方法

- ① 実行中のアプリケーションをすべて終了させます。
- ② Scanner SDK の「CD-ROM ディスク」を CD-ROM ドライブにセットします。
- ③ Windows のスタートボタンをクリックし、「プログラムとファイルの検索」を選択します。



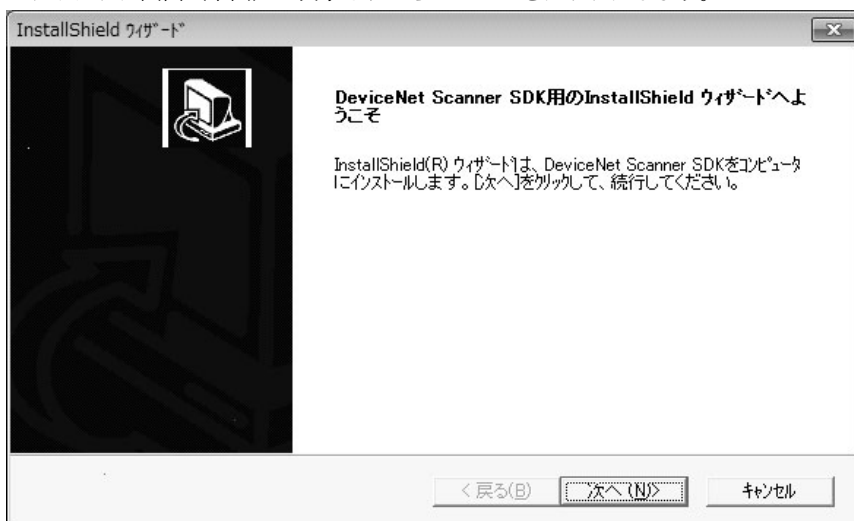
- ④表示された「プログラムとファイルの検索」の入力欄に「d:¥setup」と入力します。
ドライブ指定は、CD-ROM ディスクを挿入したドライブを指定してください。たとえば、ドライブ F に挿入した場合は、「f:¥setup」と入力します。



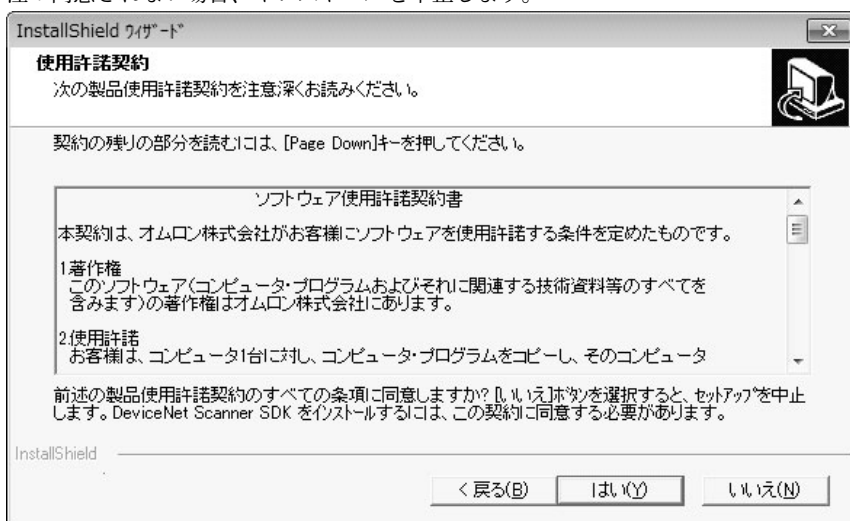
- ⑤検索結果表示一覧から「setup」をクリックしてください。



- ⑥ユーザーアカウント制御のウィンドウが表示されますが、問題ありませんので「はい(Y)」ボタンをクリックして、インストールを続けます。
- ⑦各ソフトウェアのインストールが開始されます。
インストーラがインストール準備を開始します。
- ⑧セットアップ画面（下図）が表示されたら「次へ(N)>」をクリックします。



- ⑨使用許諾契約が表示されます。
記載内容のすべての条項を確認のうえ、同意されるのであれば、「はい(Y)」をクリックします。
同意されない場合は、「いいえ(N)」をクリックします。
注：同意されない場合、インストールを中止します。



- ⑩ インストール先のフォルダを選択します。

初期状態では、次のフォルダがインストール先となっています。

C:\Program Files\OMRON\DeviceNet Scanner SDK

フォルダを変更する場合は、**参照(R)...**をクリックして、ドライブ・フォルダを指定し直します（存在しないフォルダを指定した場合は、自動的に作成されます）。

指定が完了したら、**次へ(N) >**をクリックします。

スタートメニューの登録先（プログラムフォルダ）を指定する画面が表示されます。



- ⑪ Windows のスタートメニュー内で、DeviceNet Scanner SDK のショートカットを作成するプログラムフォルダを指定します。

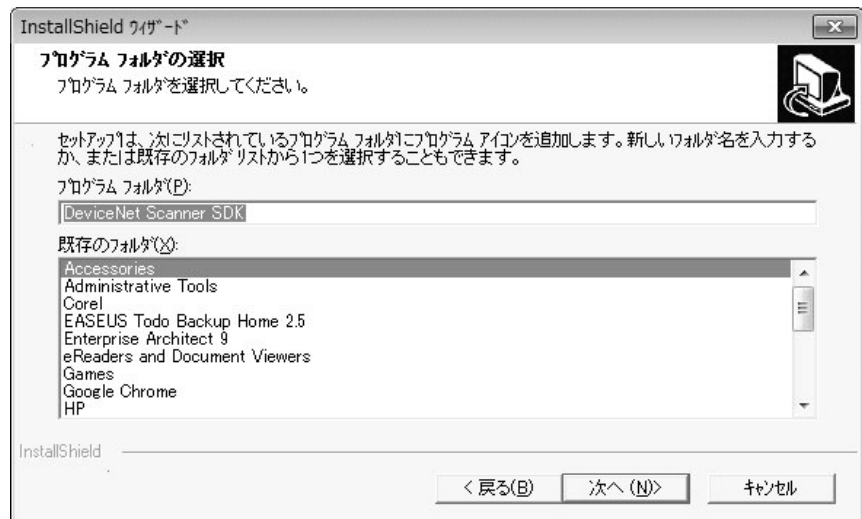
初期状態では、「DeviceNet Scanner SDK」というプログラムフォルダにショートカットが作成されるようになっています。

変更する場合は、「既存のフォルダ」から選択するか、直接フォルダ名を入力します。（存在しないフォルダを指定した場合は、自動的に作成されます）。

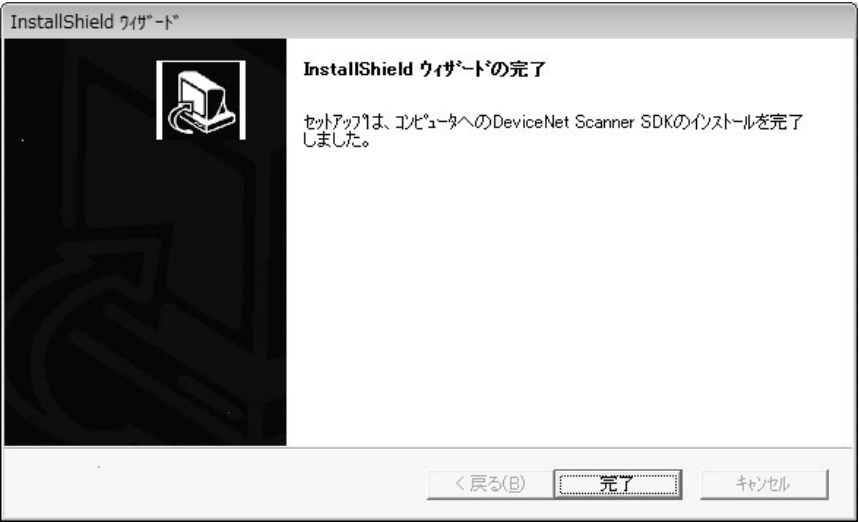
指定が完了したら、**次へ(N) >**をクリックします。

インストールが実行され、ファイルのコピーが行われます。

インストール中は、進行状況がパーセント数値で表示されます。



- ⑫インストール処理が終了すると、完了のメッセージが表示されます（下図）。
「完了」をクリックすると、インストールが完了します。



■インストールファイル

Scanner SDK をインストールすると、次のフォルダが作成され、必要なファイルがインストールされます。

ここではデフォルトのフォルダにインストールしたときの例で示します。

デフォルトのフォルダ	内容
¥Program Files¥OMRON¥DeviceNet Scanner SDK¥Manual¥	本マニュアルの PDF ファイルがインストールされます。
¥Program Files¥OMRON¥DeviceNet Scanner SDK¥Program¥	サンプルプログラムの実行ファイルがインストールされます。
¥Program Files¥OMRON¥DeviceNet Scanner SDK¥SDK¥Include¥	インクルードファイルのサンプルがインストールされます。
¥Program Files¥OMRON¥DeviceNet Scanner SDK¥SDK¥Lib¥	Microsoft Visual C++でスタティックリンクのときに使用する Lib ファイルがインストールされます。
¥Program Files¥OMRON¥DeviceNet Scanner SDK¥SDK¥Sample¥	サンプルプログラムがインストールされます。

■アンインストールについて

DeviceNet Scanner SDK が不要になった場合は、以下の手順により、DeviceNet Scanner SDK のファイルや、インストール情報を削除してください（インストールしたファイルのみが削除されます。後から作成したファイルなどは削除されません）。

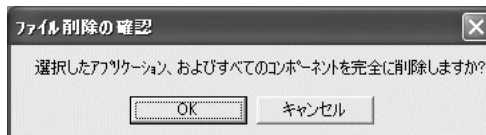
① Windows のスタートボタンをクリックし、「コントロールパネル」を選択します。

② 「コントロールパネル」のウィンドウ内の「プログラムの追加と削除」をダブルクリックします。

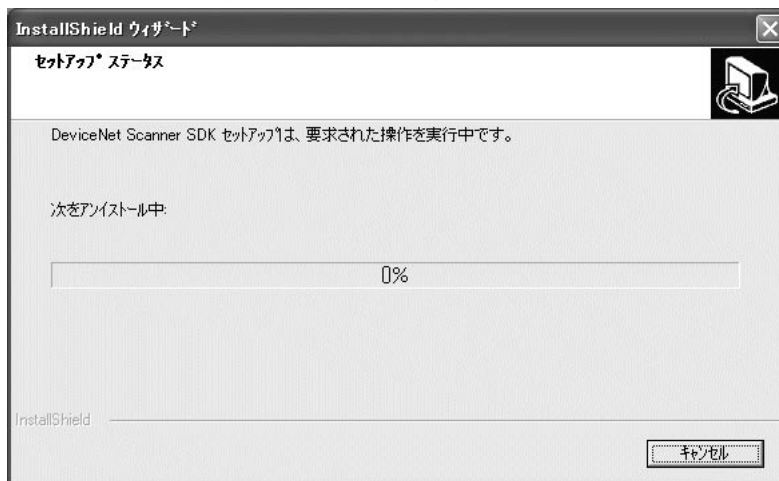
③ 一覧から「DeviceNet Scanner SDK」を選択し（下図）、**変更と削除** をクリックします。



④ 削除の確認メッセージが表示されたら、**OK** をクリックします。

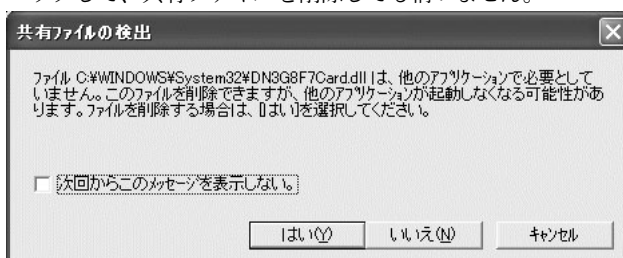


⑤ アンインストールが開始されます（下図）。



⑥ アンインストール処理中に、「共有ファイルの検出」メッセージが表示され、削除の確認を求められることがあります（下図）。

- ・他の DeviceNet アプリケーション（コンフィグレータや、NetXServer、アナライザなど）をインストールしているときは、[いいえ] をクリックして、共有ファイルを削除しないようにしてください。
- ・他に DeviceNet アプリケーションをインストールしていないときは、[はい] をクリックして、共有ファイルを削除しても構いません。



⑦ アンインストール処理が終了すると、完了メッセージが表示されます（下図）。

 をクリックすると、アンインストールが完了します。



2-5 DeviceNet への接続

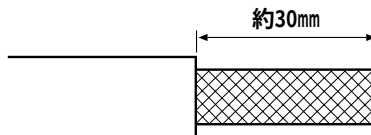
本ボードを取り付けたら、DeviceNet の通信ケーブルを接続します。

このマニュアルでは、DeviceNet の配線のうち、本ボードに関する部分のみ説明します。通信ケーブルの配線や、スレーブ機器の接続などについては、「DeviceNet ユーザーズマニュアル」（SCCC-308□）を参照してください。

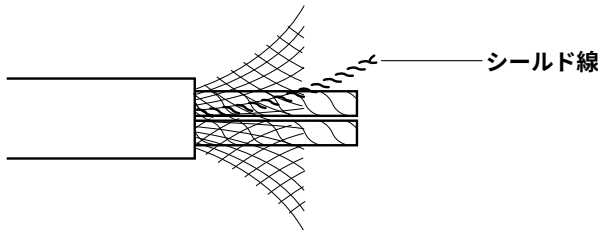
■通信ケーブルの加工とコネクタへの取り付け

ここでは、ネットワークの通信ケーブルへのコネクタの取り付け方法について説明します。以下の手順に従って、通信ケーブルを加工し、コネクタを取り付けてください。

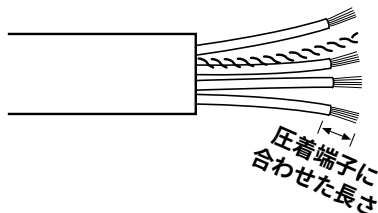
- ①シールドの網をあまり傷つけないように注意しながら、ケーブルの被覆を約 30mm むいてください。ただし、あまり余分にむきすぎると、短絡の原因となったり、ノイズの影響を受けやすくなりますのでご注意ください。



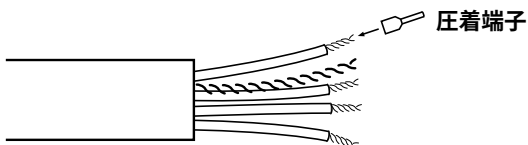
- ②シールドの網を丁寧にほぐします。信号線と電源線の他に、より合わせた、むき出しのシールド線が 1 本あります（網の部分よりも少し硬いので触れればわかります）。



- ③余分なシールドの網を切り取り、信号線と電源線を包むアルミテープをはがしてから、信号線と電源線の被覆を圧着端子に合わせてむきます。むき出した信号線と電源線は、それぞれしっかり、より合わせてください。



- ④ケーブルの被覆をむいた部分に圧着端子を装着し、専用の工具を使って、しっかり圧着してください。



参 考

ケーブルの圧着端子として、以下の製品を推奨します。

・フェニックス・コンタクト社製 AI シリーズ

コネクタの種類		形 XW4B-05C1-H1-D	形 XW4B-05C4-TF-D	形 XW4G-05C1-H1-D	専用工具
ケーブルの種類		形 XW4B-05C1-V1R-D	形 XW4B-05C4-T-D	形 XW4G-05C4-TF-D	
細いケーブル用	信号線	AI 0.25-6YE	AI 0.25-8YE	AI 0.25-8YE	CRIMPFOX ZA3
	電源線	AI 0.5-6WH	AI 0.5-10WH	AI 0.5-10WH	
太いケーブル	信号線	AI-6	AI-10	AI-10	
	電源線	AI 2.5-8BU	AI 2.5-10BU	AI 2.5-10BU	

フェニックス・コンタクト株式会社

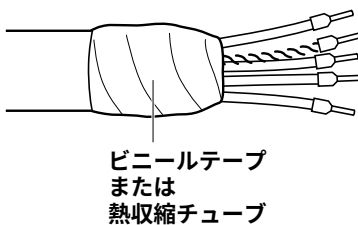
〒222-0033 神奈川県横浜市港北区新横浜 1-7-9

友泉新横浜 1 丁目ビル 6F

Tel 045-471-0030

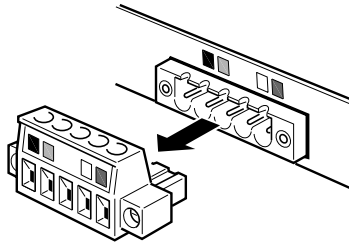
Fax 045-471-0031

- ⑤ケーブルをビニールテープや収縮チューブで処理します。



■通信ケーブルの接続

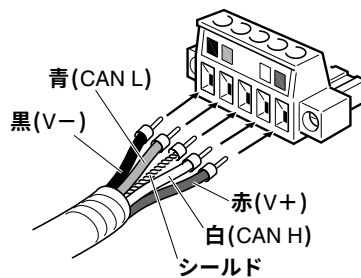
- ①本ボードの DeviceNet 通信コネクタからコネクタを取り外してください。



参考

コネクタを取り付けたままで通信ケーブルを接続できる場合は、取り外さなくても作業を行えます。

- ②コネクタの向きに気を付けながら、信号線、電源線、シールド線をコネクタの各穴に差し込みます。次の図のように、左から黒、青、シールド、白、赤の順で差し込みます。



お願い

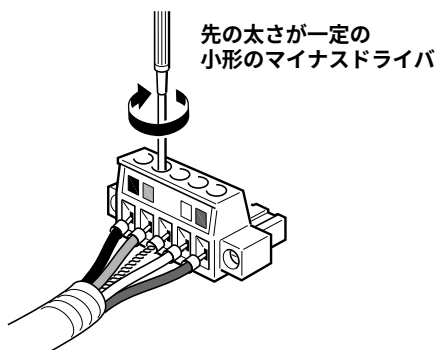
信号線や電源線、シールド線を差し込む前に、コネクタのケーブル固定用のネジを十分ゆるめておいてください。ネジが締められたままだと、締め付ける側ではなく、裏の隙間に信号線が入り、固定できないことがあります。

参考

- コネクタと本ボードには、ケーブルの色に対応したシールが貼られています。ケーブルの色を、シールの色と合わせることで、配線が合っているか確認することができます。
- ケーブルの色は、次のようになっています。

色	信号種別	記号
黒	通信電源－側	V－
青	信号 Low 側	CAN L
－	シールド	S
白	信号 High 側	CAN H
赤	通信電源＋側	V＋

- ③コネクタのケーブル固定用のネジで、各線ごとにしっかり締め付けてください。
先だけが細くなっている通常のドライバでは、途中でつかえて奥まで入りません。太さが一定の小型のマイナスドライバを使用してください。
ケーブル固定用のネジは、適正締め付けトルク $0.25 \sim 0.3 \text{ N} \cdot \text{m}$ で締め付けてください。
太いケーブルを使用する場合は、ケーブルの張力によりコネクタが抜けることがないように、ケーブルに余裕を持たせて配線してください。

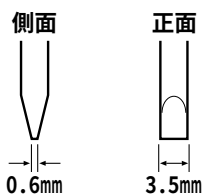


参考

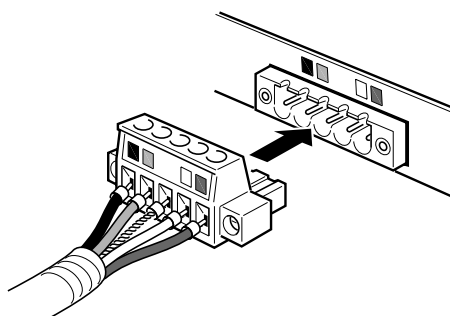
DeviceNet コネクタのネジの締め付けには、専用ドライバ 形 XW4Z-00C が適しています。

オムロン製 形 XW4Z-00C

先端の形状：

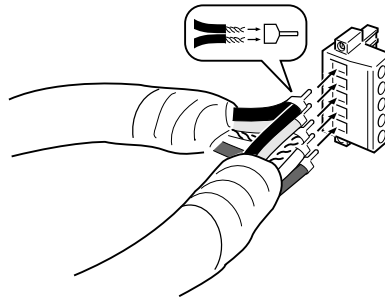


- ④コネクタとボード側コネクタの向きを合わせてから、しっかり奥に入るまでコネクタを差し込み、コネクタの両端にあるネジをしっかり締め付けてください。
コネクタ取り付け用のネジは、適正締め付けトルク $0.25 \sim 0.3 \text{ N} \cdot \text{m}$ で締め付けてください。



●標準角型コネクタを使用する場合（細いケーブルのみ）

細いケーブルをマルチドロップ方式で接続する場合は、2本の線を1つの圧着端子で圧着してから、コネクタに差し込んでください。



参考

2本の線を1つの圧着端子で圧着する場合は、圧着端子として次の製品を推奨します。

コネクタの種類	圧着端子	専用工具
形 XW4B-05C1-H1-D MSTB2.5/5-ST-5.08 形 XW4B-05C1-V1R-D	AI-TWIN2×0.5-8WH フエニックス・コンタクト社製	CRIMPFOX UD6 または CRIMPFOX ZA3
形 XW4G-05C1-H1-D	H0.5/16.5 ZH ワイドミュラー社製	

フエニックス・コンタクト株式会社

〒222-0033 神奈川県横浜市港北区新横浜 1-7-9
友泉新横浜 1 丁目ビル 6F

TEL 045-471-0030

FAX 045-471-0031

日本ワイドミュラー株式会社

〒183-0055 東京都府中市府中町 1-14-1 朝日生命ビル 11F

TEL 042-330-7891

●マルチドロップ用コネクタ（別売）を使用する場合

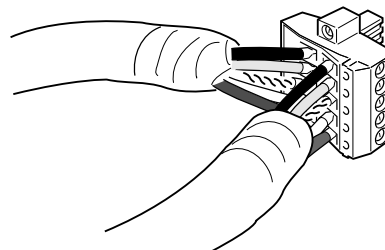
（細いケーブルの場合、太いケーブルの場合）

太いケーブルを使用してマルチドロップ方式で配線する場合は、以下のマルチドロップ用コネクタを使用してください。

形 XW4B-05C4-T-D（ネジなし平行型マルチドロップ用コネクタ）：オムロン製

形 XW4B-05C4-TF-D（ネジ付き平行型マルチドロップ用コネクタ）：オムロン製

形 XW4G-05C4-TF-D（ネジ付きクランプ式平行型マルチ分岐用コネクタ）：オムロン製
ただし、マルチドロップ用コネクタは、パソコンによっては、ケースとぶつかるため、使用できないことがあります。また、隣のスロットを使用できなくなることがあります。ご使用になるパソコンで、十分に確認してください。



お願い

- 通信ケーブルの接続は、必ずパソコン本体や、全スレーブの電源と、通信電源がすべて OFF の状態で行ってください。
- 配線には必ず圧着端子を使用してください。
より合わせただけの電線を接続すると、ケーブルの抜けや断線などにより、誤動作したり、製品を破損する恐れがあります。
- 圧着端子装着の際には、各圧着端子の適用圧着工具を使用して、正しい方法で装着してください。適用工具、圧着方法については、圧着端子のメーカまでお問い合わせください。適用工具の不使用や、誤った装着方法は、ケーブルの断線の原因になります。
- 信号線、電源線、シールド線の接続は、間違えないように気を付けてください。
- 各線の固定用ネジはしっかり締めてください。適正な締め付けトルクは、 $0.25 \sim 0.3 \text{ N} \cdot \text{m}$ です。
- 通信中に信号線、電源線、シールド線が抜けないようにご注意ください。
- 通信ケーブルを引っ張らないでください。コネクタの抜けや断線の恐れがあります。
- 通信ケーブルは、曲げ半径を十分にとり、無理に曲げないようにしてください。無理に曲げると、コネクタの抜けや断線の恐れがあります。
- 通信ケーブルに物を乗せないでください。断線の恐れがあります。
- 配線を十分に確認してから、通電してください。

第 3 章 API 関数の使用手順

ここでは、機能ごとに API 関数の使用フローの例や、使用上の注意点などについて説明します。DeviceNet PCI ボード（以下、「本ボード」）を使用するためのアプリケーションを実際に作成する際の参考としてください。

3-1 アプリケーション開発環境

■推奨する開発環境

- Microsoft Visual C++（Ver.6.0 以降）

■他の開発環境での注意事項

開発環境により API の使用の可否が異なります。
以下、開発環境の例を挙げますが、詳細は開発環境の提供元にご確認をお願いします。

- Microsoft Visual C++（Ver.6.0）の場合

API 関数を使用可能（Declare 宣言せずに API 関数を使用できます）。

- Microsoft Visual Basic 6.0 の場合

Declare 宣言することにより、Explicit メッセージ用以外の API 関数を使用できます。

- Microsoft Visual Basic 2005 の場合

使用不可（Declare 宣言しても API 関数は使用できません）。

3-2 API 関数の種類

本ボードでは、次のような API 関数が用意されています。

■ボード管理用 API

●ボードサービス API

ボードサービス API は、Scanner SDK やドライバのバージョン確認、ボードのオープン/クローズ、ネットワークへの加入/離脱などを行うための API です。本ボードを使用して通信を行うためには、ボードのオープンとネットワークへの加入が必要です。

●PC ボード割り込みサービス API

PC ボード割り込みサービス API は、本ボードからの割り込みを検知するための API です。検知できる割り込みは次の 2 つです。

割り込み	名称	機能
BD_WDT	ボードウォッチドッグタイマ タイムアウト割り込み	この割り込みが発生した場合は、本ボードが何らかの異常により動作を停止しています。この割り込みを検知して、必要な異常処理を行うことができます。
SCAN	1 スキャン（通信サイクル）完了 割り込み	この割り込みは、I/O 通信中に 1 通信サイクルが完了するごとに発生します。この割り込みが発生した後に、最新の入力データ（IN データ）を取得することができます。

割り込みを検知するためには、本ボードからパソコンに対して割り込みが発生するように、割り込みコントロールレジスタを設定する必要があります。また、割り込みの検知には 2 種類の方法があります。検知方法については、「3-3 イベントのチェック方法」（P.3-5）を参照してください。

■マスタ機能用 API

●スキャンリスト操作 API

マスタ機能を使用してスレーブと I/O 通信を行うためには、スレーブの情報をスキャンリストに登録する必要があります。スキャンリスト操作 API は、スキャンリストを作成するための API です。スキャンリストとスキャンリスト操作 API 関数の関係については、「パラメータ操作 API 関数について」（P.3-8）を参照してください。

●I/O 通信サービス API

I/O 通信サービス API は、マスタ機能における I/O 通信の開始/停止、通信サイクル時間の設定、実際のスレーブ情報の取得などを行うための API です。

●I/O データアクセスサービス API

I/O データアクセスサービス API は、I/O データの読み出し/書き込みを行うための API です。本ボードには、関数で読み出し/書き込みできる I/O エリアと、ボードの内部バッファの 2 つのエリアがありますが、I/O 通信では内部バッファのみ使用しています。ユーザアプリケーションで I/O データを使用する場合は、すべての OUT データを書き込み後、2 つのエリア間のデータ更新を行い、すべての IN データを取得するようにしてください。

■スレーブ機能用 API

●スレーブスキャンリスト操作 API

スレーブ機能を使用するためには、スレーブとして動作するための情報をスレーブスキャンリストに登録する必要があります。スレーブスキャンリスト操作 API は、スレーブスキャンリストに登録するための API です。スレーブスキャンリストとスレーブスキャンリスト操作 API 関数の関係については、「パラメータ操作 API 関数について」(P.3-8)を参照してください。

●I/O 通信サービス API

I/O 通信サービス API は、スレーブ機能において、マスタとの I/O 通信を一時的に停止/開始するための API です。

●I/O データアクセスサービス API

マスタ機能における I/O データアクセスサービス API と同様の関数が、スレーブ機能においても用意されています。

■Explicit メッセージ用 API

●メッセージ監視タイマサービス API

メッセージ監視タイマは、Explicit メッセージリクエストを送信後、相手からの Explicit メッセージレスポンスを受信するまでの時間を監視するタイマです。この時間内にレスポンスを受信できない場合、そのリクエストはタイムアウトとなります。メッセージ監視タイマサービス API は、相手デバイスごとにこのメッセージ監視タイマを設定するための API です。

メッセージ監視タイマのデフォルト値は 2 秒 (2000ms) です。相手ノードからのレスポンスが遅いときは、この値を長く設定してください。ただし、レスポンス待ちの間は、同じデバイスに対して次のリクエストを送信することはできません。

●クライアントメッセージサービス API

クライアントメッセージサービス API は、任意のデバイスのパラメータを設定したり、そのデバイスの持つさまざまな情報をモニタリングするために、Explicit リクエストメッセージを送信してレスポンスを受信するための API です。

Scanner SDK は、送信したリクエストに対するレスポンスを受信すると、自動的に DLL 内部の受信キューに格納します。受信キューは相手デバイスごとに存在し、パソコンに空きメモリがある限りキューイングされますが、レスポンスの受信を検知した後はできるだけ早く取り出してください。

なお、Explicit メッセージはノイズなどにより消失する可能性があります。また、相手デバイスによってはメッセージを受信しそこねる場合があります。Explicit リクエストを送信する場合は、必ずリトライ処理をするようにしてください。

Explicit レスポンスメッセージ受信の検知には 2 種類の方法があります。検知方法については、「3-3 イベントのチェック方法」(P.3-5)を参照してください。

●サーバメッセージサービス API

サーバメッセージサービス API は、任意のデバイスプロファイルを実装するために、Explicit リクエストメッセージを受信してレスポンスを送信するための API です。

Scanner SDK は、登録したオブジェクト宛のリクエストを受信すると、自動的に DLL 内部の受信キューに格納します。受信キューは登録したオブジェクトごとに存在し、パソコンに空きメモリがある限りキューイングされますが、リクエストの受信を検知した後はできるだけ早く取り出してください。

Explicit リクエストメッセージ受信の検知には 2 種類の方法があります。検知方法については、「3-3 イベントのチェック方法」(P.3-5)を参照してください。

■保守用 API

●ステータスサービス API

ステータスサービス API は、I/O 通信やメッセージ通信の中で発生する異常や、マスタ機能での通信サイクル時間を監視するための API です。ユーザアプリケーションを作成するときは、異常ステータスを監視し、必要によりエラー処理を行ってください。

●異常履歴アクセスサービス API

異常履歴アクセスサービス API は、ボード内に登録された異常履歴の読み出し/クリアを行うための API です。異常履歴機能の詳細については、「3-10 異常履歴機能の使い方」(P.3-16)を参照してください。

●PC ウォッチドッグタイマサービス API

PC ウォッチドッグタイマサービス API は、本ボードがユーザアプリケーションの正常動作を監視するための API です。PC ウォッチドッグタイマ管理機能については、「3-11 PC ウォッチドッグタイマ管理機能の使い方」(P.3-17)を参照してください。

3-3 イベントのチェック方法

■イベントの種類

Scanner SDK では、以下のイベントが発生します。

イベント種別	内容
本ボードからの割り込み発生	本ボードからパソコン側に割り込みがあったときに発生するイベント（割り込みコントロールレジスタの設定が必要）
Explicit クライアントレスポンス受信	Explicit クライアントリクエストメッセージを送信した後、サーバ側からのレスポンスを受信したときに発生するイベント
Explicit サーバリクエスト受信	オブジェクトを登録した後、そのオブジェクト宛の Explicit サーバリクエストメッセージを受信したときに発生するイベント

■チェック方法

本ボードでは、次の 2 種類の方法でイベントのチェックができます。用途や状況に応じて、使いやすい方を使用してください。

- ・ Windows メッセージによるイベント通知
- ・ ポーリングによるイベントチェック

Windows メッセージによるイベント通知

イベント発生時に Windows メッセージが自動的に発行されます。イベント通知先として、スレッドまたはウィンドウを指定することができます。割り込みステータスや受信データ長などのイベント固有の情報も、このとき同時に通知されます。

Windows メッセージによるイベント通知を行う場合は、あらかじめ `SCAN_Reg~Message()` 関数により、通知メッセージの設定を行っておく必要があります。

ユーザアプリケーション側では、イベント通知を受け取ったら、対応するイベントの処理を実行してください。たとえば、Explicit クライアントレスポンス受信イベントの場合は、受信したサービスデータサイズ分のデータバッファを用意した上で、レスポンスの読み出しを行います。

ポーリングによるイベントチェック

ユーザアプリケーションで `SCAN_Peek~()` 関数を使用することにより、イベントチェックを行います。

チェックの結果、イベントが発生している場合は、対応するイベントの処理を実行してください。たとえば、Explicit クライアントレスポンス受信イベントの場合は、`SCAN_ClientEventLength()` 関数により、受信データサイズを調べ、受信したサービスデータサイズ分のデータバッファを用意した上で、レスポンスの読み出しを行います。

3-4 エラーのチェック方法

■関数の戻り値によるエラーのチェック

Scanner SDK の API 関数は、すべて BOOL 型となっていて、関数の実行結果が、次のように戻り値として返されるようになっています。

FALSE : 関数が異常終了

TRUE : 関数が正常終了

この場合、詳細なエラー情報を、GetLastError()関数で調べることができます。

GetLastError()関数の結果得られたエラーコードと、その意味、処置については、「エラーコード」(P.7-4)を参照してください。

参考

GetLastError は WindowsAPI の関数です。以下のように DeviceNet PCI ボード API を発行した直後にコールすることで、詳細エラー情報を取得することができます。

＊ボードオープン API コールでの GetLastError()関数使用例

```
DWORD dwErrCode;
//DeviceNet PCI ボードオープンの API コール
if(SCAN_Open(DeviceNo, Handle)!=false){
    //関数がエラーを返してきた場合
    //詳細エラーコードを取得する
    dwErrCode=GetLastError( );
}
```

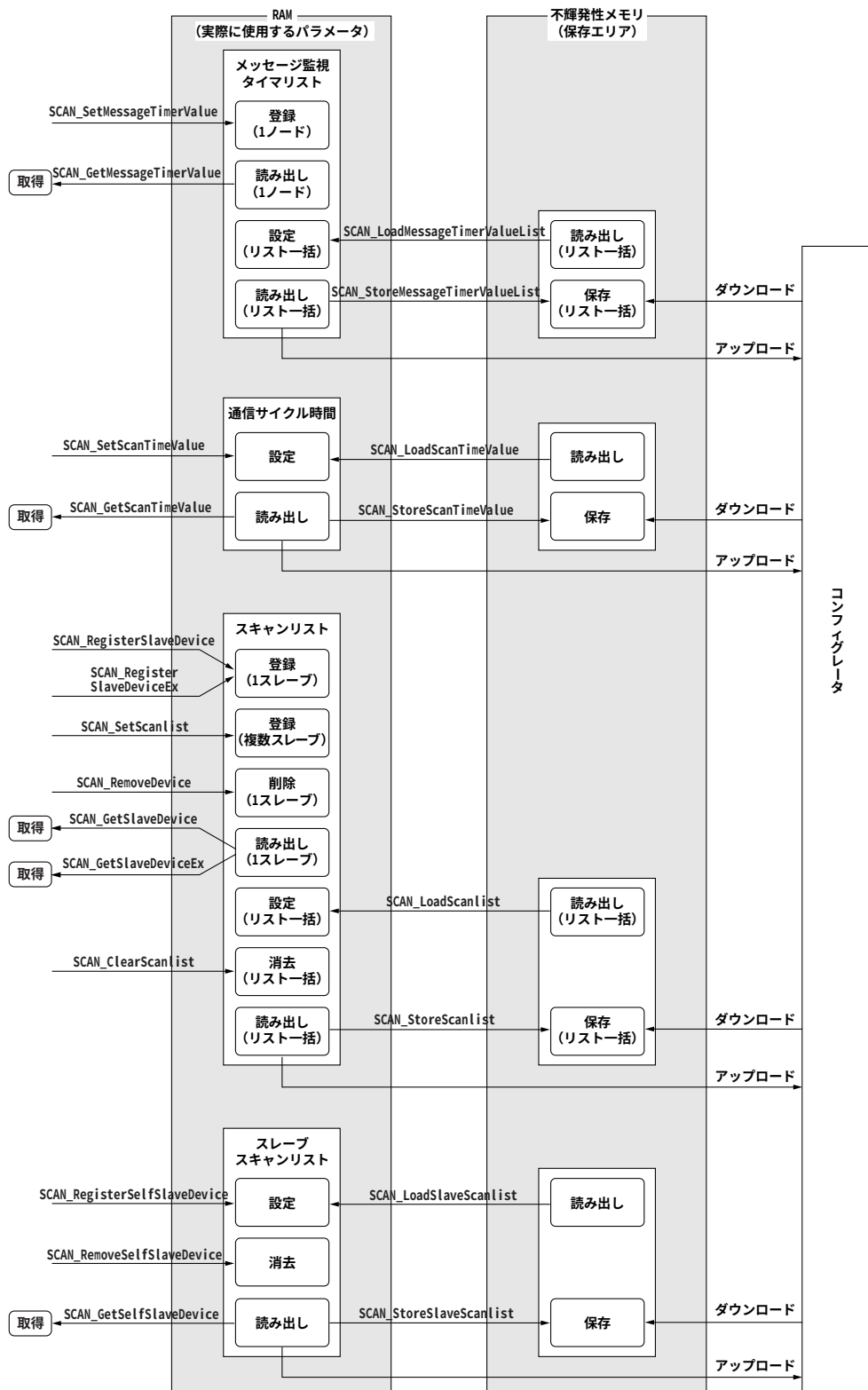
3-5 パラメータについて

■パラメータの種類

本ボードには、次のパラメータがあります。

名称	意味	起動後の初期値
メッセージ 監視タイマリスト	全ノードアドレス分のメッセージ監視タイマのリストです。Explicit リクエストメッセージを送信してから、レスポンスを受信するまでの監視時間です。	全ノードアドレスにデフォルト値の 0（2 秒）が設定されています。
通信サイクル時間	マスタ機能で使⽤します。 I/O 通信の周期です。	デフォルト値の 0（自動設定）が設定されています。
（マスタ） スキャンリスト	マスタ機能で使⽤します。 I/O 通信の対象とするスレーブのノードアドレスと、ノードごとの I/O 通信に必要なパラメータを登録したリストです。	デフォルトではスキャンリストは存在しません。 API 関数による登録が必要です。
スレーブ スキャンリスト	スレーブ機能で使⽤します。 スレーブとして動作するために必要なパラメータを登録したリストです。	デフォルトではスレーブスキャンリストは存在しません。 API 関数による登録が必要です。

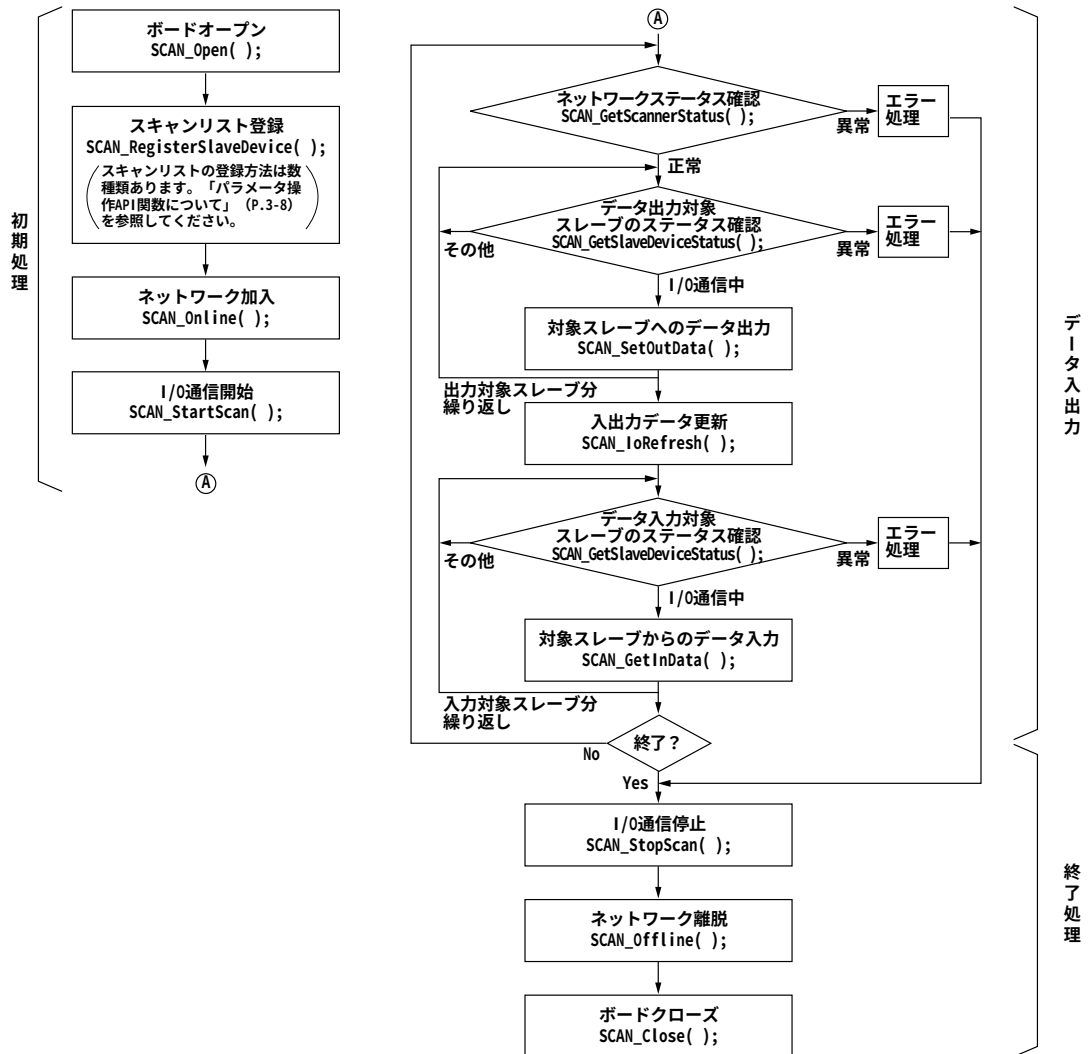
■パラメータ操作 API 関数について



3-6 I/O 通信機能の使用手順

■マスタ機能の使用手順

Scanner SDK でマスタ機能を使用する場合は、下図のような手順で API 関数を使用します。
使用方法の詳細は、「API 関数リファレンス」(P.4-1)を参照してください。

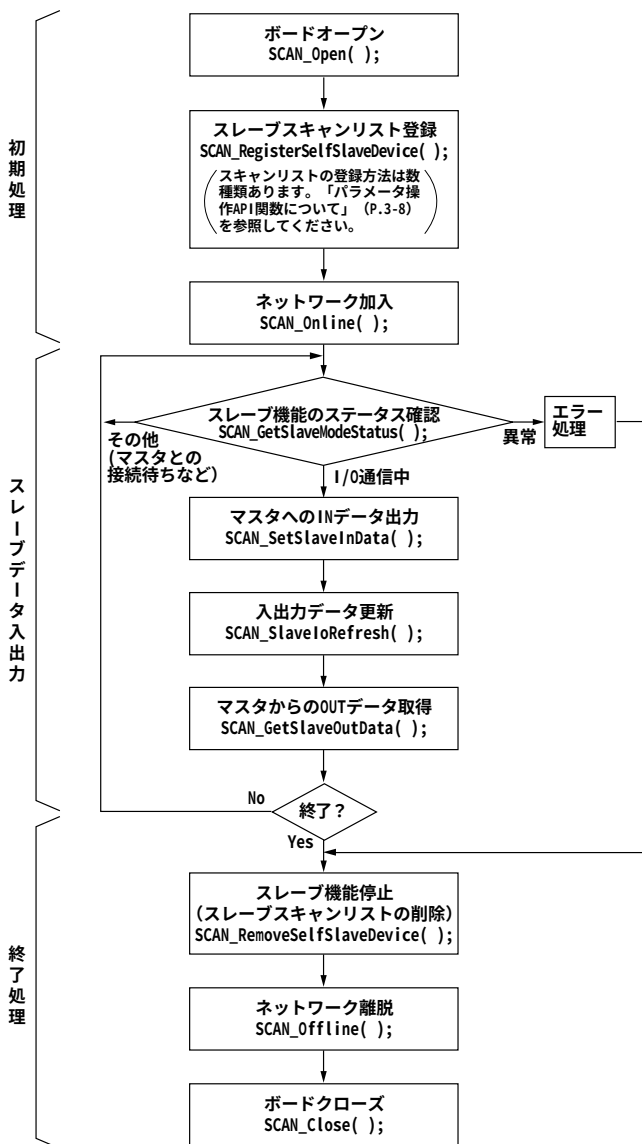


参考

SCAN_StartScan()を呼び出してから、実際に I/O 通信が始まるまで、
数秒の時間を必要とします。

■スレーブ機能の使用手順

Scanner SDK でスレーブ機能を使用する場合は、下図のような手順で API 関数を使用します。使用方法の詳細は、「API 関数リファレンス」(P.4-1)を参照してください。



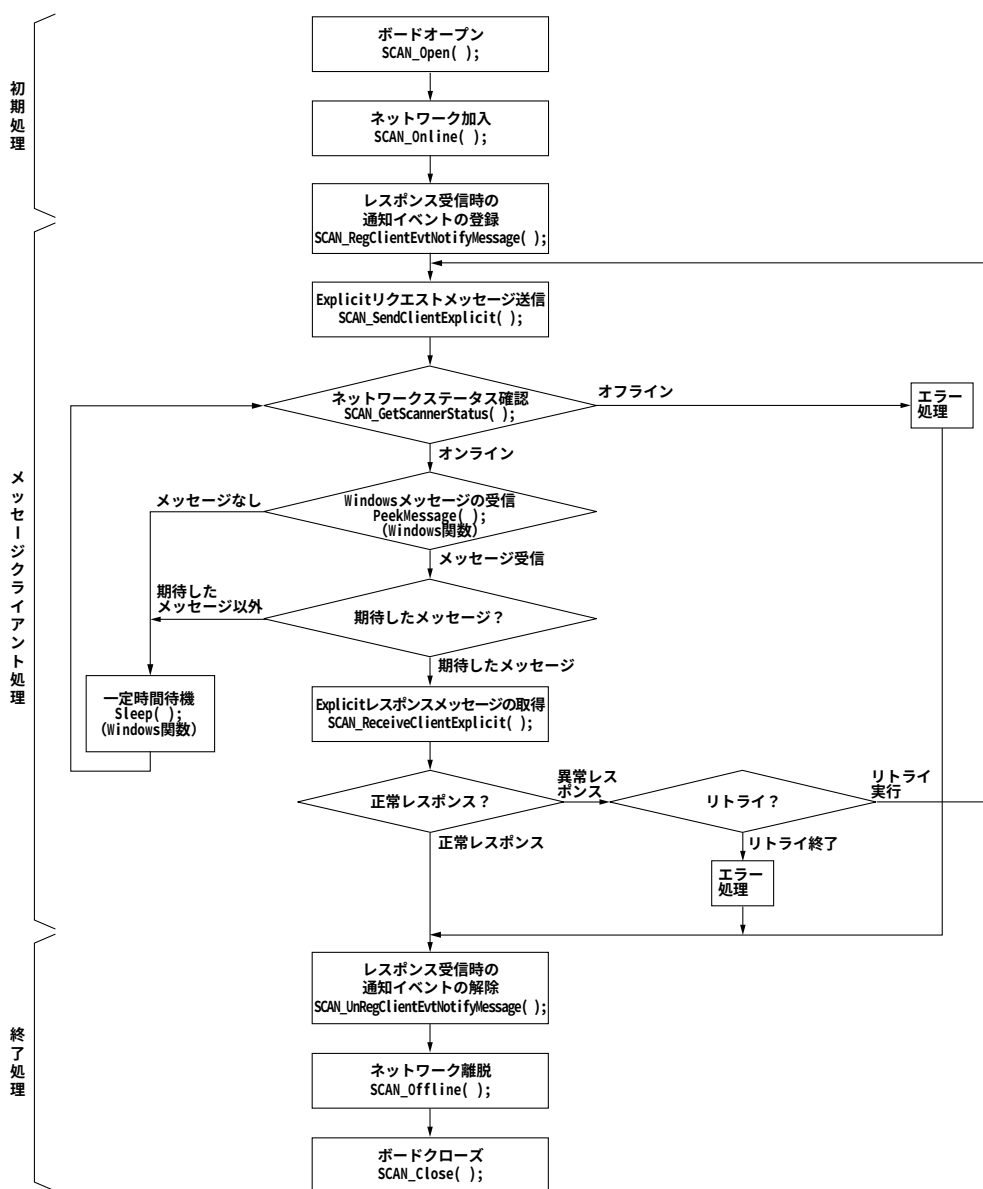
3-7 Explicit メッセージクライアント機能の使用手順

Scanner SDK で Explicit メッセージクライアント機能を使用する場合は、レスポンス受信の検知方法により、2 種類の方法があります。それぞれ以下の図のような手順で API 関数を使用します。使用方法の詳細は、「API 関数リファレンス」(P.4-1)を参照してください。

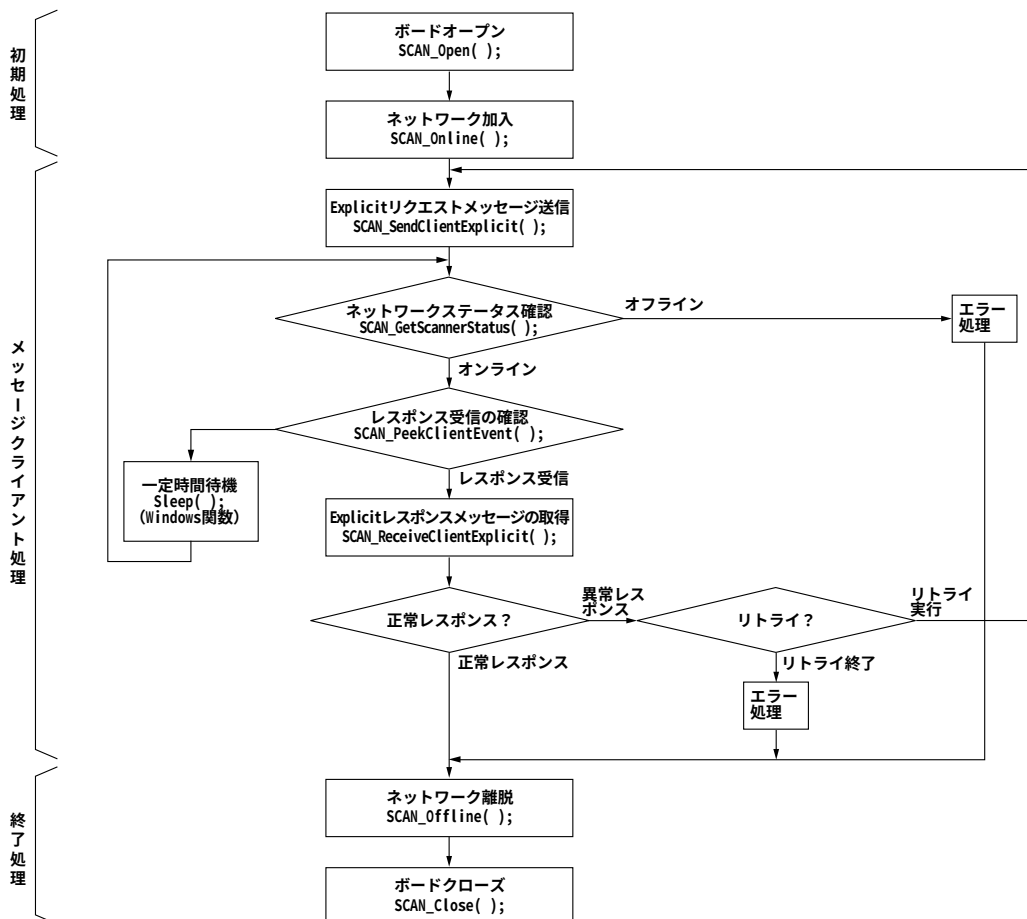
参考

Explicit メッセージ通信では相手ノードの状態によってメッセージを受信できない場合や、エラーレスポンスが返る場合がありますので、必ずリトライ処理を行ってください。

■Windows メッセージによるイベント通知を使用する場合



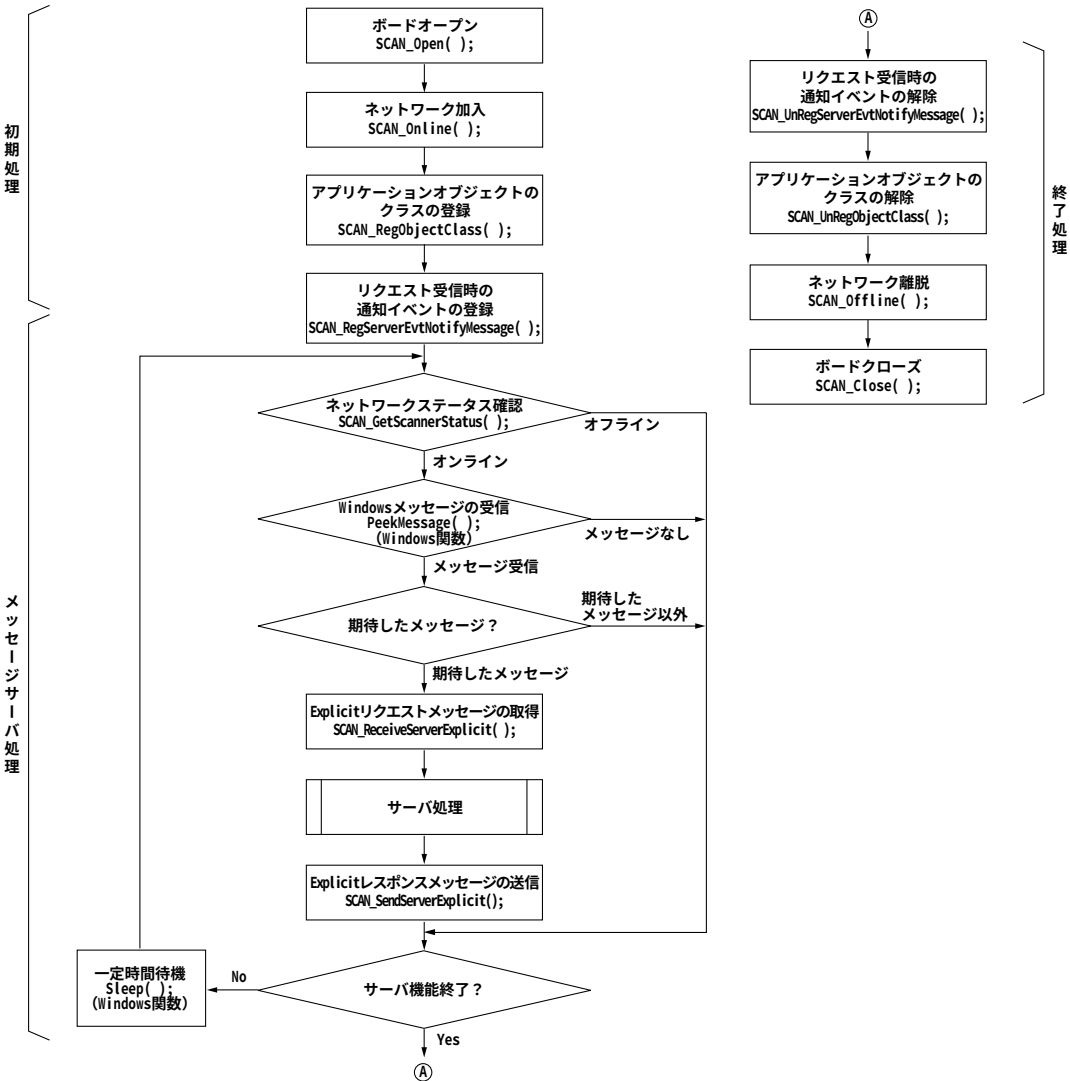
■ポーリングによるイベントチェックを使用する場合



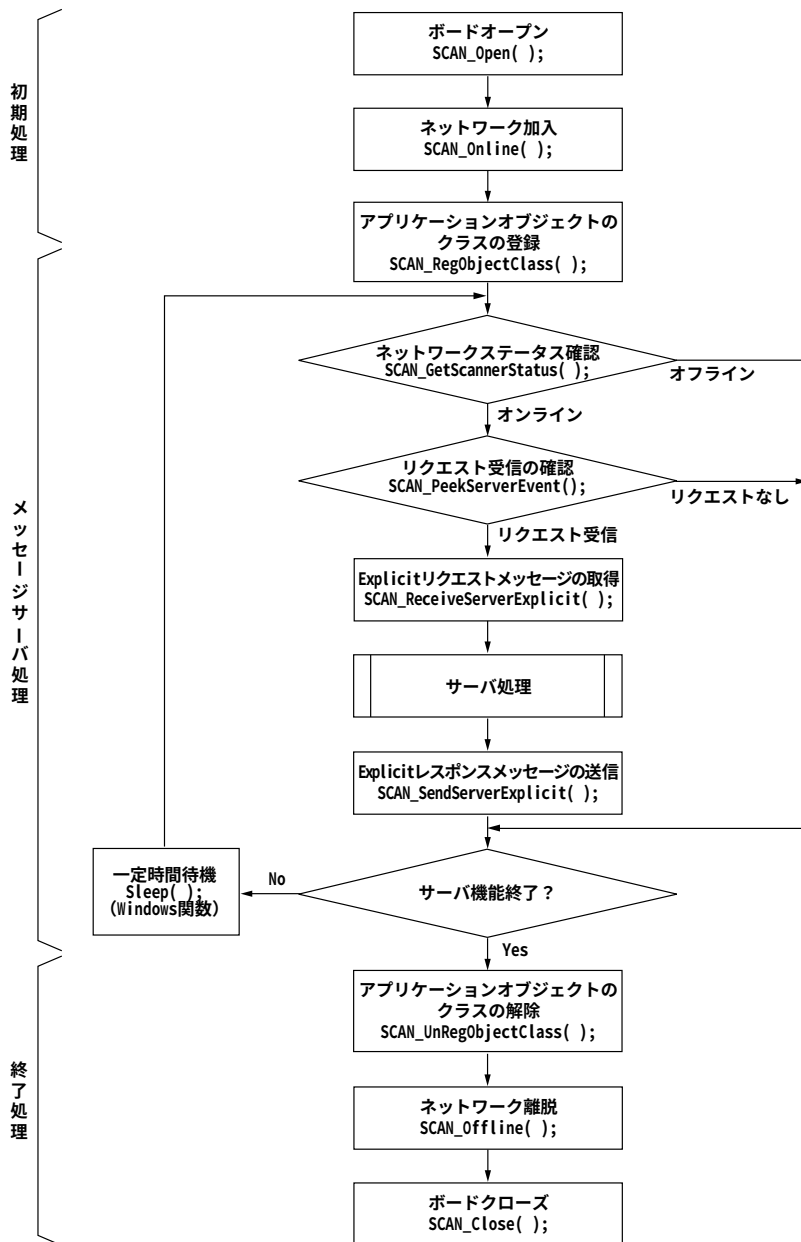
3-8 Explicit メッセージサーバ機能の使用手順

Scanner SDK で Explicit メッセージサーバ機能を使用する場合は、リクエスト受信の検知方法により、2 種類の方法があります。それぞれ以下の図のような手順で API 関数を使用します。使用方法の詳細は、「API 関数リファレンス」(P.4-1) を参照してください。

■Windows メッセージによるイベント通知を使用する場合



■ポーリングによるイベントチェックを使用する場合



3-9 リセット機能の使い方

SCAN_Reset()関数を使用することで、本ボードをリセットし、起動時と同じ状態に初期化することができます。本関数を実行すると、本ボード用に確保されていたすべてのリソースが解放されるので、リセット以前にユーザアプリケーションから設定した情報もすべて失われます。

リセット後は、必要な情報をすべて再設定してください。

参 考

SCAN_Reset()関数の戻り値として TRUE が返された時点で、本ボードのリセットは完了しています。特に時間間隔をあげなくても、すぐに次の関数を実行することができます。

3-10 異常履歴機能の使い方

本ボードには、発生した異常情報を記録・保持する異常履歴機能があります。

異常が発生すると、本ボード内の RAM の異常履歴テーブルに、異常 1 件につき 1 レコード、最大 64 レコード記録されます。異常履歴がテーブルの最大格納数（64 レコード）まで記録されたときは、新たに異常履歴が発生すると、一番古いレコードから順に破棄され、発生した異常データが新たに格納されます。

異常履歴テーブルには、以下の情報が格納されます。

- ・ 異常コード（「■異常履歴データ」7-6 ページ参照）
- ・ 詳細コード（「■異常履歴データ」7-6 ページ参照）
- ・ 異常発生日時（パソコンの時刻情報を使用しています）

異常履歴の保存

異常を検知すると、その内容は、異常の発生時刻とともに異常履歴として本ボード内部の RAM に記録されます。

また、重要度の高い異常については、RAM だけでなく、EEP-ROM にも記録されます（記録される異常コードは、「■異常コード一覧」7-7 ページ参照）。EEP-ROM に記録された異常履歴は、パソコンの電源断または本ボードのリセットが発生しても保持されます。本ボードは立ち上がり時に、EEP-ROM 内の異常履歴を RAM にコピーします。

API 関数を使用して異常履歴の内容を読み出す場合は、RAM 上の異常履歴が対象になります。ただし、異常履歴をクリアする場合は、RAM および EEPROM 両方の異常履歴がクリアされます。

異常履歴の読み出し／クリアの方法

下表の関数により、本ボード内に記録されている異常履歴の読み出しや、クリアを行うことができます。

関数	名称	機能	参照ページ
SCAN_GetErrorLog()	異常履歴読み出し	本ボード内の異常履歴記録データを、一括して読み出します。	P.4-56
SCAN_ClearErrorLog()	異常履歴クリア	本ボード内の異常履歴記録データを、すべてクリアします。	P.4-56

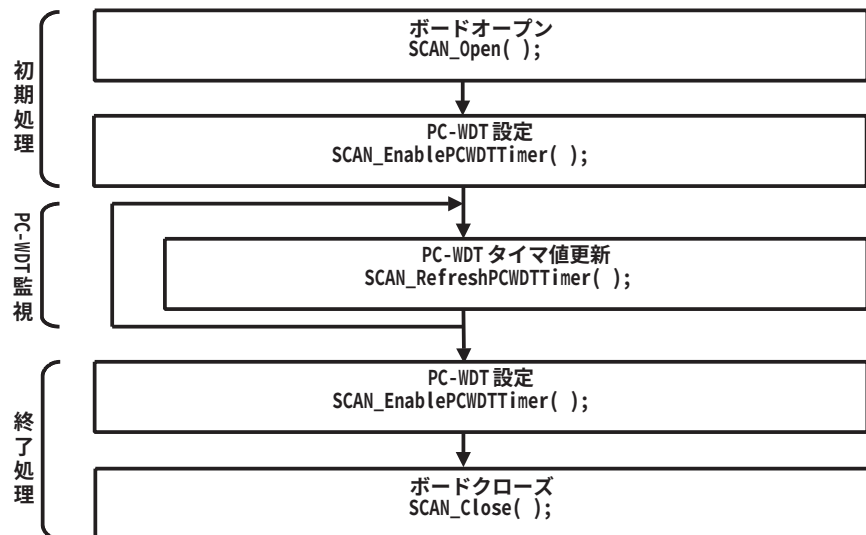
3-11 PC ウォッチドッグタイマ管理機能の使い方

本ボードには、本ボードを制御するユーザアプリケーションが停止したときに、リモート I/O 通信を自動的に停止させることができる、PC ウォッチドッグタイマ機能があります。

PC ウォッチドッグタイマ機能使用中は、一定時間以内にユーザアプリケーションによるタイマ値の更新が実行されないと、自動的にリモート I/O 通信が停止します。

ユーザアプリケーション側では、定期的に PC ウォッチドッグタイマのタイマ値を更新することで、正常に動作していることを本ボードに通知するようにしてください。なお、タイムアウト検知最大遅れは、(監視時間+10ms)×2 となります。

本ボードで PC ウォッチドッグタイマ機能を使用する場合は、下図のような手順で API 関数を使用します。



参 考

本ボード起動時は、PC ウォッチドッグタイマ機能は、無効になっています。また、SCAN_Reset()関数を実行した際にも無効となります。PC ウォッチドッグタイマ機能使用中に、再度 PC-WDT を有効として SCAN_Enable PCWDTTimer()を実行すると、タイマ値が更新されます。

最後の更新から、設定されているタイマ値分の時間が経過しても、更新が行われない場合は、ユーザアプリケーションが停止しているものと判断し、リモート I/O 通信を自動停止します。

3

A P I 関数の使用手順

第 4 章 API 関数リファレンス

この章では、DeviceNet PCI ボード（以下、「本ボード」）で使用する、DN3G8F7 Scanner.DLL 内の API 関数の詳細について説明します。

4-1 関数一覧

■ボード管理用 API

ボードサービス API

DLL およびドライババージョンの確認、指定カードのオープン／クローズなどカードの初期処理および終了処理を提供する API 関数です。

API 関数	概要
SCAN_GetVersion	DLL バージョンを取得する
SCAN_GetDriverVersion	ドライババージョンを取得する
SCAN_IsExistCard	ボードの存在をチェックする
SCAN_Open	ボードをオープンする
SCAN_Close	ボードをクローズする
SCAN_Online	ボードをネットワークに加入（オンライン）にする
SCAN_Offline	ボードをネットワークから離脱（オフライン）にする
SCAN_Reset	ボードをリセットする

PC ボード割り込みサービス API

本ボードから PC への割り込み設定および、割り込み通知メッセージの設定／解除機能を提供する API 関数です。

API 関数	概要
SCAN_GetIrqControl	PC 割り込みコントロールレジスタを取得する
SCAN_SetIrqControl	PC 割り込みコントロールレジスタを設定する
SCAN_RegIrqEvtNotifyMessage	PC 割り込み通知メッセージを登録する
SCAN_UnRegIrqEvtNotifyMessage	PC 割り込み通知メッセージを解除する
SCAN_PeekIrqEvent	PC 割り込みステータスをチェックする
SCAN_ClearIrqEvent	PC 割り込みステータスをクリアする

■マスタ機能用 API

スキャンリスト操作 API

マスタ機能を動作させる場合のスキャンリスト操作を提供する API 関数です。

API 関数	概要
SCAN_RegisterSlaveDevice	スキャンリストへスレーブを登録する
SCAN_GetSlaveDevice	スキャンリストからスレーブ情報を取得する
SCAN_RemoveDevice	スキャンリストからスレーブを削除する
SCAN_ClearScanlist	スキャンリストを消去する
SCAN_StoreScanlist	スキャンリストを不揮発メモリへ保存する
SCAN_LoadScanlist	スキャンリストを不揮発メモリから読み込む
SCAN_RegisterSlaveDeviceEx	スキャンリストへスレーブを登録する（詳細設定）
SCAN_GetSlaveDeviceEx	スキャンリストからスレーブ情報を取得する（詳細設定）
SCAN_SetScanlist	スキャンリストへ複数スレーブを登録する

I/O 通信サービス API

リモート I/O 通信（マスタ機能）の開始／停止、コネクション開設など通信機能を提供する API 関数です。

API 関数	概要
SCAN_StartScan	リモート I/O 通信を開始する
SCAN_StopScan	リモート I/O 通信を停止する
SCAN_SetScanTimeValue	通信サイクル時間を設定する
SCAN_GetScanTimeValue	通信サイクル時間を取得する
SCAN_StoreScanTimeValue	通信サイクル時間を不揮発メモリへ保存する
SCAN_LoadScanTimeValue	通信サイクル時間を不揮発メモリから読み込む
SCAN_GetActualSlaveDevice	実在するスレーブの情報を取得する
SCAN_ConnectSlaveDevice	指定スレーブとの I/O 通信を開始する
SCAN_DisconnectSlaveDevice	指定スレーブとの I/O 通信を停止する

I/O データアクセスサービス API

マスタ機能での I/O データのリフレッシュおよび設定／取得機能を提供する API 関数です。

API 関数	概要
SCAN_IoRefresh	I/O リフレッシュを実行する
SCAN_GetInData	スレーブ入力データを取得する
SCAN_SetOutData	スレーブ出力データを設定する
SCAN_SendMasterCosToSlave	COS 通信をしているマスタからスレーブへ OUT データの送信をする

■スレーブ機能用 API

スレーブスキャンリスト操作 API

スレーブ機能を動作させる場合のスレーブスキャンリスト操作を提供する API 関数です。

API 関数	概要
SCAN_RegisterSelfSlaveDevice	スレーブスキャンリストに登録する
SCAN_RemoveSelfSlaveDevice	スレーブスキャンリストを削除する
SCAN_GetSelfSlaveDevice	スレーブスキャンリスト情報を取得する
SCAN_StoreSlaveScanlist	スレーブスキャンリストを不揮発メモリへ保存する
SCAN_LoadSlaveScanlist	スレーブスキャンリストを不揮発メモリから読み込む

I/O 通信サービス API

リモート I/O 通信（スレーブ機能）の開始／停止の機能を提供する API 関数です。

API 関数	概要
SCAN_ConnectMasterDevice	マスタとの I/O 通信を開始する
SCAN_DisconnectMasterDevice	マスタとの I/O 通信を停止する

I/O データアクセスサービス API

スレーブ機能での I/O データのリフレッシュおよび設定／取得機能を提供する API 関数です。

API 関数	概要
SCAN_SlaveIoRefresh	スレーブ I/O リフレッシュを実行する
SCAN_GetSlaveOutData	マスタ出力データを取得する
SCAN_SetSlaveInData	マスタ入力データを設定する
SCAN_SendSlaveCosToMaster	COS 通信をしているスレーブからマスタへ IN データの送信をする

■Explicit メッセージ用 API

メッセージ監視タイマサービス API

Explicit メッセージの監視タイマ設定機能を提供する API 関数です。

API 関数	概要
SCAN_SetMessageTimerValue	メッセージ監視タイマを設定する
SCAN_GetMessageTimerValue	メッセージ監視タイマを取得する
SCAN_StoreMessageTimerValueList	メッセージ監視タイマリストを不揮発メモリへ保存する
SCAN_LoadMessageTimerValueList	メッセージ監視タイマリストを不揮発メモリから読み込む

クライアントメッセージサービス API

Explicit メッセージのクライアントとして動作する場合の機能を提供する API 関数です。

API 関数	概要
SCAN_RegClientEvtNotifyMessage	クライアントレスポンスイベント通知メッセージを登録する
SCAN_UnRegClientEvtNotifyMessage	クライアントレスポンスイベント通知メッセージを解除する
SCAN_PeekClientEvent	クライアントレスポンスイベントをチェックする
SCAN_GetClientEventLength	クライアントレスポンスのサイズを取得する
SCAN_SendClientExplicit	Explicit クライアントメッセージを送信する
SCAN_ReceiveClientExplicit	Explicit クライアントメッセージを取得する

サーバメッセージサービス API

Explicit メッセージのサーバとして動作する場合の機能を提供する API 関数です。

API 関数	概要
SCAN_RegObjectClass	オブジェクトクラス ID を登録する
SCAN_UnRegObjectClass	オブジェクトクラス ID を解除する
SCAN_RegServerEvtNotifyMessage	サーバリクエストイベント通知メッセージを設定する
SCAN_UnRegServerEvtNotifyMessage	サーバリクエストイベント通知メッセージを解除する
SCAN_PeekServerEvent	サーバリクエストイベントをチェックする
SCAN_GetServerEventLength	サーバリクエストのサイズを取得する
SCAN_ReceiveServerExplicit	Explicit サーバメッセージを取得する
SCAN_SendServerExplicit	Explicit サーバメッセージを送信する

■保守用 API

ステータスサービス API

各種ステータス取得機能を提供する API 関数です。

API 関数	概要
SCAN_GetNetworkStatus	ネットワークステータスを取得する
SCAN_GetScannerStatus	スキャナステータスを取得する
SCAN_GetMasterModeStatus	マスタ機能ステータスを取得する
SCAN_GetSlaveDeviceStatus	スレーブ詳細ステータスを取得する
SCAN_IsScanlistSlaveDeviceRegist	スキャンリストにスレーブが登録されているかチェックする
SCAN_IsDeviceConnection	スレーブとコネクションが開設されているかチェックする
SCAN_GetCycleTime	通信サイクル時間現在値を取得する
SCAN_GetMaxCycleTime	通信サイクル時間最大値を取得する
SCAN_GetMinCycleTime	通信サイクル時間最小値を取得する
SCAN_ClearCycleTime	通信サイクル時間最大値・最小値をクリアする
SCAN_GetSlaveModeStatus	スレーブ機能ステータスを取得する

異常履歴アクセスサービス API

異常履歴アクセス機能を提供する API 関数です。

API 関数	概要
SCAN_GetErrorLog	異常履歴を取得する
SCAN_ClearErrorLog	異常履歴をクリアする

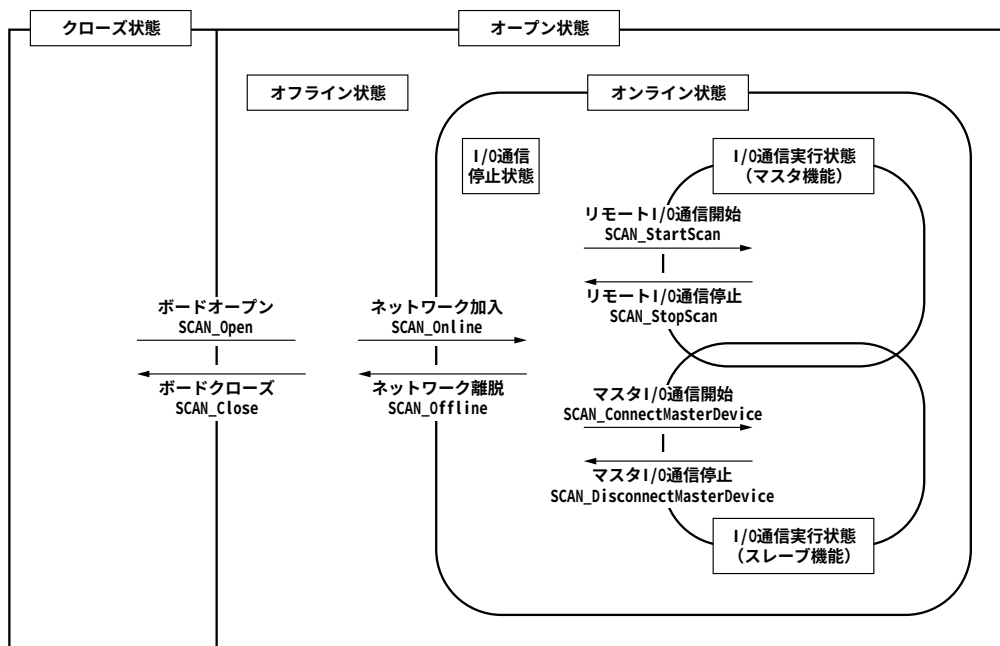
PC ウォッチドッグタイマサービス API

PC-WDT コントロール機能を提供する API 関数です。

API 関数	概要
SCAN_EnablePCWDTTimer	PC ウォッチドッグタイマの有効/無効を設定する
SCAN_RefreshPCWDTTimer	PC ウォッチドッグタイマをリフレッシュする

4-2 ボードの状態

本ボードの状態は、API 関数によって次のように遷移します。本ボードの状態によって使用できる API 関数に制限があります（各関数の説明の「使用範囲」参照）。



クローズ状態：

本ボード用のリソースをすべて解放した状態です。この状態では、本ボード用の制御はほとんど行えません。

本ボード起動時やリセット後は、この状態になっています。

オープン状態：

本ボード用のリソースを確保し、本ボードに対して制御を行える状態です。

SCAN_Open()関数でオープン状態に移行するときに、「デバイスハンドル」が決められます。以後オープン中は、デバイスハンドルによってボードを指定して、関数を使用します。

オフライン状態：

本ボードはオープンされているにもかかわらず、DeviceNet ネットワークからは離脱している状態です。

オンライン状態：

本ボードがオープンされ、DeviceNet ネットワークにも加入している状態です。

リモート I/O 停止状態：

本ボードのオープンと、ネットワークへの加入は行われていても、リモート I/O は停止している状態です。

リモート I/O 実行状態：

本ボードのオープンや、ネットワークへの加入、リモート I/O の実行がすべて行われている状態です。

4-3 ボード管理用 API

■ボードサービス API

SCAN_GetVersion()

DLL バージョン読み出し

使用範囲：制限なし（クローズ状態でも可）

機 能

現在使用している API 関数（DN3G8F7Scanner.DLL）のバージョン情報を取得します。

呼び出し形式

BOOL SCAN_GetVersion(DWORD *Version);

引 数

型	変数名	内容
DWORD *	Version	バージョン情報を受け取るためのバッファアドレス

戻 り 値

正常に処理が完了した場合は TRUE を、エラーが発生した場合には FALSE を返します。詳細なエラー情報は GetLastError() で取得することができます。

説 明

- ・DLL のバージョン確認が必要な場合に使用してください。
- ・DLL バージョンは、次の形式の BCD コードで Version に格納されます。

31	16 15	0 ビット
メジャーバージョン	マイナーバージョン	

例：バージョン 1.13 の場合

Version=0x00010013

SCAN_GetDriverVersion()

ドライババージョン読み出し

使用範囲：オープン状態

機 能

現在使用している本ボード用のドライバのバージョン情報を取得します。

呼び出し形式

BOOL SCAN_GetDriverVersion(DWORD Handle, DWORD *Version);

引 数

型	変数名	内容
DWORD	Handle	SCAN_Open() で得られたデバイスハンドル
DWORD *	Version	ドライババージョン情報を受け取るためのバッファアドレス

戻 り 値

正常に処理が完了した場合は TRUE を、エラーが発生した場合には FALSE を返します。詳細なエラー情報は GetLastError() で取得することができます。

説 明

- ・ドライバのバージョン確認が必要な場合に使用してください。
- ・ドライババージョンは、次の形式の BCD コードで Version に格納されます。

31	16 15	0 ビット
メジャーバージョン	マイナーバージョン	

例：バージョン 1.13 の場合

Version=0x00010013

SCAN_IsExistCard()

ボードチェック

使用範囲：制限なし（クローズ状態でも可）

機 能

指定されたボード ID を持つ本ボードが存在するかチェックします。

呼び出し形式

BOOL SCAN_IsExistCard(DWORD BoardId);

引 数

型	変数名	内容
DWORD	BoardId	存在をチェックしたい本ボードのボード ID 指定範囲：0x0～0x7（0～7）

戻 り 値

指定されたボード ID を持つ本ボードが存在する場合は TRUE を、エラーあるいは該当ボード ID を持つ本ボードが存在しない場合は FALSE を返します。詳細なエラー情報は GetLastError() で取得することができます。

説 明

- ・装着されている本ボードのボード ID をチェックする場合などに使用してください。
- ・ボード ID は、本ボード上のロータリスイッチで設定します。

SCAN_Open()

ボードオープン

使用範囲： **クローズ状態**

機 能

指定されたボード ID を持つ本ボードをオープンし、使用可能にします。

呼び出し形式

BOOL SCAN_Open(DWORD BoardId, DWORD *Handle);

引 数

型	変数名	内容
DWORD	BoardId	オープンしたい本ボードのボード ID 指定範囲：0x0～0x7（0～7）
DWORD *	Handle	デバイスハンドルを受け取るためのバッファアドレス

戻 り 値

指定されたボード ID を持つ本ボードをオープンできた場合は TRUE を、エラーが起きた場合、または該当ボード ID を持つ本ボードが存在しない場合は FALSE を返します。詳細なエラー情報は GetLastError() で取得することができます。

説 明

- ・本ボードを使用するためには、まずオープンが必要です。
- ・1 枚の本ボードは、1 つのアプリケーション（プロセス）からのみオープンできます。
- ・ボード ID は、本ボード上のロータリスイッチで設定した値です。

SCAN_Close()

ボードクローズ

使用範囲： **オープン状態**

機 能

指定されたデバイスハンドルをクローズし、本ボードを使用不可にします。

呼び出し形式

BOOL SCAN_Close(DWORD Handle);

引 数

型	変数名	内容
DWORD	Handle	SCAN_Open()で得られたデバイスハンドル

戻 り 値

指定されたデバイスハンドルを正しくクローズできた場合は TRUE を、エラーが発生した場合は FALSE を返します。詳細なエラー情報は GetLastError()で取得することができます。

説 明

- ・この関数を実行すると、デバイスハンドルのクローズと共に、本ボードはリセットされた状態になり、設定された情報はすべてクリアされます。アプリケーションとして必要な終了処理は、この関数の前に実行してください。

SCAN_Online()

ネットワーク加入（オンライン）

使用範囲： **オフライン状態**

機 能

指定された本ボードをネットワークに加入させ、オンライン状態にします。

呼び出し形式

BOOL SCAN_Online(DWORD Handle, WORD MacId, WORD BaudRate);

引 数

型	変数名	内容
DWORD	Handle	SCAN_Open()で得られたデバイスハンドル
WORD	MacId	本ボードに設定するノードアドレス 指定範囲：0x0～0x3F（0～63）
WORD	BaudRate	本ボードに設定する通信速度 指定範囲： ONLINE125K（0）：125k ビット/s ONLINE250K（1）：250k ビット/s ONLINE500K（2）：500k ビット/s

戻 り 値

指定された本ボードが正常にオンライン状態になった場合は TRUE を、エラーが発生した場合は FALSE を返します。詳細なエラー情報は GetLastError()で取得することができます。

説 明

- ・オンライン処理が完了するまで、約3秒かかります。
- ・次の場合は、FALSE を返し、オンライン状態になりません。
GetLastError()で要因を確認し、処置をしてから、改めてオンラインにしてください。
 - ・バスオフ異常
 - ・ノードアドレス重複
 - ・ネットワーク電源異常
 - ・送信タイムアウト
- ・同一ネットワーク上の他のノードと異なった通信速度を設定した場合、他ノードでネットワークエラーが発生する可能性があります。
- ・通信速度自動認識機能を有したスレーブのみを接続する場合、「送信タイムアウト」が発生することがあります。この場合は、SCAN_Online()のリトライ処理を行ってください。

SCAN_Offline()

ネットワーク離脱（オフライン）

使用範囲： **オンライン状態**

機 能

指定された本ボードをネットワークから離脱させ、オフライン状態にします。

呼び出し形式

BOOL SCAN_Offline(DWORD Handle);

引 数

型	変数名	内容
DWORD	Handle	SCAN_Open()で得られたデバイスハンドル

戻 り 値

指定された本ボードが正常にオフラインになった場合は TRUE を、エラーが発生した場合は FALSE を返します。詳細なエラー情報は GetLastError() で取得することができます。

SCAN_Reset()

ボードリセット

使用範囲： **オープン状態**

機 能

指定された本ボードをリセットします。

呼び出し形式

BOOL SCAN_Reset(DWORD Handle);

引 数

型	変数名	内容
DWORD	Handle	SCAN_Open()で得られたデバイスハンドル

戻 り 値

正しく処理が終了した場合は TRUE を、エラーが発生した場合は FALSE を返します。詳細なエラー情報は GetLastError() で取得することができます。

説 明

- ・この関数を実行すると本ボードはリセットされた状態になり、設定された情報はすべてクリアされます。そのため、ボードオープン以降のすべての処理を再び実行する必要があります。
- ・戻り値として TRUE が返された時点で、本ボードの初期化処理は完了していますので、すぐに次の関数を実行することができます。
- ・リモート I/O 機能を使用していた場合や、本ボードをクライアントとして他ノードと通信していた場合は、本ボードをリセットすると、相手側で通信タイムアウトが発生し、NS LED が赤点滅します。

■PC ボード割り込みサービス API

SCAN_GetIrqControl() 割り込みコントロールレジスタ読み出し

使用範囲： **オープン状態**

機 能 指定された本ボードで発生した割り込みをパソコン側に通知するかどうかを指定する割り込みコントロールレジスタの値を取得します。

呼び出し形式 BOOL SCAN_GetIrqControl(DWORD Handle, BYTE*IrqReg);

引 数	型	変数名	内容
	DWORD	Handle	SCAN_Open() で得られたデバイスハンドル
	BYTE *	IrqReg	割り込みコントロールレジスタの値を受け取るためのバッファアドレス

戻 り 値 指定されたボードより正しくレジスタ値が読み出せた場合は TRUE を、エラーが発生した場合は FALSE を返します。詳細なエラー情報は GetLastError() で取得することができます。

説 明 ・割り込みコントロールレジスタに設定している値を確認する場合などに使用します。
・割り込みコントロールレジスタは、次の形式で IrqReg に格納されます。

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ビット名	予約						BD_WDT	SCAN

BD_WDT : ボード WDT 割り込み
SCAN : 1 スキャン完了割り込み

SCAN_SetIrqControl() 割り込みコントロールレジスタ書き込み

使用範囲： **オープン状態**

機 能 指定された本ボードで発生した割り込みをパソコン側に通知するかどうかを指定する割り込みコントロールレジスタに値を設定します。

呼び出し形式 BOOL SCAN_SetIrqControl(DWORD Handle, BYTE IrqReg);

引 数	型	変数名	内容
	DWORD	Handle	SCAN_Open() で得られたデバイスハンドル
	BYTE	IrqReg	新しく設定するレジスタの値 (※) データフォーマットは、SCAN_GetIrqControl() と同一

戻 り 値 指定されたボードに正しくレジスタ値が書き込めた場合は TRUE を、エラーが発生した場合は FALSE を返します。詳細なエラー情報は GetLastError() で取得することができます。

説 明 ・割り込みコントロールレジスタ設定後、該当割り込みが発生するとパソコン側に通知されます。

SCAN_RegIrqEvtNotifyMessage() 割り込み通知メッセージ登録

使用範囲： **オープン状態**

機 能

本ボードからの割り込みが発生した場合に通知する Windows メッセージを登録します。

呼び出し形式

BOOL SCAN_RegIrqEvtNotifyMessage
(DWORD Handle, DWORD ThreadId, HWND, hWnd, UNIT Msg);

引 数

型	変数名	内容
DWORD	Handle	SCAN_Open()で得られたデバイスハンドル
DWORD	ThreadId	通知するスレッド ID (未設定：NULL)
HWND	hWnd	通知するウィンドウハンドル (未設定：NULL)
UNIT	Msg	通知メッセージ 指定範囲：WM_USER+0x100～WM_USER+0x7FFF

戻 り 値

通知メッセージの登録が正しく行われた場合は TRUE を、スレッド ID およびウィンドウハンドルの両方が NULL 指定の場合など、エラーが発生した場合は FALSE を返します。詳細なエラー情報は GetLastError()で取得することができます。

説 明

- ・イベント通知先として、スレッド ID またはウィンドウハンドルを指定してください。
- ・通知メッセージと共に WPARAM に割り込みステータスが、LPARAM に通知メッセージが通知されます。

SCAN_UnRegIrqEvtNotifyMessage() 割り込み通知メッセージ解除

使用範囲： **オープン状態**

機 能

通知メッセージ登録を解除します。

呼び出し形式

BOOL SCAN_UnRegEvtNotifyMessage(DWORD Handle);

引 数

型	変数名	内容
DWORD	Handle	SCAN_Open()で得られたデバイスハンドル

戻 り 値

通知メッセージの登録解除が正しく行われた場合は TRUE を、エラーが発生した場合は FALSE を返します。詳細なエラー情報は GetLastError()で取得することができます。

SCAN_PeekIrqEvent()

割り込みイベントチェック

使用範囲： **オープン状態****機 能**

指定されたボードで発生した割り込み要因をチェックします。

呼び出し形式

BOOL SCAN_PeekIrqEvent (DWORD Handle, BYTE *IrqStatus);

引 数

型	変数名	内容
DWORD	Handle	SCAN_Open()で得られたデバイスハンドル
BYTE *	IrqStatus	発生した割り込み要因を受け取るためのバッファアドレス (※) データフォーマットは、SCAN_GetIrqControl()と同一

戻 り 値

指定されたボードより正しくステータス値が読み出せた場合は TRUE を、エラーが発生した場合は FALSE を返します。詳細なエラー情報は GetLastError()で取得することができます。

説 明

- ・指定されたボードで割り込みステータスをチェックするためには、SCAN_SetIrqControl()を使用して、ボードで発生した割り込みをパソコン側に通知する指定をしておく必要があります。

SCAN_ClearIrqEvent()

割り込みステータスクリア

使用範囲： **オープン状態****機 能**

指定されたボードで発生した割り込み要因をクリアします。

呼び出し形式

BOOL SCAN_ClearIrqEvent(DWORD Handle, BYTE IrqClrMask);

引 数

型	変数名	内容
DWORD	Handle	SCAN_Open()で得られたデバイスハンドル
BYTE	IrqClrMask	クリアする割り込みマスク 該当割り込みビット位置を ON することにより、クリア指定となります。 (※) データフォーマットは、SCAN_GetIrqControl()と同一

戻 り 値

指定されたボードのレジスタに正しく値が書き込めた場合は TRUE を、エラーが発生した場合は FALSE を返します。詳細なエラー情報は GetLastError()で取得することができます。

説 明

- ・イベント通知または SCAN_PeekIrqEvent()によるポーリングチェックによって割り込みを検知し、割り込みに対応した処理を実行した後、対応する割り込み要因をこの関数でクリアしてください。

4-4 マスタ機能用 API

■スキャンリスト操作 API

SCAN_RegisterSlaveDevice()

スキャンリスト スレーブ登録

使用範囲: **オープン状態**

機能

スキャンリストに指定スレーブの情報を登録します。

呼び出し形式

BOOL SCAN_RegisterSlaveDevice
(DWORD Handle, WORD MacId, WORD Outsize, WORD Insize);

引数

型	変数名	内容
DWORD	Handle	SCAN_Open()で得られたデバイスハンドル
WORD	MacId	スキャンリストに登録するアドレス 指定範囲: 0x00~0x3F (0~63)
WORD	Outsize	出力データサイズ (バイト単位) 指定範囲: 0x00~0xC8 (0~200)
WORD	Insize	入力データサイズ (バイト単位) 指定範囲: 0x00~0xC8 (0~200)

戻り値

指定されたスレーブをスキャンリストに正常に登録できた場合は TRUE を、エラーが発生した場合は FALSE を返します。詳細なエラー情報は GetLastError() で取得することができます。

説明

- ・スキャンリストに登録されているスレーブが I/O 通信の対象となります。メッセージ通信の場合は、相手が I/O 通信の対象でなければ、スキャンリストに登録する必要はありません。
- ・既に登録されているスレーブを再登録すると上書きされます。
- ・この関数で登録したスレーブは、スキャンタイプが自動選択になります。スキャンタイプの指定などを行う場合は、SCAN_RegisterSlaveDeviceEx() を使用してください。
- ・この関数で登録したスレーブの I/O データは、出力データ 1、入力データ 1 となります。

SCAN_GetSlaveDevice()

スキャンリスト スレーブ読み出し

使用範囲: **オープン状態**

機能

スキャンリストに登録されているスレーブ情報を取得します。

呼び出し形式

BOOL SCAN_GetSlaveDevice
(DWORD Handle, WORD MacId, WORD *Outsize, WORD *Insize);

引数

型	変数名	内容
DWORD	Handle	SCAN_Open()で得られたデバイスハンドル
WORD	MacId	スレーブ情報を取得するノードアドレス 指定範囲: 0x00~0x3F (0~63)
WORD*	Outsize	出力データサイズを受け取るバッファアドレス
WORD*	Insize	入力データサイズを受け取るバッファアドレス

戻り値

指定されたスレーブをスキャンリストから正常に取得できた場合は TRUE を、エラーが発生した場合は FALSE を返します。詳細なエラー情報は GetLastError() で取得することができます。

説明

- ・この関数終了後、指定したアドレスのバッファに、出力データサイズと入力データサイズ (共に、バイト単位で 0~200) が格納されます。

SCAN_RemoveDevice()

スキャンリスト スレーブ削除

使用範囲: オープン状態

機 能

スキャンリストから、指定スレーブの登録情報を削除します。

呼び出し形式

BOOL SCAN_RemoveDevice(DWORD Handle, WORD MacId);

引 数

型	変数名	内容
DWORD	Handle	SCAN_Open() で得られたデバイスハンドル
WORD	MacId	削除対象のノードアドレス 指定範囲: 0x00~0x3F (0~63)

戻 り 値

指定されたスレーブをスキャンリストから正常に削除できた場合は TRUE を、エラーが発生した場合は FALSE を返します。詳細なエラー情報は GetLastError() で取得することができます。

SCAN_ClearScanlist()

スキャンリスト クリア

使用範囲: オープン状態

機 能

スキャンリストを一括削除します。

呼び出し形式

BOOL SCAN_ClearScanlist(DWORD Handle);

引 数

型	変数名	内容
DWORD	Handle	SCAN_Open() で得られたデバイスハンドル

戻 り 値

指定された本ボードのスキャンリストを正常に削除できた場合は TRUE を、エラーが発生した場合は FALSE を返します。詳細なエラー情報は GetLastError() で取得することができます。

SCAN_StoreScanlist()

スキャンリスト ストア

使用範囲: オープン状態

機 能

スキャンリストの情報を不揮発メモリへ保存します。

呼び出し形式

BOOL SCAN_StoreScanlist(DWORD Handle);

引 数

型	変数名	内容
DWORD	Handle	SCAN_Open() で得られたデバイスハンドル

戻 り 値

スキャンリストを正常に不揮発メモリに書き込んだ場合は TRUE を、エラーが発生した場合は FALSE を返します。詳細なエラー情報は GetLastError() で取得することができます。

説 明

- ・不揮発性メモリに保存したスキャンリストの情報は、SCAN_LoadScanlist() で読み込み、スキャンリストとして登録することができます。

使用範囲: オープン状態

機能

スキャンリストの情報を不揮発メモリから読み込み、スキャンリストとして登録します。

呼び出し形式

BOOL SCAN_LoadScanlist(DWORD Handle);

引数

型	変数名	内容
DWORD	Handle	SCAN_Open()で得られたデバイスハンドル

戻り値

スキャンリストを正常に不揮発メモリから読み込んだ場合は TRUE を、エラーが発生した場合は FALSE を返します。詳細なエラー情報は GetLastError()で取得することができます。

説明

- ・不揮発性メモリへスキャンリストの情報を保存するには、SCAN_StoreScanlist()を使用します。
- ・スキャンリストの情報を読み込むと、今までのスキャンリストに上書きされ、新しく読み込んだスキャンリストが有効になります。

SCAN_DEV 構造体

スキャンリストにノード(マスタ)を登録する際の、登録情報の形式を定義する構造体です。
SCAN_RegisterSlaveDeviceEx()関数と SCAN_GetSlaveDeviceEx()関数で使用されます。
また、スキャンリストの一括登録の際に使用される FSCAN_DEV 構造体と
FILE_SCAN_DEV 構造体でも使用されます。

型	変数名	内容													
WORD	VendorID	ベンダ ID ベンダ ID の照合チェックをする場合は、照合するベンダ ID を指定してください。 照合チェックしない場合 : 0xFFFF (65535)													
WORD	ProductType	プロダクトタイプ プロダクトタイプの照合チェックをする場合は、照合するプロダクトタイプを指定してください。 照合チェックしない場合 : 0xFFFF (65535)													
WORD	ProductCode	プロダクトコード プロダクトコードの照合チェックをする場合は、照合するプロダクトコードを指定してください。 照合チェックしない場合 : 0xFFFF (65535)													
WORD	ScanType	スキャンタイプ 使用するコネクションの種類を指定します。 最大 2 つの選択が可能です。ただし、COS、サイクリックの同時選択はできません。 自動選択時は、0x8000 (32768) を指定します。 <table border="1"><tr><td>Bit5</td><td>Bit4</td><td>Bit3</td><td>Bit2</td><td>Bit1</td><td>Bit0</td><td rowspan="2">BS : Bit-Strobe CY : サイクリック</td></tr><tr><td>CY</td><td>COS</td><td>予約</td><td>BS</td><td>Poll</td><td>予約</td></tr></table>	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	BS : Bit-Strobe CY : サイクリック	CY	COS	予約	BS	Poll	予約
Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	BS : Bit-Strobe CY : サイクリック									
CY	COS	予約	BS	Poll	予約										
WORD	Output1	出力データ 1 サイズ (バイト単位) : 0x00～0xC8 (0～200)													
WORD	Input1	入力データ 1 サイズ (バイト単位) : 0x00～0xC8 (0～200)													
WORD	Output2	出力データ 2 サイズ (バイト単位) : 0x00～0xC8 (0～200) スキャンタイプに自動選択を指定する場合、またはコネクションを 1 つだけ指定する場合は、このエリアは 0 としてください。													
WORD	Input2	入力データ 2 サイズ (バイト単位) : 0x00～0xC8 (0～200) スキャンタイプに自動選択を指定する場合、またはコネクションを 1 つだけ指定する場合は、このエリアは 0 としてください。													
WORD	ConnectAccept	接続/非接続 非接続時 : 0xFFFF (65535) 接続時 : 左記以外 通常は 0 (接続時) としてください。 後からスレーブを追加する予定があり、スキャンリストに予約する場合には、0xFFFF (非接続時) としてください。この場合、スキャンリストに登録されていても I/O 通信を行いません。また、通信異常も発生しません。スレーブ追加後は、SCAN_ConnectSlaveDevice()により、I/O 通信を開始することができます。													
WORD	HeartBeatTime	COS/サイクリック送信間隔時間 (ms) 指定範囲： 0x0000 (0)													

*: コネクションパスを指定しない場合は、コネクションパスサイズを 0、コネクションパスを NULL としてください。

型	変数名	内容
WORD	Output2PathLen	出力データ 2 コネクションパスサイズ * : 0x00~0x10 (0~16) スキャンタイプに自動選択を指定する場合、またはコネクションを 1 つだけ指定する場合は、このエリアは 0 としてください。
WORD	Output2Path[16]	出力データ 2 コネクションパス *
WORD	Input2PathLen	入力データ 2 コネクションパスサイズ * : 0x00~0x10 (0~16) スキャンタイプに自動選択を指定する場合、またはコネクションを 1 つだけ指定する場合は、このエリアは 0 としてください。
WORD	Input2Path[16]	入力データ 2 コネクションパス *
WORD	Reserve	予約エリアのため、0 を設定してください。
WORD	Reserve2	予約エリアのため、0 を設定してください。

* : コネクションパスを指定しない場合は、コネクションパスサイズを 0、コネクションパスを NULL としてください。

SCAN_RegisterSlaveDeviceEx()

スキャンリスト スレーブ登録(詳細指定)

使用範囲： **オープン状態****機 能**

スレーブの詳細情報を指定して、スキャンリストに登録します。

呼び出し形式

```
BOOL SCAN_RegisterSlaveDeviceEx
(DWORD Handle, WORD, MacId, SCAN_DEV *DeviceInfo);
```

引 数

型	変数名	内容
DWORD	Handle	SCAN_Open() で得られたデバイスハンドル
WORD	MacId	スキャンリストに登録するノードアドレス 指定範囲：0x00～0x3F (0～63)
SCAN_DEV *	Device	スレーブの詳細情報を格納したバッファアドレス

戻 り 値

指定されたスレーブをスキャンリストに正常に登録できた場合は TRUE を、エラーが発生した場合は FALSE を返します。詳細なエラー情報は GetLastError() で取得することができます。

説 明

- ・既にスキャンリストに登録されているスレーブ情報は上書きされます。

SCAN_GetSlaveDeviceEx()

スキャンリスト スレーブ読み出し(詳細指定)

使用範囲： **オープン状態****機 能**

スキャンリストに登録されているスレーブの詳細情報を取得します。

呼び出し形式

```
BOOL SCAN_GetSlaveDeviceEx(DWORD Handle, WORD, MacId, SCAN_DEV *DeviceInfo);
```

引 数

型	変数名	内容
DWORD	Handle	SCAN_Open() で得られたデバイスハンドル
WORD	MacId	スレーブ情報を取得するノードアドレス 指定範囲：0x00～0x3F (0～63)
SCAN_DEV *	Device	スレーブの詳細情報を受け取るバッファアドレス

戻 り 値

指定されたスレーブをスキャンリストから正常に取得できた場合は TRUE を、エラーが発生した場合は FALSE を返します。詳細なエラー情報は GetLastError() で取得することができます。

FSCAN_DEV 構造体

スキャンリストにノードを一括登録する際の、個々のスレーブ登録情報の形式を定義する構造体です。

FILE_SCAN_DEV 構造体内で使用され、SCAN_SetScanList()関数で使用されます。

型	変数名	内容
WORD	MacId	登録するノードアドレス
SCAN_DEV	ScanDevice	スレーブ情報

FILE_SCAN_DEV 構造体

スキャンリストにノードを一括登録する際の、全スレーブの登録情報をまとめた構造体です。

SCAN_SetScanList()関数で使用されます。

型	変数名	内容
WORD	DeviceCount	設定スレーブ数
FSCAN_DEV	Device [DEVICE_MAX]	スキャンスレーブ情報

SCAN_SetScanlist()

スキャンリスト 複数スレーブ登録

使用範囲: **オープン状態**

機 能

スキャンリストへ複数のスレーブをまとめて登録します。

呼び出し形式

BOOL SCAN_SetScanlist(DWORD Handle, LPCTSTR FilePath);

引 数

型	変数名	内容
DWORD	Handle	SCAN_Open()で得られたデバイスハンドル
LPCTSTR	LPCTSTR	スキャンリストファイルパスバッファアドレス (※) FILE_SCAN_DEV 構造体フォーマットファイル

戻 り 値

スキャンリストを正常に登録できた場合は TRUE を、エラーが発生した場合は FALSE を返します。詳細なエラー情報は GetLastError()で取得することができます。

■ I/O 通信サービス API

SCAN_StartScan()

リモート I/O 通信開始

使用範囲： **オンライン状態**

機 能

リモート I/O を開始します。

呼び出し形式

BOOL SCAN_StartScan(DWORD Handle, BOOL ErrStop);

引 数

型	変数名	内容
DWORD	Handle	SCAN_Open()で得られたデバイスハンドル
BOOL	ErrStop	通信異常時リモート I/O 停止/継続指定 TRUE : 通信異常時停止 FALSE : 停止しない

戻 り 値

指定した本ボードのリモート I/O が正常に開始された場合は TRUE を、エラーが発生した場合は FALSE を返します。詳細なエラー情報は GetLastError()で取得することができます。

説 明

- ・スキャンリストに登録されているスレーブとの間で、リモート I/O 通信を行います。
- ・この関数を呼び出してから、実際に I/O 通信が開始するまで、数秒の時間を必要とします。
- ・この関数を呼び出す前に、スキャンリスト操作 API 関数を使用して、スキャンリストを作成しておく必要があります。
- ・ErrStop に TRUE を指定した場合、リモート I/O 実行中に通信異常が発生すると、すべてのスレーブとのリモート I/O を停止します。

SCAN_StopScan()

リモート I/O 通信停止

使用範囲： **I/O通信実行状態**

機 能

リモート I/O を停止します。

呼び出し形式

BOOL SCAN_StopScan(DWORD Handle);

引 数

型	変数名	内容
DWORD	Handle	SCAN_Open()で得られたデバイスハンドル

戻 り 値

指定した本ボードのリモート I/O が正常に停止できた場合は TRUE を、エラーが発生した場合は FALSE を返します。詳細なエラー情報は GetLastError()で取得することができます。

SCAN_SetScanTimeValue()

通信サイクル時間書き込み

使用範囲： **オープン状態**

機 能

通信サイクル時間を設定します。

呼び出し形式

BOOL SCAN_SetScanTimeValue(DWORD Handle, WORD ScanTime);

引 数

型	変数名	内容
DWORD	Handle	SCAN_Open()で得られたデバイスハンドル
WORD	ScanTime	通信サイクル時間 (ms) 指定範囲： 0x0000 (0) : 自動 (常に最速で実行します) 0x0001~0x01F4 (1~500) : 1~500ms

戻 り 値

通信サイクル時間が正常に設定できた場合は TRUE を、エラーが発生した場合は FALSE を返します。詳細なエラー情報は GetLastError() で取得することができます。

説 明

- 通信サイクル時間は、デフォルトで自動 (最速) に設定されています。自動設定での通信サイクル時間は、「通信サイクル時間」 (P.6-1) の式により算出された値となります。
- 自動設定で使用する通信サイクル時間より短い値を設定した場合は、指定した値が設定値となりますが、実際の通信は自動設定での通信サイクル時間で行われます。

SCAN_GetScanTimeValue()

通信サイクル時間読み出し

使用範囲： **オープン状態**

機 能

設定されている通信サイクル時間を取得します。

呼び出し形式

BOOL SCAN_GetScanTimeValue(DWORD Handle, WORD *ScanTime);

引 数

型	変数名	内容
DWORD	Handle	SCAN_Open()で得られたデバイスハンドル
WORD *	ScanTime	通信サイクル時間を受け取るためのバッファアドレス 取得範囲： 0x0000 (0) : 自動 (常に最速で実行しています) 0x0001~0x01F4 (1~500) : 1~500ms

戻 り 値

通信サイクル時間を正常に取得できた場合は TRUE を、エラーが発生した場合は FALSE を返します。詳細なエラー情報は GetLastError() で取得することができます。

説 明

- この関数は、設定値 (ms 単位) を取得します。
- 実際の通信サイクル時間を取得する場合は、SCAN_GetCycleTime() や SCAN_GetMaxCycleTime(), SCAN_GetMinCycleTime() を使用してください。

SCAN_StoreScanTimeValue()

通信サイクル時間ストア

使用範囲: オープン状態

機 能

設定されている通信サイクル時間を不揮発メモリへ保存します。

呼び出し形式

BOOL SCAN_StoreScanTimeValue(DWORD Handle);

引 数

型	変数名	内容
DWORD	Handle	SCAN_Open()で得られたデバイスハンドル

戻 り 値

通信サイクル時間を正常に不揮発メモリに書き込めた場合は TRUE を、エラーが発生した場合は FALSE を返します。詳細なエラー情報は GetLastError()で取得することができます。

説 明

- ・不揮発性メモリに保存された通信サイクル時間の値は、SCAN_LoadScanTimeValue()を使用することで、有効になります。

SCAN_LoadScanTimeValue()

通信サイクル時間ロード

使用範囲: オープン状態

機 能

通信サイクル時間を不揮発メモリから読み込み、設定値とします。

呼び出し形式

BOOL SCAN_LoadScanTimeValue(DWORD Handle);

引 数

型	変数名	内容
DWORD	Handle	SCAN_Open()で得られたデバイスハンドル

戻 り 値

通信サイクル時間を正常に不揮発メモリから読み込めた場合は TRUE を、エラーが発生した場合は FALSE を返します。詳細なエラー情報は GetLastError()で取得することができます。

説 明

- ・この関数を、一度も SCAN_SetScanTimeValue()を実行しないで使用した場合は、通信サイクル時間はデフォルトである自動設定となります。

ACTUAL_SLAVE_INFO 構造体

ネットワーク上に存在するスレーブの情報を取得する際に、取得情報が格納される構造体です。SCAN_GetActualSlaveDevice()関数で 사용됩니다。

型	変数名	内容												
WORD	VendorID	ベンダ ID スキャンリスト無効モード時 : 0xFFFF (65535) スレーブ不在時 : 0xFF00 (65280)												
WORD	ProductType	プロダクトタイプ スキャンリスト無効モード時 : 0xFFFF (65535) スレーブ不在時 : 0xFF00 (65280)												
WORD	ProductCode	プロダクトコード スキャンリスト無効モード時 : 0xFFFF (65535) スレーブ不在時 : 0xFF00 (65280)												
WORD	ScanType	スキャンタイプ 自動選択時は、0x8000 (32768) <table><tr><td>Bit5</td><td>Bit4</td><td>Bit3</td><td>Bit2</td><td>Bit1</td><td>Bit0</td></tr><tr><td>CY</td><td>COS</td><td>予約</td><td>BS</td><td>Poll</td><td>予約</td></tr></table> BS : Bit-Strobe CY : サイクリック	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	CY	COS	予約	BS	Poll	予約
Bit5	Bit4	Bit3	Bit2	Bit1	Bit0									
CY	COS	予約	BS	Poll	予約									
WORD	Output1	出力データ 1 サイズ (バイト単位) スレーブ不在時 : 0xFF00 (65280)												
WORD	Input1	入力データ 1 サイズ (バイト単位) スレーブ不在時 : 0xFF00 (65280)												
WORD	Output2	出力データ 2 サイズ (バイト単位) 2つの I/O コネクションを 設定していない場合 : 0x0000 (0) スレーブ不在時 : 0xFF00 (65280)												
WORD	Input2	入力データ 2 サイズ (バイト単位) 2つの I/O コネクションを 設定していない場合 : 0x0000 (0) スレーブ不在時 : 0xFF00 (65280)												

SCAN_GetActualSlaveDevice()

スレーブ情報読み出し

使用範囲: I/O通信実行状態

機 能

指定されたスレーブの情報を取得します。

呼び出し形式

```
BOOL SCAN_GetActualSlaveDevice
(DWORD Handle, WORD MacId, ACTUAL_SLAVE_INFO *SlaveInfo);
```

引 数

型	変数名	内容
DWORD	Handle	SCAN_Open()で得られたデバイスハンドル
WORD	MacId	スレーブ情報を読み出すノードアドレス
ACTUAL_SLAVE_INFO*	SlaveInfo	スレーブ情報を受け取るためのバッファアドレス 関数終了時、ここで指定したアドレスのバッファにスレーブ情報が格納されます。

戻 り 値

指定されたスレーブ情報を正常に取得できた場合は TRUE を、エラーが発生した場合は FALSE を返します。詳細なエラー情報は GetLastError()で取得することができます。

説 明

- ・この関数は、ネットワーク上にある実際のスレーブから情報を取得します。
- ・照合異常が発生しているスレーブについても、実際のスレーブ情報を取得することができます。
- ・スキャンリストに登録されているスレーブ情報を取得する場合は、SCAN_GetSlaveDevice()または SCAN_GetSlaveDeviceEx()を使用してください。

SCAN_ConnectSlaveDevice()

スレーブ I/O 通信開始

使用範囲: I/O通信実行状態

機 能

指定されたスレーブとの I/O 通信を開始します。

呼び出し形式

```
BOOL SCAN_ConnectSlaveDevice(DWORD Handle, WORD MacId);
```

引 数

型	変数名	内容
DWORD	Handle	SCAN_Open()で得られたデバイスハンドル
WORD	MacId	対象のノードアドレス

戻 り 値

指定されたスレーブとの I/O 通信が正常に開始できた場合は TRUE を、エラーが発生した場合は FALSE を返します。詳細なエラー情報は GetLastError()で取得することができます。

説 明

- ・この関数は、I/O 通信実行状態のとき使用できます。
- ・I/O 通信実行中に、特定のスレーブの I/O 通信停止状態（離脱状態）を解除し、I/O 通信を開始します。
- ・スレーブの交換や、追加予定のスレーブの予約（スキャンリストに登録しておくが、I/O 通信は行わないとき）のために、SCAN_DisconnectSlaveDevice()と組み合わせて使用します。

SCAN_DisconnectSlaveDevice()

スレーブ I/O 通信停止

使用範囲： I/O通信実行状態

機 能

指定されたスレーブとの I/O 通信を停止します。

呼び出し形式

BOOL SCAN_DisconnectSlaveDevice(DWORD Handle, WORD MacId);

引 数

型	変数名	内容
DWORD	Handle	SCAN_Open() で得られたデバイスハンドル
WORD	MacId	対象のノードアドレス

戻 り 値

指定されたスレーブとの I/O 通信が正常に停止できた場合は TRUE を、エラーが発生した場合は FALSE を返します。詳細なエラー情報は GetLastError() で取得することができます。

説 明

- ・この関数は、I/O 通信実行状態のとき使用できます。
- ・I/O 通信実行中に、特定のスレーブを I/O 通信停止状態（離脱状態）にして、I/O 通信を停止します。
- ・スレーブの交換や、追加予定のスレーブの予約（スキャンリストに登録しておくが、I/O 通信は行わないとき）のために、SCAN_ConnectSlaveDevice() と組み合わせて使用します。
- ・この関数で I/O 通信を停止（離脱）したスレーブは、照合異常や通信異常になりません。

■ I/O データアクセスサービス API

SCAN_IoRefresh()

マスタ 入出力データ更新

使用範囲： オープン状態

機 能

マスタ機能において、全スレーブの I/O エリアのデータを最新の状態に更新します。

呼び出し形式

BOOL SCAN_IoRefresh(DWORD Handle);

引 数

型	変数名	内容
DWORD	Handle	SCAN_Open() で得られたデバイスハンドル

戻 り 値

指定された本ボードのマスタ機能 IN・OUT エリアの更新が正常に終了した場合は TRUE を、エラーが発生した場合は FALSE を返します。詳細なエラー情報は GetLastError() で取得することができます。

説 明

- ・この関数を使用することで、API 関数でアクセスできる I/O エリアと、実際にボード内で通信に使われている I/O エリアの間でのデータ交換が実行され、最新の I/O 状態となります。
- ・この関数は、全スレーブの I/O エリアを対象とします。
- ・IN データは、この関数終了後に SCAN_GetInData() で取得してください。
- ・OUT データについては、必要なスレーブに対して SCAN_SetOutData() を実行した後で、この関数を実行してください。

IO_DATA_CTL 構造体

本ボード内の入出力エリアとのアクセスデータ形式を定義する構造体です。
 SCAN_GetInData(), SCAN_SetOutData(), SCAN_GetSlaveOutData(),
 SCAN_SetSlaveInData()関数で 사용됩니다。

型	変数名	内容
WORD	DataSize	データサイズ (バイト単位) 指定範囲: 0x0~0xC8 (0~200)
BYTE	Data[200]	データ格納バッファ

データサイズには、読み出しデータサイズ、または書き込みデータサイズを指定してください。

SCAN_GetInData()

スレーブ 入力データ読み出し

使用範囲: **オープン状態**

機 能

本ボード内の IN エリアから、指定されたスレーブの入力データを読み出します。

呼び出し形式

```
BOOL SCAN_GetInData(DWORD Handle,
                     WORD MacId, IO_DATA_CTL *InData1, IO_DATA_CTL *InData2);
```

引 数

型	変数名	内容
DWORD	Handle	SCAN_Open()で得られたデバイスハンドル
WORD	MacId	入力データを読み出すノードアドレス 指定範囲: 0x0~0x3F (0~63)
IO_DATA_CTL*	InData1	入力データ 1 格納バッファアドレス
IO_DATA_CTL*	InData2	入力データ 2 格納バッファアドレス (未使用時は NULL)

戻 り 値

指定された入力データを正常に読み出せた場合は TRUE を、エラーが発生した場合は FALSE を返します。詳細なエラー情報は GetLastError()で取得することができます。

説 明

- ・本ボード内の IN エリアは、この関数を SCAN_IoRefresh()の後で実行しないと、スレーブの最新の入力が反映されません。
- ・スレーブとの間で I/O コネクションを 1 つしか使用していない場合 (SCAN_RegisterSlaveDevice()を使用した場合など)、入力データ 2 格納バッファアドレスには NULL を指定してください。
- ・IO_DATA_CTL 構造体のデータサイズには、指定したノードアドレスの入力データサイズを指定してください。読み出された入力データは、IO_DATA_CTL 構造体のデータ格納バッファに格納されます。

SCAN_SetOutData() スレーブ 出力データ書き込み

使用範囲： **オープン状態**

機 能 指定されたスレーブの出力データを、本ボード内の OUT エリアに書き込みます。

呼び出し形式 `BOOL SCAN_SetOutData(DWORD Handle,
WORD MacId, IO_DATA_CTL *OutData1, IO_DATA_CTL *OutData2);`

引 数	型	変数名	内容
	DWORD	Handle	SCAN_Open()で得られたデバイスハンドル
	WORD	MacId	出力データを書き込むノードアドレス 指定範囲：0x0～0x3F (0～63)
	IO_DATA_CTL*	OutData1	出力データ 1 格納バッファアドレス
	IO_DATA_CTL*	OutData2	出力データ 2 格納バッファアドレス (未使用時は NULL)

戻り値 指定された出力データを正常に書き込んだ場合は TRUE を、エラーが発生した場合は FALSE を返します。詳細なエラー情報は GetLastError()で取得することができます。

説 明

- ・この関数の後で SCAN_IoRefresh()を実行しないと、本ボード内の OUT エリアが、スレーブの最新の出力に反映されません。
- ・スレーブとの間で I/O コネクションを 1 つしか使用していない場合
(SCAN_RegisterSlaveDevice()を使用した場合など)、出力データ 2 格納バッファアドレスには NULL を指定してください。
- ・IO_DATA_CTL 構造体のデータサイズには、指定したノードアドレスの出力データサイズを指定してください。また、スレーブへの出力データは、IO_DATA_CTL 構造体のデータ格納バッファに格納してください。

SCAN_SendMasterCosToSlave() COS 出力データ送信

使用範囲： **I/O通信実行状態**

機 能 COS コネクションで通信しているスレーブに出力データを送信します。

呼び出し形式 `BOOL SCAN_SendMasterCosToSlave(DWORD Handle, WORD MacId);`

引 数	型	変数名	内容
	DWORD	Handle	SCAN_Open()で得られたデバイスハンドル
	WORD	MacId	出力データを書き込むノードアドレス 指定範囲：0x0～0x3F (0～63)

戻り値 スレーブへの出力データ送信が正常に行われた場合は TRUE を、エラーが発生した場合は FALSE を返します。詳細なエラー情報は GetLastError()で取得することができます。

説 明

- ・出力データを送信する前に、SCAN_SetOutData()を使用して、COS コネクションにおけるスレーブへの出力データを設定しておく必要があります。
- ・COS 通信は即時通信を保証するものではありません。複数のノードで COS 通信を行っている場合、送信を待たされる可能性があります。

4-5 スレーブ機能用 API

■スレーブスキャンリスト操作 API

SELF_DEV 構造体

スレーブスキャンリストに I/O 情報を登録する際の、登録情報の形式を定義する構造体です。SCAN_RegisterSelfSlaveDevice(), SCAN_GetSelfSlaveDevice() 関数で使用されます。

型	変数名	内容												
WORD	ScanType	スキャンタイプ 使用する接続の種類を指定します。 最大 2 つの選択が可能です。ただし、COS、サイクリックの同時選択はできません。 自動選択時は、0x8000 (32768) を指定します。 <table><tr><td>Bit5</td><td>Bit4</td><td>Bit3</td><td>Bit2</td><td>Bit1</td><td>Bit0</td></tr><tr><td>CY</td><td>COS</td><td>予約</td><td>BS</td><td>Poll</td><td>予約</td></tr></table> BS : Bit-Strobe CY : サイクリック	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	CY	COS	予約	BS	Poll	予約
Bit5	Bit4	Bit3	Bit2	Bit1	Bit0									
CY	COS	予約	BS	Poll	予約									
WORD	Output1	出力データ 1 (マスター→スレーブ) サイズ (バイト単位) : 0x00～0xC8 (0～200)												
WORD	Input1	入力データ 1 (スレーブ→マスタ) サイズ (バイト単位) : 0x00～0xC8 (0～200)												
WORD	Output2	出力データ 2 (マスター→スレーブ) サイズ (バイト単位) : 0x00～0xC8 (0～200) スキャンタイプに自動選択を指定する場合、またはコネクションを 1 つだけ指定する場合は、このエリアは 0 としてください。												
WORD	Input2	入力データ 2 (スレーブ→マスタ) サイズ (バイト単位) : 0x00～0xC8 (0～200) スキャンタイプに自動選択を指定する場合、またはコネクションを 1 つだけ指定する場合は、このエリアは 0 としてください。												
WORD	ConnectAccept	接続/非接続 非接続時 : 0xFFFF (65535) 接続時 : 上記以外 通常は 0 (接続時) としてください。 スキャンリストへの登録は行っても、I/O 通信を行わない場合には、0xFFFF (非接続時) としてください。この場合、マスタでは「スレーブ不在」の異常が発生します。												
WORD	ErrorOutData	通信異常時出力データ保持/クリア 出力データ保持 : 0xFFFF (65535) クリア : 上記以外 マスタとの間で通信異常が発生したときの、出力データ (マスター→スレーブ) の状態を指定します。												

SCAN_RegisterSelfSlaveDevice() スレーブスキャンリスト登録

使用範囲： **オープン状態**

機 能 スレーブ機能の情報を、スレーブスキャンリストに登録します。

呼び出し形式 BOOL SCAN_RegisterSelfSlaveDevice(DWORD Handle, SELF_DEV* DeviceInfo);

引 数	型	変数名	内容
	DWORD	Handle	SCAN_Open()で得られたデバイスハンドル
	SELF_DEV*	DeviceInfo	スレーブ機能の情報を格納したバッファアドレス

戻 り 値 指定されたスレーブ情報をスレーブスキャンリストに正常に登録できた場合は TRUE を、エラーが発生した場合は FALSE を返します。詳細なエラー情報は GetLastError() で取得することができます。

説 明 ・接続指定の場合は、この関数終了後にマスタからのコネクション開設要求を受信すると、自動的に I/O 通信が開始されるようになります。

SCAN_RemoveSelfSlaveDevice() スレーブスキャンリスト削除

使用範囲： **オープン状態**

機 能 スレーブスキャンリスト情報を削除します。

呼び出し形式 BOOL SCAN_RemoveSelfSlaveDevice(DWORD Handle);

引 数	型	変数名	内容
	DWORD	Handle	SCAN_Open()で得られたデバイスハンドル

戻 り 値 スレーブスキャンリストを正常に削除できた場合は TRUE を、エラーが発生した場合は FALSE を返します。詳細なエラー情報は GetLastError() で取得することができます。

SCAN_GetSelfSlaveDevice() スレーブスキャンリスト読み出し

使用範囲： **オープン状態**

機 能 スレーブスキャンリスト情報を取得します。

呼び出し形式 BOOL SCAN_GetSelfSlaveDevice(DWORD Handle, SELF_DEV* DeviceInfo);

引 数	型	変数名	内容
	DWORD	Handle	SCAN_Open()で得られたデバイスハンドル
	SELF_DEV*	DeviceInfo	スレーブ機能の情報を受け取るためのバッファアドレス

戻 り 値 スレーブスキャンリストを正常に取得できた場合は TRUE を、エラーが発生した場合は FALSE を返します。詳細なエラー情報は GetLastError() で取得することができます。

説 明 ・この関数終了後、SELF_DEV 構造体にスレーブスキャンリストの情報が格納されます。

SCAN_StoreSlaveScanlist()

スレーブスキャンリスト ストア

使用範囲: **オープン状態****機 能**

スレーブスキャンリスト情報を不揮発メモリへ保存します。

呼び出し形式

BOOL SCAN_StoreSlaveScanlist(DWORD Handle);

引 数

型	変数名	内容
DWORD	Handle	SCAN_Open()で得られたデバイスハンドル

戻 り 値

スレーブスキャンリストを正常に不揮発メモリに書き込めた場合は TRUE を、エラーが発生した場合は FALSE を返します。詳細なエラー情報は GetLastError()で取得することができます。

説 明

- ・不揮発性メモリに保存したスレーブスキャンリスト情報は、SCAN_LoadSlaveScanlist()を使用することで、スレーブスキャンリストとして登録できます。

SCAN_LoadSlaveScanlist()

スレーブスキャンリスト ロード

使用範囲: **オープン状態****機 能**

スレーブスキャンリスト情報を不揮発メモリから読み込み、スレーブスキャンリストとして登録します。

呼び出し形式

BOOL SCAN_LoadSlaveScanlist(DWORD Handle);

引 数

型	変数名	内容
DWORD	Handle	SCAN_Open()で得られたデバイスハンドル

戻 り 値

スレーブスキャンリストを正常に不揮発メモリから読み込めた場合は TRUE を、エラーが発生した場合は FALSE を返します。詳細なエラー情報は GetLastError()で取得することができます。

説 明

- ・不揮発性メモリにスレーブスキャンリストを保存するには、SCAN_StoreSlaveScanlist()を使用します。

■ I/O 通信サービス API

SCAN_ConnectMasterDevice()

マスタ I/O 通信開始

使用範囲： **オンライン状態**

機 能

指定されたマスタとの I/O 通信を開始します。

呼び出し形式

BOOL SCAN_ConnectMasterDevice(DWORD Handle, WORD ErrorOutData);

引 数

型	変数名	内容
DWORD	Handle	SCAN_Open() で得られたデバイスハンドル
WORD	ErrorOutData	通信異常時出力データ保持／クリア 出力データ保持 : 0xFFFF (65535) クリア : 上記以外

戻 り 値

指定されたマスタとの I/O 通信が正常に開始できた場合は TRUE を、エラーが発生した場合は FALSE を返します。詳細なエラー情報は GetLastError() で取得することができます。

説 明

- ・停止中のマスタとの I/O 通信を開始します。
- ・この関数を使うためには、SCAN_RegisterSelfSlaveDevice() でスレーブとしての情報が登録されている必要があります。

SCAN_DisconnectMasterDevice()

マスタ I/O 通信停止

使用範囲： **I/O 通信実行状態**

機 能

指定されたマスタとの I/O 通信を停止します。

呼び出し形式

BOOL SCAN_DisconnectMasterDevice(DWORD Handle);

引 数

型	変数名	内容
DWORD	Handle	SCAN_Open() で得られたデバイスハンドル

戻 り 値

指定されたマスタとの I/O 通信が正常に停止できた場合は TRUE を、エラーが発生した場合は FALSE を返します。詳細なエラー情報は GetLastError() で取得することができます。

説 明

- ・マスタとの I/O 通信を停止します。このとき、マスタでは通信異常が発生します。
- ・停止した I/O 通信は、SCAN_ConnectMasterDevice() で再開することができます。

■ I/O データアクセスサービス API

SCAN_SlaveloRefresh()

スレーブ 入出力データ更新

使用範囲：オープン状態

機 能

本ボード内のスレーブ機能の IN エリア・OUT エリアを、実行中のリモート I/O 通信の最新の状態で更新します。スレーブ機能において、I/O エリアのデータを最新の状態に更新します。

呼び出し形式

BOOL SCAN_SlaveIoRefresh(DWORD Handle);

引 数

型	変数名	内容
DWORD	Handle	SCAN_Open() で得られたデバイスハンドル

戻 り 値

指定された本ボードのスレーブ機能 IN・OUT エリアの更新が正常に終了した場合は TRUE を、エラーが発生した場合は FALSE を返します。詳細なエラー情報は GetLastError() で取得することができます。

説 明

- ・この関数を使用することで、API 関数でアクセスできるスレーブ機能用 I/O エリアと、実際にボード内で通信に使われているスレーブ機能用 I/O エリアの間でのデータ交換が実行され、最新の I/O 状態となります。
- ・IN データは、この関数終了後に SCAN_GetSlaveOutData() で取得してください。
- ・OUT データについては、必要なスレーブに対して SCAN_SetSlaveInData() を実行した後で、この関数を実行してください。

IO_DATA_CTL 構造体

本ボード内の入出力エリアとのアクセスデータ形式を定義する構造体です。

SCAN_GetInData()、SCAN_SetOutData()、SCAN_GetSlaveOutData()、SCAN_SetSlaveInData() 関数で使用されます。

型	変数名	内容
WORD	DataSize	データサイズ (バイト単位) 指定範囲：0x0～0xC8 (0～200)
BYTE	Data[256]	データ格納バッファ

データサイズには、読み出しデータサイズ、または書き込みデータサイズを指定してください。

SCAN_GetSlaveOutData()

マスタ 出力データ読み出し

使用範囲： **オープン状態**

機 能

マスタからの出力データを、本ボード内のスレーブ機能用 OUT エリアより読み出します。

呼び出し形式

```
BOOL SCAN_GetSlaveOutData(DWORD Handle,  
                           IO_DATA_CTL *OutData1, IO_DATA_CTL *OutData2);
```

引 数

型	変数名	内容
DWORD	Handle	SCAN_Open() で得られたデバイスハンドル
IO_DATA_CTL*	OutData1	出力データ 1 格納バッファアドレス
IO_DATA_CTL*	OutData2	出力データ 2 格納バッファアドレス (未使用時は NULL)

戻 り 値

指定された出力データを正常に読み出した場合は TRUE を、エラーが発生した場合は FALSE を返します。詳細なエラー情報は GetLastError() で取得することができます。

説 明

- ・本ボード内のスレーブ機能用 OUT エリアは、この関数を SCAN_SlaveIoRefresh() の後で実行しないと、マスタからの最新の出力が反映されません。
- ・スレーブ機能で I/O コネクションを 1 つしか使用していない場合、出力データ 2 格納バッファアドレスには NULL を指定してください。
- ・IO_DATA_CTL 構造体のデータサイズには、スレーブ機能でのマスタからの出力サイズ (OUT サイズ) を指定してください。読み出されたマスタからの出力データは、IO_DATA_CTL 構造体のデータ格納バッファに格納されます。

SCAN_SetSlaveInData()

マスタ 入力データ書き込み

使用範囲： **オープン状態**

機 能

マスタへの入力データを、本ボード内のスレーブ機能用 IN エリアに書き込みます。

呼び出し形式

```
BOOL SCAN_SetSlaveInData(DWORD Handle,  
                          IO_DATA_CTL *InData1, IO_DATA_CTL *InData2);
```

引 数

型	変数名	内容
DWORD	Handle	SCAN_Open() で得られたデバイスハンドル
IO_DATA_CTL*	InData1	入力データ 1 格納バッファアドレス
IO_DATA_CTL*	InData2	入力データ 2 格納バッファアドレス (未使用時は NULL)

戻 り 値

指定された入力データを正常に書き込んだ場合は TRUE を、エラーが発生した場合は FALSE を返します。詳細なエラー情報は GetLastError() で取得することができます。

説 明

- ・この関数の後で SCAN_SlaveIoRefresh() を実行しないと、本ボード内のスレーブ機能用 IN エリアが、マスタでの最新の入力に反映されません。
- ・スレーブ機能で I/O コネクションを 1 つしか使用していない場合、入力データ 2 格納バッファアドレスには NULL を指定してください。
- ・IO_DATA_CTL 構造体のデータサイズには、スレーブ機能でのマスタへの入力サイズ (IN サイズ) を指定してください。また、マスタへの入力データは、IO_DATA_CTL 構造体のデータ格納バッファに格納してください。

SCAN_SendSlaveCosToMaster()

COS 入力データ送信

使用範囲： I/O通信実行状態

機 能

COS コネクションで通信しているマスタに入力データを送信します。

呼び出し形式

BOOL SCAN_SendSlaveCosToMaster(DWORD Handle);

引 数

型	変数名	内容
DWORD	Handle	SCAN_Open()で得られたデバイスハンドル

戻 り 値

マスタへの入力データ送信が正常に行われた場合は TRUE を、エラーが発生した場合は FALSE を返します。詳細なエラー情報は GetLastError()で取得することができます。

説 明

- ・入力データを送信する前に、SCAN_SetSlaveInData()を使用して、COS コネクションにおけるマスタへの入力データを設定しておく必要があります。
- ・COS 通信は即時通信を保証するものではありません。複数のノードで COS 通信を行っている場合、送信を待たされる可能性があります。

4-6 Explicit メッセージ用 API

■メッセージ監視タイマサービス API

SCAN_SetMessageTimerValue()

メッセージ監視タイマ登録

使用範囲： **オープン状態**

機 能 指定ノードアドレスとの Explicit クライアント接続のメッセージ監視タイマ値を登録します。

呼び出し形式 BOOL SCAN_SetMessageTimerValue(DWORD Handle, WORD MacId, WORD Epr);

引 数	型	変数名	内容
	DWORD	Handle	SCAN_Open()で得られたデバイスハンドル
	WORD	MacId	対象ノードアドレス 指定範囲：0x0～0x3F（0～63）
	WORD	Epr	登録するメッセージ監視タイマ値（msec） 指定範囲：0x1F4～0x7530（500～30000）

戻 り 値 メッセージ監視タイマ値の登録が正しく行われた場合は TRUE を、エラーが発生した場合は FALSE を返します。詳細なエラー情報は GetLastError() で取得することができます。

説 明

- ・メッセージ監視タイマ値を変更する場合に使用してください。
- ・メッセージ監視タイマ値を 0 で指定した場合は、デフォルト値（2 秒＝2000msec）が使用されます。

SCAN_GetMessageTimerValue()

メッセージ監視タイマ読み出し

使用範囲： **オープン状態**

機 能 指定ノードアドレスとの Explicit クライアント接続のメッセージ監視タイマ値を取得します。

呼び出し形式 BOOL SCAN_GetMessageTimerValue(DWORD Handle, WORD MacId, WORD *Epr);

引 数	型	変数名	内容
	DWORD	Handle	SCAN_Open()で得られたデバイスハンドル
	WORD	MacId	対象ノードアドレス 指定範囲：0x0～0x3F（0～63）
	WORD *	Epr	メッセージ監視タイマ値（msec）を受け取るためのバッファアドレス

戻 り 値 メッセージ監視タイマ値の取得が正しく行われた場合は TRUE を、エラーが発生した場合は FALSE を返します。詳細なエラー情報は GetLastError() で取得することができます。

説 明

- ・現在のメッセージ監視タイマ値を確認する場合などに使用してください。
- ・SCAN_SetMessageTimerValue() で設定されていない場合は、メッセージ監視タイマ値はデフォルト値の 0（2 秒）が読み出されます。

SCAN_StoreMessageTimerValueList()

メッセージ監視タイマリストストア

使用範囲： **オープン状態**

機能

Explicit クライアント接続のメッセージ監視タイマリストを不揮発メモリに保存します。

呼び出し形式

BOOL SCAN_StoreMessageTimerValueList(DWORD Handle);

引数

型	変数名	内容
DWORD	Handle	SCAN_Open()で得られたデバイスハンドル

戻り値

不揮発メモリへの保存が正しく行われた場合は TRUE を、エラーが発生した場合は FALSE を返します。詳細なエラー情報は GetLastError()で取得することができます。

説明

- 全ノードアドレス分のメッセージ監視タイマ値が保存されます。
- 不揮発性メモリに保存されたメッセージ監視タイマリストは、SCAN_LoadMessageTimerValueList()を使用することで、有効になります。

SCAN_LoadMessageTimerValueList()

メッセージ監視タイマリストロード

使用範囲： **オープン状態**

機能

Explicit クライアント接続の監視タイマリストを不揮発メモリから読み込み、メッセージ監視タイマとして登録します。

呼び出し形式

BOOL SCAN_LoadMessageTimerValueList(DWORD Handle);

引数

型	変数名	内容
DWORD	Handle	SCAN_Open()で得られたデバイスハンドル

戻り値

不揮発メモリから読み込みが正しく行われた場合は TRUE を、エラーが発生した場合は FALSE を返します。詳細なエラー情報は GetLastError()で取得することができます。

説明

- 全ノードアドレス分のメッセージ監視タイマ値が、保存されている値で更新されます。
- この関数を、一度も SCAN_StoreMessageTimerValueList()を実行しないで使用した場合は、全ノードアドレス分のメッセージ監視タイマ値はデフォルト値の 0（2 秒）となります。

■クライアントメッセージサービス API

SCAN_RegClientEvtNotifyMessage() クライアントレスポンス通知メッセージ登録

使用範囲： **オープン状態**

機 能

指定ノードアドレスからのクライアントレスポンス受信イベントが発生した場合に通知する、Windows メッセージを登録します。

呼び出し形式

BOOL SCAN_RegClientEvtNotifyMessage
(DWORD Handle, WORD MacId, DWORD ThreadId, HWND hWnd, UINT Msg);

引 数

型	変数名	内容
DWORD	Handle	SCAN_Open()で得られたデバイスハンドル
WORD	MacId	イベントをチェックするノードアドレス 指定範囲：0x0～0x3F (0～63)
DWORD	ThreadId	通知するスレッド ID (未設定：NULL)
HWND	hWnd	通知するウィンドウハンドル (未設定：NULL)
UINT	Msg	通知メッセージ (イベント ID) 指定範囲：WM_USER+0x100～WM_USER+0x7FFF

戻 り 値

メッセージ登録が正しく行われた場合は TRUE を、スレッド ID およびウィンドウハンドル両方が NULL の場合、エラーが発生した場合は FALSE を返します。詳細なエラー情報は GetLastError() で取得することができます。

説 明

・通知メッセージと共に WPARAM にクライアントレスポンス送信元 (相手) ノードアドレスが、LPARAM に受信レスポンスサイズ (バイト単位) が通知されます。

SCAN_UnRegClientEvtNotifyMessage() クライアントレスポンス通知メッセージ解除

使用範囲： **オープン状態**

機 能

指定ノードアドレスの通知メッセージ登録を解除します。

呼び出し形式

BOOL SCAN_UnRegClientEvtNotifyMessage(DWORD Handle, WORD MacId);

引 数

型	変数名	内容
DWORD	Handle	SCAN_Open()で得られたデバイスハンドル
WORD	MacId	解除ノードアドレス

戻 り 値

通知メッセージの登録解除が正しく行われた場合は TRUE を、エラーが発生した場合は FALSE を返します。詳細なエラー情報は GetLastError() で取得することができます。

SCAN_PeekClientEvent() クライアントレスポンス イベントチェック

使用範囲： オンライン状態

機 能

クライアントレスポンスイベントがあるかチェックします。

呼び出し形式

BOOL SCAN_PeekClientEvent(DWORD Handle, WORD MacId,);

引 数

型	変数名	内容
DWORD	Handle	SCAN_Open()で得られたデバイスハンドル
WORD	MacId	イベントをチェックするレスポンス送信元（相手）ノードアドレス 指定範囲：0x00～0x3F（0～63）

戻 り 値

該当イベントが存在する場合は TRUE を、該当イベントなしなどエラーが発生した場合は FALSE を返します。詳細なエラー情報は GetLastError()で取得することができます。

SCAN_GetClientEventLength() クライアントレスポンス サイズ長読み出し

使用範囲： オンライン状態

機 能

クライアントレスポンスのサイズ（バイト単位）を取得します。

呼び出し形式

BOOL SCAN_GetClientEventLength(DWORD Handle, WORD MacId, DWORD *Length);

引 数

型	変数名	内容
DWORD	Handle	SCAN_Open()で得られたデバイスハンドル
WORD	MacId	レスポンスサイズを読み出すレスポンスの送信元（相手）ノードアドレス
DWORD *	Length	レスポンスサイズを受け取るためのバッファアドレス

戻 り 値

該当イベントのサイズが取得できた場合は TRUE を、該当イベントなしなどエラーが発生した場合は FALSE を返します。詳細なエラー情報は GetLastError()で取得することができます。

説 明

・SCAN_ReceiveClientExplicit()でレスポンスメッセージを取り出す前に、この関数で読み出したレスポンスサイズ以上のサービスデータ格納バッファを用意してください。

CLIENT_REQ 構造体

クライアントリクエスト Explicit メッセージの形式を定義する構造体です。
SCAN_SendClientExplicit()関数で使用されます。

型	変数名	内容
DWORD	MessageID	メッセージ ID アプリケーションでメッセージを識別するために任意の値を設定してください。ただし、特に識別する必要がない場合は 0 としてください。
WORD	MacId	送信先（相手）ノードアドレス
WORD	ServiceCode	サービスコード
WORD	ClassID	クラス ID
WORD	InstanceID	インスタンス ID
WORD	DataLength	サービスデータのデータサイズ（バイト単位） 指定範囲：0x0～0x228（0～552）
BYTE *	ServiceData	サービスデータ格納バッファアドレス バッファには DataLength で指定したバイト数分のサービスデータを用意しておきます。

SCAN_SendClientExplicit()

クライアント Explicit メッセージ送信

使用範囲： **オンライン状態**

機 能

クライアントリクエストメッセージを送信します。

呼び出し形式

BOOL SCAN_SendClientExplicit(DWORD Handle, CLIENT_REQ *Msg);

引 数

型	変数名	内容
DWORD	Handle	SCAN_Open()で得られたデバイスハンドル
CLIENT_REQ *	Msg	送信するリクエストメッセージ格納バッファアドレス

戻 り 値

指定したノードアドレスへの送信イベント登録が正常に終了した場合は TRUE を、エラーが発生した場合は FALSE を返します。詳細なエラー情報は GetLastError()で取得することができます。

説 明

- ・ CLIENT_REQ 構造体には、メッセージ ID や、送信先ノードアドレス、サービスデータのサイズ、リクエストとして送信するサービスに対応したデータ（ServiceCode、ClassID、InstanceID、ServiceData）を格納しておく必要があります。
- ・ CLIENT_REQ 構造体の MessageID には、アプリケーションでメッセージを識別する必要がある場合は、任意の値を設定してください。必要がない場合は、0 としてください。設定したメッセージ ID は、SCAN_ReceiveClientExplicit()終了後に、CLIENT_RES 構造体の MessageID に格納されます。

注：送信先のスレーブのタイプによって、メッセージを受信しそこねる場合がありますので、Explicit メッセージを送信する場合、必ずリトライ処理をするようにしてください。

CLIENT_RES 構造体

クライアントレスポンス Explicit メッセージの形式を定義する構造体です。
SCAN_ReceiveClientExplicit()関数で使用されます。

型	変数名	内容
DWORD	MessageID	メッセージ ID 関数終了時には、リクエスト送信時に設定したメッセージ ID が格納されます。
WORD	MacId	送信元（相手）ノードアドレス 関数呼び出し前に、取得するクライアントレスポンスの相手ノードアドレスを指定します。
WORD	ServiceCode	サービスコード 関数終了時には、サービスコードが格納されます。
WORD	DataLength	サービスデータ格納バッファサイズ（バイト単位） 指定範囲：0x0～0x228（0～552） 関数呼び出し前に、サービスデータ格納バッファとして確保したサイズを指定します。 クライアントレスポンス通信メッセージまたは SCAN_GetClientEventLength()で取得した受信レスポンスサイズ分のバッファを、サービスデータ格納バッファとして確保してください。 関数終了時には、実際に格納されたサービスデータのバイト数がセットされます。
BYTE *	ServiceData	サービスデータ格納バッファアドレス 関数呼び出し前に、DataLength で指定したバイト数分のバッファを確保しておく必要があります。 関数終了時には、ここで指定したアドレスのバッファに、受信したサービスデータが格納されます。

SCAN_ReceiveClientExplicit() クライアント Explicit メッセージ読み出し

使用範囲： オンライン状態

機 能

受信キューからクライアントレスポンスメッセージを取得します。

呼び出し形式

BOOL SCAN_ReceiveClientExplicit(DWORD Handle, CLIENT_RES *Msg);

引 数

型	変数名	内容
DWORD	Handle	SCAN_Open()で得られたデバイスハンドル
CLIENT_RES *	Msg	レスポンスメッセージ格納バッファアドレス

戻 り 値

指定したノードアドレスから、レスポンスメッセージが正常に取得できた場合は TRUE を、エラーが発生した場合は FALSE を返します。詳細なエラー情報は GetLastError()で取得することができます。

説 明

- ・ CLIENT_RES 構造体の送信元（相手）ノードアドレス、サービスデータ格納バッファサイズ、サービスデータ格納バッファアドレスは、この関数呼び出し時にセットしておく必要があります。
- ・ 通知メッセージあるいは SCAN_GetClientEventLength()で得られる受信レスポンスサイズをもとに、レスポンスメッセージ格納バッファは、十分なサイズを確保してください。
- ・ この関数が正常に終了すると、CLIENT_RES 構造体に、メッセージ ID、サービスコード、サービスデータ格納バッファサイズが格納され、指定したサービスデータ格納バッファに、受信したサービスデータ（レスポンス）が格納されます。

■サーバメッセージサービス API

SCAN_RegObjectClass()

オブジェクトクラス ID 登録

使用範囲： **オープン状態**

機 能

アプリケーションプログラムで実装する、オブジェクトクラスを登録します。

呼び出し形式

BOOL SCAN_RegObjectClass(DWORD Handle, WORD ClassId);

引 数

型	変数名	内容
DWORD	Handle	SCAN_Open() で得られたデバイスハンドル
WORD	ClassId	登録するオブジェクトクラス ID

戻 り 値

指定された本ボードへのオブジェクトクラス ID 登録が正常に終了した場合は TRUE を、エラーが発生した場合は FALSE を返します。詳細なエラー情報は GetLastError() で取得することができます。

説 明

- ・オブジェクトを登録する場合は、そのオブジェクト宛のサーバリクエストメッセージを受信し、レスポンスを送信する必要があります。
- ・登録可能なオブジェクトクラス ID は以下の範囲の値です。また最大登録個数は 16 個です。
0x0000～0x0063 (0x0002, 0x0003, 0x0005, 0x002B, 0x002F を除く)
0x0064～0x007F, 0x0088～0x008F, 0x0098～0x009F,
0x00A8～0x00AF, 0x00B8～0x00BF, 0x00C0～0x00C7
- ・Identity Object Class (ID=1) を登録する場合、サービスコードが Reset の処理を必ずサポートしてください。

SCAN_UnRegObjectClass()

オブジェクトクラス ID 解除

使用範囲： **オープン状態**

機 能

登録されている、オブジェクトクラスを削除します。

呼び出し形式

BOOL SCAN_UnRegObjectClass(DWORD Handle, WORD ClassId);

引 数

型	変数名	内容
DWORD	Handle	SCAN_Open() で得られたデバイスハンドル
WORD	ClassId	削除するオブジェクトクラス ID

戻 り 値

指定した本ボードから、オブジェクトクラス ID 削除が正常に終了した場合は TRUE を、エラーが発生した場合は FALSE を返します。詳細なエラー情報は GetLastError() で取得することができます。

SCAN_RegServerEvtNotifyMessage()

サーバ通知メッセージ登録

使用範囲： **オープン状態**

機 能	登録したオブジェクト宛のサーバリクエスト受信イベントが発生した場合に通知する、Windows メッセージを登録します。																			
呼び出し形式	BOOL SCAN_RegServerEvtNotifyMessage (DWORD Handle, WORD ClassId, DWORD ThreadId, HWND hWnd, UINT Msg);																			
引 数	<table border="1"> <thead> <tr> <th>型</th><th>変数名</th><th>内容</th></tr> </thead> <tbody> <tr> <td>DWORD</td><td>Handle</td><td>SCAN_Open() で得られたデバイスハンドル</td></tr> <tr> <td>WORD</td><td>ClassId</td><td>サーバリクエスト受信イベントに使用するオブジェクトクラス ID</td></tr> <tr> <td>DWORD</td><td>ThreadId</td><td>通知するスレッド ID (未設定: NULL)</td></tr> <tr> <td>HWND</td><td>hWnd</td><td>通知するウィンドウハンドル (未設定: NULL)</td></tr> <tr> <td>UINT</td><td>Msg</td><td>通知メッセージ (イベント ID) 指定範囲: WM_USER+0x100~WM_USER+0x7FFF</td></tr> </tbody> </table>		型	変数名	内容	DWORD	Handle	SCAN_Open() で得られたデバイスハンドル	WORD	ClassId	サーバリクエスト受信イベントに使用するオブジェクトクラス ID	DWORD	ThreadId	通知するスレッド ID (未設定: NULL)	HWND	hWnd	通知するウィンドウハンドル (未設定: NULL)	UINT	Msg	通知メッセージ (イベント ID) 指定範囲: WM_USER+0x100~WM_USER+0x7FFF
型	変数名	内容																		
DWORD	Handle	SCAN_Open() で得られたデバイスハンドル																		
WORD	ClassId	サーバリクエスト受信イベントに使用するオブジェクトクラス ID																		
DWORD	ThreadId	通知するスレッド ID (未設定: NULL)																		
HWND	hWnd	通知するウィンドウハンドル (未設定: NULL)																		
UINT	Msg	通知メッセージ (イベント ID) 指定範囲: WM_USER+0x100~WM_USER+0x7FFF																		
戻 り 値	メッセージ登録が正しく行われた場合は TRUE を、スレッド ID およびウィンドウハンドル両方が NULL の場合など、エラーが発生した場合は FALSE を返します。詳細なエラー情報は GetLastError() で取得することができます。																			
説 明	・通知メッセージと共に WPARAM にサーバリクエストの宛先オブジェクトクラス ID が、LPARAM に受信リクエストサイズ (バイト単位) が通知されます。																			

SCAN_UnRegServerEvtNotifyMessage()

サーバ通知メッセージ解除

使用範囲： **オープン状態**

機 能	指定オブジェクトクラス ID の通知メッセージ登録を解除します。										
呼び出し形式	BOOL SCAN_UnRegServerEvtNotifyMessage(DWORD Handle, WORD ClassId);										
引 数	<table border="1"> <thead> <tr> <th>型</th><th>変数名</th><th>内容</th></tr> </thead> <tbody> <tr> <td>DWORD</td><td>Handle</td><td>SCAN_Open() で得られたデバイスハンドル</td></tr> <tr> <td>WORD</td><td>ClassId</td><td>解除するオブジェクトクラス ID</td></tr> </tbody> </table>		型	変数名	内容	DWORD	Handle	SCAN_Open() で得られたデバイスハンドル	WORD	ClassId	解除するオブジェクトクラス ID
型	変数名	内容									
DWORD	Handle	SCAN_Open() で得られたデバイスハンドル									
WORD	ClassId	解除するオブジェクトクラス ID									
戻 り 値	通知メッセージの登録解除が正しく行われた場合は TRUE を、エラーが発生した場合は FALSE を返します。詳細なエラー情報は GetLastError() で取得することができます。										

SCAN_PeekServerEvent()

サーバリクエスト イベントチェック

使用範囲： **オープン状態**

機 能

指定オブジェクトクラス ID 宛のサーバリクエストイベントがあるかチェックします。

呼び出し形式

BOOL SCAN_PeekServerEvent(DWORD Handle, WORD ClassId);

引 数

型	変数名	内容
DWORD	Handle	SCAN_Open() で得られたデバイスハンドル
WORD	ClassId	イベントをチェックするクラス ID

戻 り 値

該当イベントが存在する場合は TRUE を、該当イベントなしなどエラーが発生した場合は FALSE を返します。詳細なエラー情報は GetLastError() で取得することができます。

4

SCAN_GetServerEventLength()

サーバリクエスト サイズ読み出し

使用範囲： **オープン状態**

機 能

指定オブジェクトクラス ID 宛のサーバリクエストのサイズ（バイト単位）を取得します。

呼び出し形式

BOOL SCAN_GetServerEventLength(DWORD Handle, WORD ClassId, DWORD *Length);

引 数

型	変数名	内容
DWORD	Handle	SCAN_Open() で得られたデバイスハンドル
WORD	ClassId	リクエストサイズを読み出すリクエストの宛先クラス ID
DWORD *	Length	リクエストサイズを受け取るためのバッファアドレス

戻 り 値

該当イベントのサイズが取得できた場合は TRUE を、該当イベントなしなどエラーが発生した場合は FALSE を返します。詳細なエラー情報は GetLastError() で取得することができます。

説 明

- ・SCAN_ReceiveServerExplicit() でリクエストメッセージを取り出す前に、この関数で取得したリクエストサイズ以上のサービスデータ格納バッファを用意してください。

SERVER_REQ 構造体

サーバリクエスト Explicit メッセージの形式を定義する構造体です。
SCAN_ReceiveServerExplicit()関数で使われます。

型	変数名	内容
DWORD	MessageID	メッセージ ID 関数終了時には、サーバリクエストのメッセージ ID が格納されます。
WORD	ServiceCode	サービスコード 関数終了時には、サーバリクエストのサービスコードが格納されます。
WORD	ClassID	クラス ID 関数呼び出し前に、取得するサーバリクエストのクラス ID を指定します。
WORD	InstanceID	インスタンス ID 関数終了時には、サーバリクエストのインスタンス ID が格納されます。
WORD	DataLength	サービスデータ格納バッファサイズ (バイト単位) 指定範囲: 0x0~0x228 (0~552) 関数呼び出し前に、サービスデータ格納バッファとして確保したサイズを指定します。 サーバ通知メッセージまたは SCAN_GetServerEventLength で取得した受信リクエストサイズ分のバッファを、サービスデータ格納バッファとして確保してください。 関数終了時には、実際に格納されたサービスデータのバイト数がセットされます。
BYTE *	ServiceData	サービスデータ格納バッファアドレス DataLength で指定したバイト数分のサービスデータを用意しておく必要があります。 関数終了時には、ここで指定したアドレスのバッファに受信したサービスデータが格納されます。

SCAN_ReceiveServerExplicit()

サーバ Explicit メッセージ読み出し

使用範囲： オンライン状態

機能

受信キューからサーバーリクエストメッセージを取得します。

呼び出し形式

```
BOOL SCAN_ReceiveServerExplicit(DWORD Handle, SERVER_REQ *Msg);
```

引 数

型	変数名	内容
DWORD	Handle	SCAN_Open()で得られたデバイスハンドル
SERVER_REQ *	Msg	リクエストメッセージ格納バッファアドレス

戻り値

指定した本ボードより、リクエストメッセージが正常に取得できた場合は TRUE を、エラーが発生した場合は FALSE を返します。詳細なエラー情報は GetLastError()で取得することができます。

說明

- ・SERVER_REQ 構造体のクラス ID、サービスデータ格納バッファサイズ、サービスデータ格納バッファアドレスは、この関数の呼び出し前にセットしておく必要があります。
- ・通知メッセージあるいは SCAN_GetServerEventLength() で得られるイベントデータ長をもとに、リクエストメッセージ格納バッファは、十分なサイズを確保してください。
- ・この関数が正常に終了すると、SERVER_REQ 構造体には、本ポードによって自動的に決定されたメッセージ ID と、クライアント側から送信されたサービスコード、クラス ID、インスタンス ID、サービスデータサイズが格納され、指定したサービスデータ格納バッファには、受信したサービスデータ（リクエスト）が格納されます。
- ・取得したメッセージ ID は、サーバリクエストに対応したサーバレスポンスを送信するときに、SERVER_RES 構造体のメッセージ ID として指定してください。

SERVER_RES 構造体

サーバレスポンス Explicit メッセージの形式を定義する構造体です。
SCAN_SendServerExplicit()関数で使用されます。

型	変数名	内容
DWORD	MessageID	メッセージ ID SCAN_ReceiveServerExplicit で得られたリクエストのメッセージ ID を指定します。
WORD	ServiceCode	サービスコード レスポンスとして送信するサービスコードを指定します。 正常レスポンスの場合： SCAN_ReceiveServerExplicit で得られたサービスコードの最上位ビット（ビット 15）を ON にして、レスポンスのサービスコードとします。 異常レスポンスの場合： 0x94 (148) を異常レスポンスのサービスコードに指定します。
WORD	DataLength	サービスデータのデータサイズ（バイト単位） 指定範囲：0x0～0x228（0～552）
BYTE *	ServiceData	サービスデータ格納バッファアドレス DataLength で指定したバイト数分のサービスデータを用意しておきます。

SCAN_SendServerExplicit()

サーバ Explicit メッセージ送信

使用範囲： **オンライン状態**

機 能

サーバレスポンスメッセージを送信します。

呼び出し形式

BOOL SCAN_SendServerExplicit(DWORD Handle, SERVER_RES *Msg);

引 数

型	変数名	内容
DWORD	Handle	SCAN_Open()で得られたデバイスハンドル
SERVER_RES *	Msg	送信するレスポンスメッセージ格納バッファアドレス

戻 り 値

指定した本ボードへの送信イベント登録が正常に終了した場合は TRUE を、エラーが発生した場合は FALSE を返します。詳細なエラー情報は GetLastError()で取得することができます。

説 明

- ・ SERVER_RES 構造体には、メッセージ ID、サービスコード、サービスデータサイズ、サービスデータ格納バッファアドレスを格納し、指定したサービスデータ格納バッファには、レスポンスとして送信するサービスに対応したデータ（サービスデータ）を格納しておく必要があります。
- ・ SERVER_RES 構造体のメッセージ ID には、必ず SCAN_ReceiveServerExplicit()終了後に、SERVER_REQ 構造体の MessageID に格納されている値を指定してください。

4-7 保守用 API

■ステータスサービス API

SCAN_GetNetworkStatus()

ネットワークステータス読み出し

使用範囲： オープン状態

機能

ネットワーク（CAN）ステータスを取得します。

呼び出し形式

BOOL SCAN_GetNetworkStatus(DWORD Handle, WORD *Status);

引数	型	変数名	内容
	DWORD	Handle	SCAN_Open()で得られたデバイスハンドル
	WORD *	Status	ステータスを受け取るためのバッファアドレス

戻り値

指定された本ボードのネットワークステータスが正常に取得できた場合は TRUE を、エラーが発生した場合は FALSE を返します。詳細なエラー情報は GetLastError() で取得することができます。

説明

・ネットワークステータスは、下図のように 16 ビットで構成されています。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
—	B14	—	B12	B11	B10	B09	B08	—	—	—	—	—	—	B01	B00

各ビットの意味は、次のようになっています。

ビット	記号	意味
0	B00	CAN アクティブ
1	B01	ネットワークフレーム検知
8	B08	ネットワーク電源異常
9	B09	送信タイムアウト異常
10	B10	受信オーバーライト発生
11	B11	受信オーバーロードワーニング
12	B12	送信オーバーロードワーニング
14	B14	バスオフ異常

・この関数で取得するステータスは制御に必要なものではありませんが、通信上の問題が発生したときに、ネットワークに起因する問題かどうかを切り分けるための参考にすることができます。

機能

スキャナステータスを取得します。

呼び出し形式

BOOL SCAN_GetScannerStatus(DWORD Handle, DWORD *Status);

引数

型	変数名	内容
DWORD	Handle	SCAN_Open()で得られたデバイスハンドル
DWORD *	Status	ステータスを受け取るためのバッファアドレス

戻り値

指定された本ボードのスキャナステータスが正常に取得できた場合は TRUE を、エラーが発生した場合は FALSE を返します。詳細なエラー情報は GetLastError() で取得することができます。

説明

- ・スキャナステータスは、下図のように 32 ビットで構成されています。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
—	—	B13	—	—	—	—	B08	B07	B06	B05	B04	B03	—	B01	B00
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
B31	—	—	—	—	—	—	—	—	B22	B21	—	—	—	B17	B16

各ビットの意味は、次のようになっています。

ビット	記号	意味
0	B00	スキャナ機能異常発生中
1	B01	マスタ機能異常発生中
3	B03	スレーブ機能異常発生中
4	B04	メモリ異常
5	B05	バスオフ異常
6	B06	MAC ID 重複異常
7	B07	ネットワーク電源異常
8	B08	送信タイムアウト異常
13	B13	メッセージ監視タイマ不正
16	B16	オンライン
17	B17	スキャン (I/O 通信実行)
21	B21	スレーブ機能有効モード
22	B22	スレーブスキャンタイプ自動設定
31	B31	異常履歴あり

- ・マスタ機能またはスレーブ機能で I/O 通信を行う場合、スキャナステータスのビット 0 で、異常発生を監視してください。
- ・異常を通知するビットは、ビット 0～13 です。ビット 0 は、ビット 1～13 の OR となっています。
- ・ビット 1 は、マスタ機能で異常が発生したとき ON になります。このビットは、SCAN_GetMasterModeStatus() で取得するマスタ機能ステータスのビット 0、2 の OR となっています。
- ・ビット 3 は、スレーブ機能で異常が発生したとき ON になります。このビットは、SCAN_GetSlaveModeStatus() で取得するスレーブ機能ステータスのビット 2、3 の OR となっています。

SCAN_GetMasterModeStatus()

マスタ機能ステータス読み出し

使用範囲：オープン状態

機能

マスタ機能ステータスを取得します。

呼び出し形式

BOOL SCAN_GetMasterModeStatus(DWORD Handle, WORD *Status);

引数

型	変数名	内容
DWORD	Handle	SCAN_Open()で得られたデバイスハンドル
WORD *	Status	ステータスを受け取るためのバッファアドレス

戻り値

指定された本ボードのマスタ機能ステータスが正常に取得できた場合は TRUE を、エラーが発生した場合は FALSE を返します。詳細なエラー情報は GetLastError()で取得することができます。

説明

- マスタ機能ステータスは、下図のように 16 ビットで構成されています。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
B15	—	—	—	—	—	—	—	—	—	—	—	—	B02	—	B00

各ビットの意味は、次のようになっています。

ビット	記号	意味
0	B00	照合異常
2	B02	I/O 通信異常
15	B15	I/O 通信中

- ビット 15 は、マスタとしての I/O 通信が開始していることを表します。各スレーブの I/O データ入出力処理は、SCAN_GetSlaveDeviceStatus()により、スレーブの状態を個別に確認してから実行してください。
- 照合異常の詳細は、SCAN_GetSlaveDeviceStatus()により、スレーブごとに個別に確認してください。
- ビット 0、2 は、マスタ機能での異常を表します。どちらか一方でも ON になると、SCAN_GetScannerStatus()で取得するスキャナステータスのビット 1 が ON になります。

機 能

指定されたスレーブの詳細なステータスを取得します。

呼び出し形式

BOOL SCAN_GetSlaveDeviceStatus(DWORD Handle, WORD MacId, WORD *Status);

引 数

型	変数名	内容
DWORD	Handle	SCAN_Open()で得られたデバイスハンドル
WORD	MacId	スレーブ詳細ステータスを読み出すノードアドレス 指定範囲：0x0～0x3F (0～63)
WORD *	Status	ステータスを受け取るためのバッファアドレス

戻 り 値

指定されたスレーブのデバイスステータスが正常に取得できた場合は TRUE を、エラーが発生した場合は FALSE を返します。詳細なエラー情報は GetLastError()で取得することができます。

説 明

- ・スレーブのデバイスステータスは、下図のように 16 ビットで構成されています。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
B15	—	—	—	—	B10	—	B08	B07	B06	B05	B04	B03	B02	B01	B00

各ビットの意味は、次のようになっています。

ビット	記号	意味
0	B00	異常発生中
1	B01	照合異常
2	B02	スレーブ不在
3	B03	ベンダ ID 不一致
4	B04	プロダクトタイプ不一致
5	B05	プロダクトコード不一致
6	B06	コネクション未サポート
7	B07	I/O サイズ不一致
8	B08	コネクションパス不一致
10	B10	I/O 通信異常
15	B15	I/O 通信中

- ・ビット 0 は、ビット 1、10 の OR となっています。
- ・ビット 1 は、ビット 2～8 の OR となっています。照合異常の詳細が、ビット 2～8 に示されます。
- ・マスタ機能で各スレーブの I/O 入出力処理を行う前に、ビット 15 が ON であることを確認してください。

SCAN_IsScanlistSlaveDeviceRegist()

スレーブ登録チェック

使用範囲： **オープン状態**

機能

スキャンリストへのスレーブの登録の有無をチェックします。

呼び出し形式

BOOL SCAN_IsScanlistSlaveDeviceRegist(DWORD Handle, WORD MacId);

引数

型	変数名	内容
DWORD	Handle	SCAN_Open()で得られたデバイスハンドル
WORD	MacId	登録の有無をチェックするノードアドレス 指定範囲：0x0～0x3F (0～63)

戻り値

指定されたスレーブがスキャンリストに登録されている場合は TRUE を、エラーあるいは登録されていない場合は FALSE を返します。詳細なエラー情報は GetLastError()で取得することができます。

SCAN_IsDeviceConnection()

スレーブコネクション開設チェック

使用範囲： **オープン状態**

機能

スレーブとのコネクションが開設しているかをチェックします。

呼び出し形式

BOOL SCAN_IsDeviceConnection(DWORD Handle, WORD MacId);

引数

型	変数名	内容
DWORD	Handle	SCAN_Open()で得られたデバイスハンドル
WORD	MacId	コネクションの開設をチェックするノードアドレス 指定範囲：0x0～0x3F (0～63)

戻り値

指定されたデバイスと正常にコネクションが開設されている場合は TRUE を、エラーあるいはコネクションが開設されていない場合は FALSE を返します。詳細なエラー情報は GetLastError()で取得することができます。

SCAN_GetCycleTime()

通信サイクル時間現在値読み出し

使用範囲： **オープン状態**

機 能

通信サイクル時間の現在値を取得します。

呼び出し形式

BOOL SCAN_GetCycleTime(DWORD Handle, WORD *CycleTime);

引 数

型	変数名	内容
DWORD	Handle	SCAN_Open()で得られたデバイスハンドル
WORD *	CycleTime	通信サイクル時間現在値を受け取るためのバッファアドレス

戻 り 値

指定された本ボードから、通信サイクル時間現在値が正常に取得できた場合は TRUE を、エラーが発生した場合は FALSE を返します。詳細なエラー情報は GetLastError() で取得することができます。

説 明

- ・通信サイクル時間現在値は、ms 単位の値で格納されます。

SCAN_GetMaxCycleTime()

通信サイクル時間最大値読み出し

使用範囲： **オープン状態**

機 能

通信サイクル時間の最大値を取得します。

呼び出し形式

BOOL SCAN_GetMaxCycleTime(DWORD Handle, WORD *CycleTime);

引 数

型	変数名	内容
DWORD	Handle	SCAN_Open()で得られたデバイスハンドル
WORD *	CycleTime	通信サイクル時間最大値を受け取るためのバッファアドレス

戻 り 値

指定された本ボードから、通信サイクル時間最大値が正常に取得できた場合は TRUE を、エラーが発生した場合は FALSE を返します。詳細なエラー情報は GetLastError() で取得することができます。

説 明

- ・通信サイクル時間最大値には、SCAN_ClearCycleTime の実行や、電源 ON、ボードリセット以降の最大値が保持されています。

SCAN_GetMinCycleTime()

通信サイクル時間最小値読み出し

使用範囲： **オープン状態****機 能**

通信サイクル時間の最小値を取得します。

呼び出し形式

BOOL SCAN_GetMinCycleTime(DWORD Handle, WORD *CycleTime);

引 数

型	変数名	内容
DWORD	Handle	SCAN_Open()で得られたデバイスハンドル
WORD *	CycleTime	通信サイクル時間最小値を受け取るためのバッファアドレス

戻 り 値

指定された本ボードから、通信サイクル時間最小値が正常に取得できた場合は TRUE を、エラーが発生した場合は FALSE を返します。詳細なエラー情報は GetLastError() で取得することができます。

説 明

- ・通信サイクル時間最小値には、SCAN_ClearCycleTime の実行や、電源 ON、ボードリセット以降の最小値が保持されています。

SCAN_ClearCycleTime()

通信サイクル時間最大値・最小値クリア

使用範囲： **オープン状態****機 能**

本ボード内に保持されている通信サイクル時間の最大値および最小値をクリアします。

呼び出し形式

BOOL SCAN_ClearCycleTime(DWORD Handle);

引 数

型	変数名	内容
DWORD	Handle	SCAN_Open()で得られたデバイスハンドル

戻 り 値

指定された本ボード内の通信サイクル時間最大値／最小値を正常にクリアできた場合は TRUE を、エラーが発生した場合は FALSE を返します。詳細なエラー情報は GetLastError() で取得することができます。

使用範囲： **オープン状態****機 能**

スレーブ機能ステータスを取得します。

呼び出し形式

BOOL SCAN_GetSlaveModeStatus(DWORD Handle, WORD *Status, WORD *MacId);

引 数

型	変数名	内容
DWORD	Handle	SCAN_Open()で得られたデバイスハンドル
WORD *	Status	ステータスを受け取るためのバッファアドレス
WORD *	MacId	マスタノードアドレスを受け取るためのバッファアドレス

戻 り 値

指定された本ボードのスレーブ機能ステータスが正常に取得できた場合は TRUE を、エラーが発生した場合は FALSE を返します。詳細なエラー情報は GetLastError()で取得することができます。

説 明

- ・スレーブ機能ステータスは、下図のように 16 ビットで構成されています。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
B15	B14	B13	B12	—	—	—	—	—	—	—	—	B03	B02	—	—

各ビットの意味は、次のようになっています。

ビット	記号	意味
2	B02	出力データ 1/入力データ 1 I/O 通信異常
3	B03	出力データ 2/入力データ 2 I/O 通信異常
12	B12	コネクション 2 確立
13	B13	コネクション 1 確立
14	B14	出力データ 2/入力データ 2 I/O 通信中
15	B15	出力データ 1/入力データ 1 I/O 通信中

- ・マスタノードアドレスは、ビット 12 またはビット 13 が ON のときに、正しい値が格納されます。
- ・ビット 2、3 は、スレーブ機能での異常を表します。どちらか一方でも ON になると、SCAN_GetScannerStatus()で取得するスキャナステータスのビット 3 が ON になります。
- ・スレーブ機能で I/O データ入出力処理を行う前に、ビット 14 やビット 15 が ON であることを確認してください。

■異常履歴アクセスサービス API

ERROR_TIME 構造体

異常発生時刻データの形式を定義する構造体です。

ERROR_INFO 構造体内で使用され、SCAN_GetErrorLog()関数で使用されます。

型	変数名	内容
BYTE	Second	異常発生時刻 (秒)
BYTE	Minute	異常発生時刻 (分)
BYTE	Hour	異常発生時刻 (時)
BYTE	Day	異常発生時刻 (日)
BYTE	Month	異常発生時刻 (月)
BYTE	Year	異常発生時刻 (年)

ERROR_INFO 構造体

異常履歴データの形式を定義する構造体です。

ERROR_LOG 構造体内で使用され、SCAN_GetErrorLog()関数で使用されます。

型	変数名	内容
WORD	ErrorCode	異常コード
WORD	DetailCode	詳細コード
ERROR_TIME	Time	異常発生時刻

ERROR_LOG 構造体

全異常履歴データをまとめた構造体です。

SCAN_GetErrorLog()関数で使用されます。

型	変数名	内容
WORD	ErrorCount	格納異常履歴数
ERROR_INFO	ErrorInfo[64]	異常履歴情報

SCAN_GetErrorLog()

異常履歴読み出し

使用範囲： **オープン状態**

機 能

指定された本ボードから、異常履歴を取得します。

呼び出し形式

BOOL SCAN_GetErrorLog(DWORD Handle, ERROR_LOG *Log);

引 数

型	変数名	内容
DWORD	Handle	SCAN_Open()で得られたデバイスハンドル
ERROR_LOG *	Log	異常履歴データを受け取るためのバッファアドレス

戻 り 値

指定された本ボードから異常履歴データを正常に取得した場合は TRUE を、エラーが発生した場合は FALSE を返します。詳細なエラー情報は GetLastError()で取得することができます。

説 明

- ・異常コードの意味など、異常履歴機能の詳細については、「7-3 異常履歴機能」を参照してください。
- ・不揮発性メモリに保存されない異常履歴は、オープン状態中のみ保持されます。ボードクローズや、電源 OFF、ボードリセットにより、クリアされます。不揮発性メモリに保存された異常履歴は、再オープンや再起動の後でも読み出すことができます。

SCAN_ClearErrorLog()

異常履歴クリア

使用範囲： **オープン状態**

機 能

指定された本ボード内に記録されている異常履歴をクリアします。

呼び出し形式

BOOL SCAN_ClearErrorLog(DWORD Handle);

引 数

型	変数名	内容
DWORD	Handle	SCAN_Open()で得られたデバイスハンドル

戻 り 値

指定された本ボード内の異常履歴データを正常にクリアした場合は TRUE を、エラーが発生した場合は FALSE を返します。詳細なエラー情報は GetLastError()で取得することができます。

説 明

- ・不揮発性メモリに保存された異常履歴もクリアします。

■ PC ウォッチドッグタイマサービス API

SCAN_EnablePCWDTTimer()

PC-WDT 設定

使用範囲： **オープン状態**

機 能

本ボードに PC ウォッチドッグタイマ（PC-WDT）管理機能の有効／無効を設定します。

呼び出し形式

BOOL SCAN_EnablePCWDTTimer(DWORD Handle, BOOL Enable, WORD Timer);

引 数

型	変数名	内容
DWORD	Handle	SCAN_Open() で得られたデバイスハンドル
BOOL	Enable	PC-WDT の設定 TRUE : 有効 (PC-WDT 使用) FALSE : 無効 (PC-WDT 停止)
WORD	Timer	監視時間 0x0～0xFFFF (0～65535) : 左記値×10ms = 監視時間

戻 り 値

正しく処理が終了した場合は TRUE を、エラーが発生した場合は FALSE を返します。詳細なエラー情報は GetLastError() で取得することができます。

説 明

- ・ PC ウォッチドッグタイマ管理機能については、「3-11 PC ウォッチドッグタイマ管理機能の使い方」を参照してください。

SCAN_RefreshPCWDTTimer()

PC-WDT タイマ値更新

使用範囲： **オープン状態**

機 能

指定された本ボードの PC ウォッチドッグタイマ（PC-WDT）のタイマ値を更新します。

呼び出し形式

BOOL SCAN_RefreshPCWDTTimer(DWORD Handle);

引 数

型	変数名	内容
DWORD	Handle	SCAN_Open() で得られたデバイスハンドル

戻 り 値

正しく処理が終了した場合は TRUE を、エラーが発生した場合は FALSE を返します。詳細なエラー情報は GetLastError() で取得することができます。

説 明

- ・ PC ウォッチドッグタイマ管理機能を使用しているときは、設定されている PC-WDT タイマ値以内に、この関数でタイマ値を更新してください。タイマ値が更新されないまま、設定タイマ値の時間が経過すると、リモート I/O 通信が停止します。
- ・ PC ウォッチドッグタイマ管理機能については、「3-11 PC ウォッチドッグタイマ管理機能の使い方」を参照してください。

第 5 章 サンプルプログラム

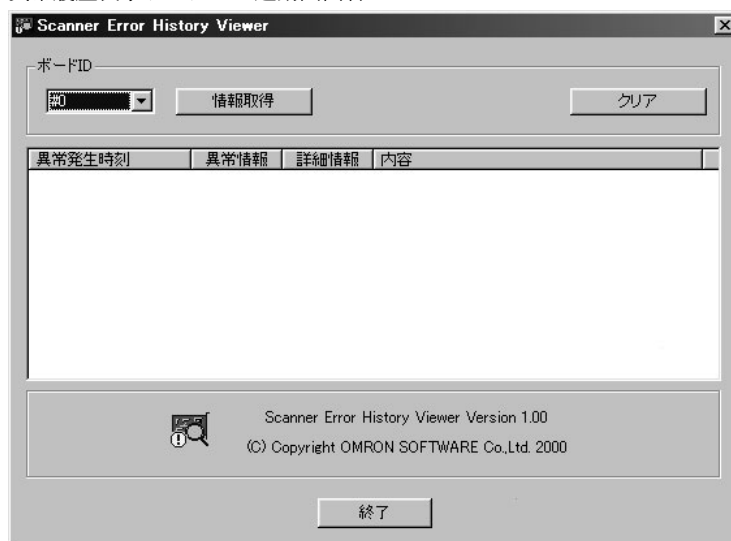
5-1 サンプルプログラムについて

Scanner SDK をインストールしたディレクトリには、次のサンプルプログラムが含まれています。

ディレクトリ	機能概要
Sample¥BusDELog	Scanner の異常履歴を表示するサンプルプログラムです。
Sample¥DemoToo	Scanner 機能を動作させ、リモート I/O 通信およびメッセージ通信を実行するサンプルプログラムです。

各サンプルプログラムは、スタートメニューのインストール時に指定したプログラムフォルダから起動することができます。

異常履歴表示サンプルの起動画面各



サンプルプログラムをビルドするには、Microsoft Visual C++ Ver6.0 以上が必要です。

なお、サンプルプログラムは、各 API の使用方法を説明するためのものであるため、そのまま使いになることはお止めください。

5-2 DeviceNet Scanner Demo の使い方

■DeviceNet Scanner Demo の機能

DeviceNet Scanner Demo には次の機能があります。

- ・ボードのオープン
- ・ボードのクローズ
- ・ネットワークへのオンライン
- ・ネットワークからのオフライン
- ・スキャンリストの作成
- ・スキャンリストの登録
- ・OUT/IN データの更新とモニタ
- ・I/O 通信開始
- ・I/O 通信停止
- ・Explicit クライアントリクエストメッセージの送信とレスポンスの受信

■I/O 通信を行う場合の使用例

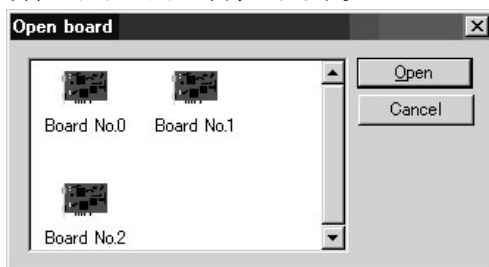
DeviceNet Scanner Demo を使用してスレーブと I/O 通信を行う場合の使用例を記載します。

(1) ボードのオープン

ボードをオープンし、使用可能にします。
ボードのオープンは、次の手順で行います。

①「Board」メニューから「Open」を選択します。

②以下のウィンドウが表示されます。



③使用するボードを選択し、「Open」ボタンをクリックします。

④指定したボードがオープンされ、使用可能になります。
ステータスバーには、ボード No. (ボード ID) が表示されます。

(2) スキャンリストの作成

DeviceNet Scanner Demo 内にスキャンリストを作成します。
スキャンリストの作成は、次の手順で行います。

- ① 「Edit」メニューから「Register to ScanList」を選択します。
- ② 以下のスレーブ登録ウィンドウが表示されます。

Add Device

MAC ID:

☒ Auto Connection
 OUT Size: Byte IN Size: Byte

☐ User Setup

☐ Use Poll Connection
 OUT Size: Byte IN Size: Byte

☐ Use Bit-Strobe Connection
 OUT Size: Byte IN Size: Byte

☐ Use Range of State Connection
 OUT Size: Byte IN Size: Byte

☐ Use Cyclic Connection
 OUT Size: Byte IN Size: Byte

☐ ACK Suppression

OK Cancel

- ③ スレーブのノードアドレス（MAC ID）と I/O サイズ（OUT Size、IN Size）を設定します。
 - I/O コネクションをユーザ指定する場合は、「User Setup」をチェックし、使用するコネクションと I/O サイズを設定します。
 - ④ 「OK」ボタンをクリックすると、メインウィンドウに登録されます。
 - ⑤ すべてのスレーブの登録が完了したら、「Cancel」ボタンをクリックします。
- 作成したスキャンリストが、以下のようにメインウィンドウに表示されます。

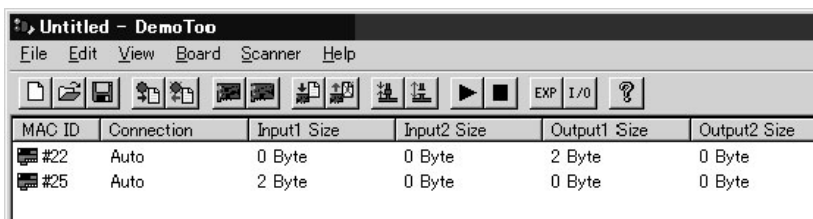
MAC ID	Connection	Input1 Size	Input2 Size	Output1 Size	Output2 Size
#22	Auto	0 Byte	0 Byte	2 Byte	0 Byte
#25	Auto	2 Byte	0 Byte	0 Byte	0 Byte

※スキャンリストがボードに登録されるまでは、MAC ID 欄のアイコンはグレーで表示されます。

(3) スキャンリストの登録

作成したスキャンリストをボードに登録します。
スキャンリストの登録は、次の手順で行います。

- ① 「Board」メニューから「Set ScanList」を選択します。
- ② スキャンリストが正しく登録されると、以下のように MAC ID 欄のアイコン表示がグレーから変化します。

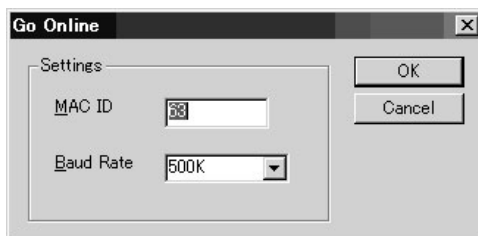


MAC ID	Connection	Input1 Size	Input2 Size	Output1 Size	Output2 Size
 #22	Auto	0 Byte	0 Byte	2 Byte	0 Byte
 #25	Auto	2 Byte	0 Byte	0 Byte	0 Byte

(4) ネットワークへのオンライン

ボードをオンラインにし、DeviceNet ネットワークに加入します。
ネットワークへのオンラインは、次の手順で行います。

- ① 「Scanner」メニューから「Online」を選択します。
- ② 以下のウィンドウが表示されます。

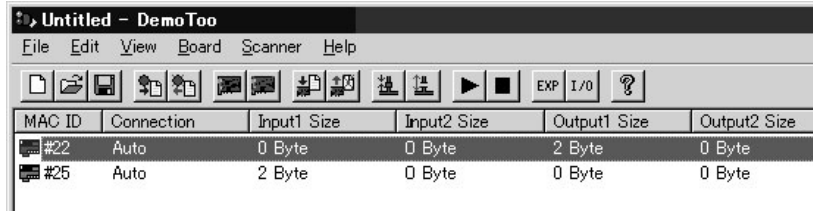


- ③ ボードのノードアドレス (MAC ID) と通信速度 (Baud Rate) を設定し、「OK」ボタンをクリックします。
- ④ ボードがオンラインになり、DeviceNet ネットワークに加入します。
ステータスバーの、一番右にはオンラインを表す「OL」が表示され、ボード No.の横にはノードアドレスが表示されます。
※この状態では、まだ I/O 通信は開始していません。

(5) OUT 初期データのセット

I/O 通信を開始する前に、OUT 初期データを設定します。
OUT データの設定は、次の手順で行います。

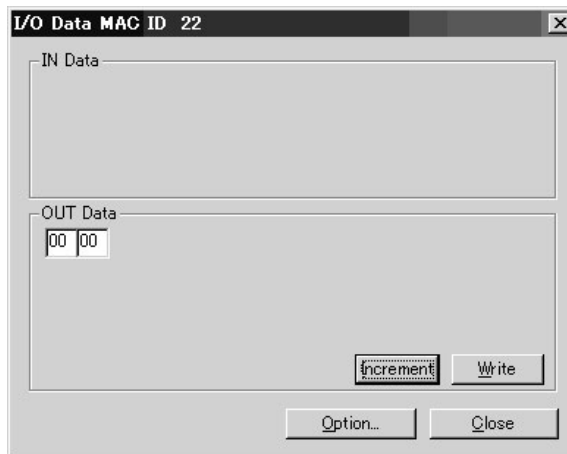
- ① OUT データを持つスレーブを選択します。



MAC ID	Connection	Input1 Size	Input2 Size	Output1 Size	Output2 Size
#22	Auto	0 Byte	0 Byte	2 Byte	0 Byte
#25	Auto	2 Byte	0 Byte	0 Byte	0 Byte

- ② 「Scanner」メニューから「I/O Data」を選択します。

- ③ 以下のウィンドウが表示されます。



I/O Data MAC ID 22

IN Data

OUT Data

00 00

Increment Write

Option... Close

- ④ 設定する OUT データを入力し、「Write」をクリックします。

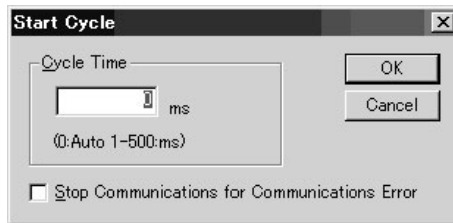
※この状態では、まだ OUT データは送信されません。

- ⑤ OUT データを持つスレーブすべてに対して、この操作を行います。

(6) I/O 通信開始

各スレーブとの間で I/O 通信を開始します。
I/O 通信の開始は、次の手順で行います。

- ① 「Scanner」メニューから「Start Cycle」を選択します。
- ② 以下のウィンドウが表示されます。



- ③ 通信サイクル時間（Cycle Time）を設定し、「OK」ボタンをクリックします。
通信サイクル時間を自動設定にする場合は、0を指定します。

- ④ I/O 通信が開始されます。

I/O 通信が正常に開始された場合は、以下のように各スレーブの Status 欄にステータスが表示されます。

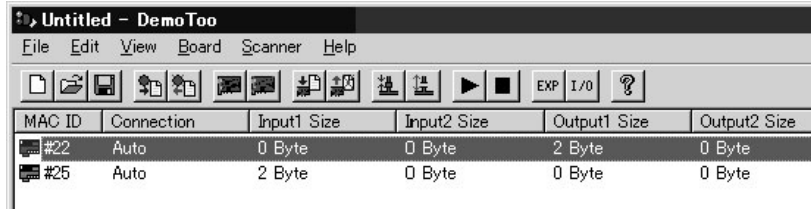
Untitled - DemoToo						
File Edit View Board Scanner Help						
MAC ID	Connection	Input1 Size	Input2 Size	Output1 Size	Output2 Size	Status
#22	Auto	0 Byte	0 Byte	2 Byte	0 Byte	I/O data communicating:
#25	Auto	2 Byte	0 Byte	0 Byte	0 Byte	I/O data communicating:

※コネクションの確立（ステータスが表示される）までに数秒かかります。

(7) OUT/IN データの更新とモニタ

I/O 通信中に、各スレーブの IN データのモニタあるいは OUT データの更新を行います。
データの更新・モニタは、次の手順で行います。

- ①スレーブを選択します。

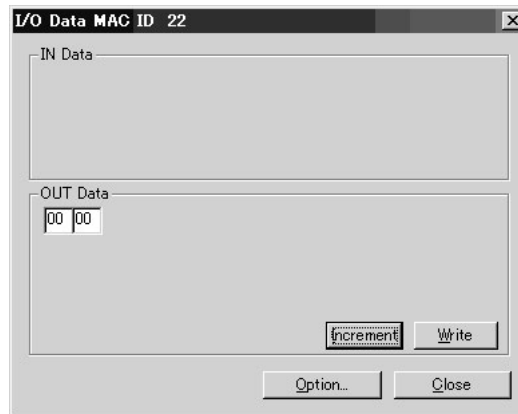


MAC ID	Connection	Input1 Size	Input2 Size	Output1 Size	Output2 Size
#22	Auto	0 Byte	0 Byte	2 Byte	0 Byte
#25	Auto	2 Byte	0 Byte	0 Byte	0 Byte

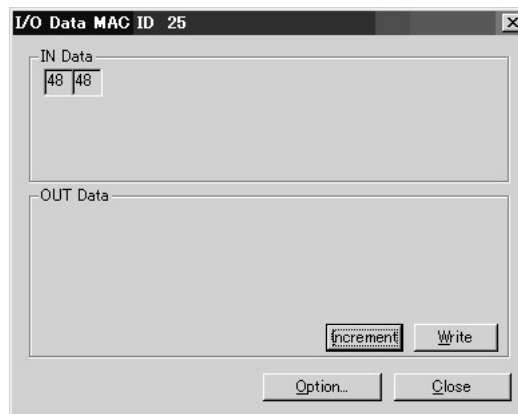
- ②「Scanner」メニューから「I/O Data」を選択します。

- ③以下の I/O Data ウィンドウが表示されます。

- ・ OUT データがある場合



- ・ IN データがある場合



IN データが表示されています。

- ④ I/O データ設定／モニタウィンドウは、同時に複数表示することができます。

●OUT データを自動更新するには

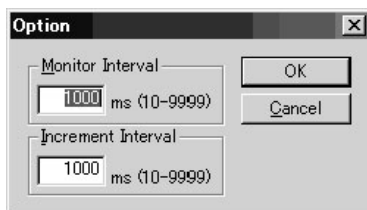
スレーブに送信する OUT データを自動更新することができます。
OUT データの自動更新は、次の手順で行います。

- ①該当スレーブの I/O Data ウィンドウを表示します。
- ②「Increment」 ボタンをクリックします。
- ③現在の OUT データが自動でインクリメントされ送信されます。
- ④「Stop」 ボタンをクリックすると、自動更新を中止します。

●IN データのモニタ間隔、OUT データの自動更新間隔を変更するには

IN データのモニタ間隔、OUT データの自動更新間隔を変更することができます。
間隔の変更は、次の手順で行います。

- ①該当スレーブの I/O Data ウィンドウを表示します。
- ②「Option」 ボタンをクリックします。
- ③以下のウィンドウが表示されます。



- ④インターバルタイム値（Monitor Interval、Increment Interval）を設定し、「OK」 ボタンをクリックします。
- ⑤設定した間隔で IN データのモニタ、OUT データの自動更新が行われます。

(8) I/O 通信停止

各スレーブとの I/O 通信を停止します。
I/O 通信の停止は、次の手順で行います。

- ① 「Scanner」メニューから「Stop Cycle」を選択します。
- ② 各スレーブとの I/O 通信が停止します。

(9) ネットワークからのオフライン

ボードをオフラインにし、DeviceNet ネットワークから離脱します。
ネットワークからのオフラインは、次の手順で行います。

- ① 「Scanner」メニューから「Offline」を選択します。
- ② PCI ボードがオフラインになり、DeviceNet ネットワークから離脱します。
ステータスバーの、一番右の「OL」と、ボード No.の横のノードアドレスが非表示になります。

(10) ボードのクローズ

ボードをクローズします。
ボードのクローズは、次の手順で行います。

- ① 「Board」メニューから「Close」を選択します。
- ② 以下のような確認ウィンドウが表示されます。



- ③ 「はい」ボタンをクリックすると、ボードがクローズされます。

■Explicit メッセージ通信を行う場合の使用例

DeviceNet Scanner Demo には、Explicit メッセージのクライアント機能があります。相手ノードに Explicit リクエストメッセージを送信し、そのレスポンスを受信することができます。

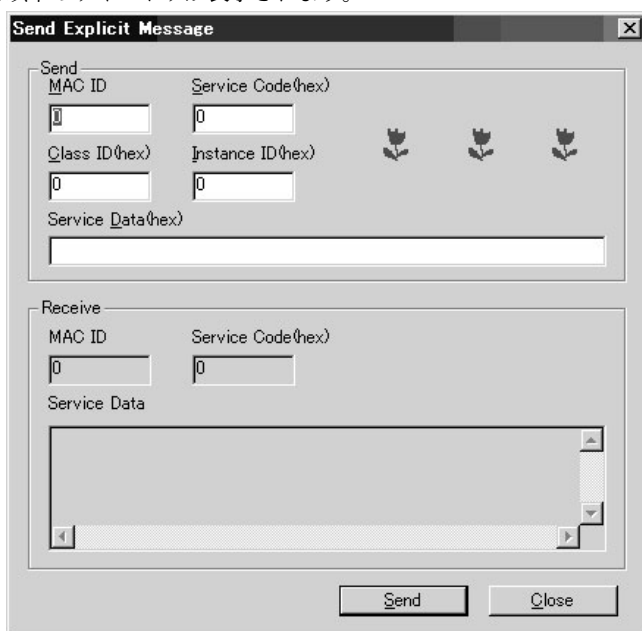
Explicit リクエストメッセージの送信と Explicit レスポンスメッセージの受信は、次の手順で行います。

- ① ボードをオープンし、ネットワークにオンラインします。

Explicit リクエストメッセージの送信は、I/O 通信の実行中/停止中にに関わりなく行うことができます。

- ② 「Scanner」メニューから「Send Explicit Message」を選択します。

- ③ 以下のウィンドウが表示されます。



The image shows a Windows-style dialog box titled "Send Explicit Message". It is divided into two main sections: "Send" and "Receive".

Send Section:

- MAC ID:** A text box containing the value "0".
- Service Code(hex):** A text box containing the value "0".
- Class ID(hex):** A text box containing the value "0".
- Instance ID(hex):** A text box containing the value "0".
- Service Data(hex):** A large, empty text area.

Receive Section:

- MAC ID:** A text box containing the value "0".
- Service Code(hex):** A text box containing the value "0".
- Service Data:** A large, empty text area with a scrollbar.

At the bottom of the dialog, there are two buttons: "Send" and "Close".

- ④ 送信先ノードアドレス (MAC ID) と、Explicit リクエストメッセージの各パラメータを設定し、「Send」ボタンをクリックします。

- ⑤ Explicit リクエストメッセージが送信され、レスポンスデータが表示されます。

- ⑥ Explicit リクエストメッセージの送信を終了する場合は、「Close」ボタンをクリックします。

第 6 章 通信タイミング

6-1 リモート I/O 通信の性能

ここでは、DeviceNet PCI ボード（以下、「本ボード」）とオムロン製スレーブを使用した場合の、リモート I/O 通信の性能について説明します。厳密な入出力のタイミングが要求される場合などに参考としてください。

なお、ここで示す計算では、次の条件を満たしていることを前提としています。

- ・必要なすべてのスレーブが加入していること
- ・ネットワーク上の全ノードで異常が発生していないこと
- ・パソコンから本ボードへの I/O リフレッシュ要求やコマンド要求がないこと

参 考

上記の条件を満たしていない場合は、ここで示す計算結果と一致しないことがあります。

■通信サイクル時間

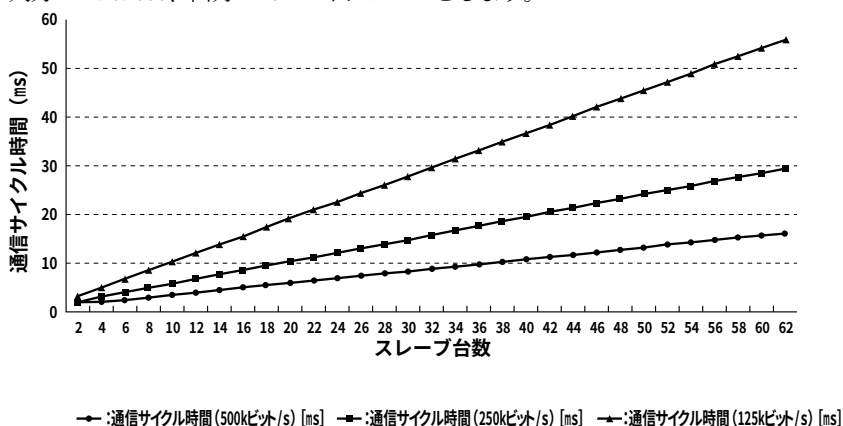
通信サイクル時間とは、あるスレーブへの I/O データ通信を処理してから、再び同じスレーブへの I/O データ通信を処理するまでの時間のことです。通信サイクル時間は、最大入出力応答時間を算出するためにも使用されます。

通信サイクル時間は、ネットワーク中にマスタが 1 台だけか／複数台存在するか、メッセージ通信が行われたかなど、様々な要因によって変化します。ここでは、マスタが 1 台の場合について説明します。マスタが複数台の場合は、「1 ネットワーク内にマスタが複数台の場合」（P.6-4）を参照してください。

通信サイクル時間のグラフ

下記に、出力 16 点と入力 16 点のスレーブが混在した場合の、スレーブ台数に対する通信サイクル時間のグラフを示します。

入力：Bit-strobe、出力：Poll コネクションとします。



通信サイクル時間の計算式

1 ネットワーク内にマスタが 1 台の場合の、通信サイクル時間 (T_{RM}) の計算式を、以下に示します。

ただし、計算結果が 2ms 未満の場合は、 $T_{RM}=2ms$ として動作します。

T_{RM}	$= \Sigma (1 \text{ スレーブ通信時間})$ $+ \text{多点ユニット処理時間}$ $+ \text{Explicit メッセージ処理時間}$ $+ \text{COS/Cyclic コネクション通信時間 [ms]}$ $+ 0.01 \times N + 1.0 \quad [\text{ms}]$
----------	---

1 スレーブ通信時間：

1 つのスレーブの通信に要する時間

「 $\Sigma (1 \text{ スレーブ通信時間})$ 」は、ネットワーク内の全スレーブの 1 スレーブ通信時間の合計を表します。

多点ユニット処理時間：

3.5 [ms]

IN・OUT の片方または両方が 8 バイトを超えるスレーブが 1 台でもある場合のみ加算

Explicit メッセージ処理時間：

$(0.11 \times T_B) \times n$ [ms]

Explicit メッセージ通信 (送信または受信) 実行時のみ加算

n ： CPU ユニットの 1 サイクルタイム内で同時に発生する Explicit メッセージ数 (送信および受信をともに含む)

T_B ： 定数 (500k ビット/s 時=2、125k ビット/s 時=4、125k ビット/s 時=8)

COS/Cyclic コネクション通信時間 [ms]：

$\{ (0.05 + 0.008 \times S) \times T_B \} \times n$ [ms]

S ： COS/Cycli コネクションの IN および OUT の合計サイズ (バイト単位)

T_B ： 500k ビット/s=2、250k ビット/s=4、125k ビット/s=8

n ： 1 通信サイクル時間内で同時に発生する、COS/Cyclic コネクションの対象ノード数

N ：スレーブの台数

1 スレーブ通信時間

1 スレーブ通信時間とは、1 つのスレーブとの通信に要する時間のことです。
スレーブのタイプ別の、1 スレーブ通信時間 (T_{RT}) の計算式を、以下に示します。

●OUT が 8 バイト以内の、出力スレーブの場合

$$T_{RT} = 0.016 \times T_B \times S_{OUT1} + 0.11 \times T_B + 0.07 \quad [\text{ms}]$$

S_{OUT1} : 出力スレーブの OUT チャンネル数
 T_B : 500k ビット/s=2、250k ビット/s=4、125k ビット/s=8

●IN が 8 バイト以内の、入力スレーブの場合

$$T_{RT} = 0.016 \times T_B \times S_{IN1} + 0.06 \times T_B + 0.05 \quad [\text{ms}]$$

S_{IN1} : IN スレーブの IN チャンネル数
 T_B : 500k ビット/s=2、250k ビット/s=4、125k ビット/s=8

●IN と OUT のどちらも 8 バイト以内の、入出力混在スレーブの場合

$$T_{RT} = 0.016 \times T_B \times (S_{OUT2} + S_{IN2}) + 0.11 \times T_B + 0.07 \quad [\text{ms}]$$

S_{OUT2} : 入出力混在スレーブの OUT チャンネル数
 S_{IN2} : 入出力混在スレーブの IN チャンネル数
 T_B : 500k ビット/s=2、250k ビット/s=4、125k ビット/s=8

●IN・OUT の片方または両方が 8 バイトを超える、スレーブの場合

$$T_{RT} = T_{OH} + T_{\text{BYTE-IN}} \times B_{IN} + T_{\text{BYTE-OUT}} \times B_{OUT} \quad [\text{ms}]$$

T_{OH} : プロトコルオーバーヘッド
 $T_{\text{BYTE-IN}}$: IN のバイト伝送時間
 B_{IN} : IN のバイト数
 $T_{\text{BYTE-OUT}}$: OUT のバイト伝送時間
 B_{OUT} : OUT バイト数

通信速度	T_{OH}	$T_{\text{BYTE-IN}}$	$T_{\text{BYTE-OUT}}$
500k ビット/s	0.306ms	0.040ms	0.036ms
250k ビット/s	0.542ms	0.073ms	0.069ms
125k ビット/s	1.014ms	0.139ms	0.135ms

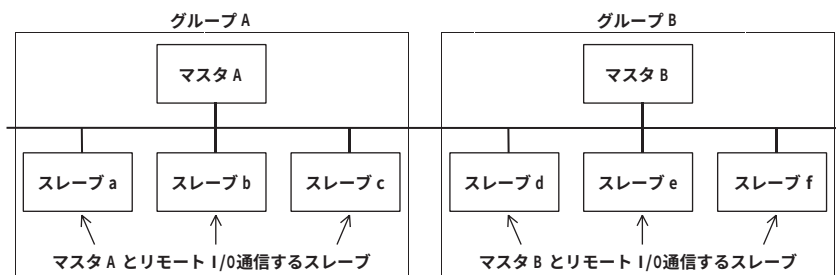
IN のみのスレーブの場合は B_{OUT} を 0 とし計算し、OUT のみのスレーブの場合は B_{IN} を 0 とし計算してください。

■1 ネットワーク内にマスタが複数台の場合

1 ネットワーク内にマスタが複数台ある場合は、通信サイクル時間（ T_{RM} ）は以下のようになります。

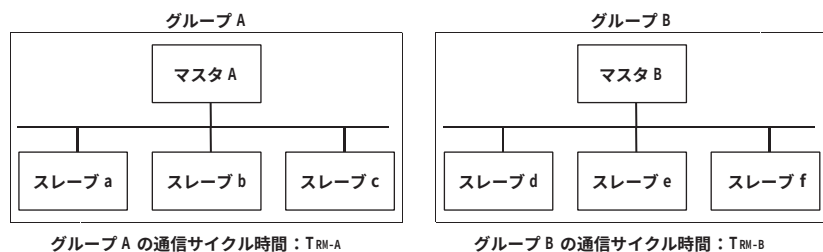
ここでは、マスタが2台の場合を例に、説明していきます。

初めに、下図のように、マスタ A とリモート I/O 通信しているグループと、マスタ B とリモート I/O 通信しているグループに、ノードをグループ分けします。



上図では、便宜上スレーブの位置もマスタごとに分けてありますが、実際のスレーブの物理的な位置は無関係です。

次に、「通信サイクル時間」（P.6-1）を参照して、それぞれのグループが別々のネットワークだった場合の、それぞれの通信サイクル時間を計算します。



マスタが2台あるネットワークでは、ネットワーク全体の通信サイクル時間は、以下のようになります。

$$T_{RM} = T_{RM-A} + T_{RM-B}$$

ここでは、マスタが2台あるネットワークを例に示しましたが、マスタが複数台あるネットワークでも同様に、リモート I/O 通信ごとにグループ分けし、それぞれが独立したネットワークだった場合の通信サイクル時間の合計が、ネットワーク全体の通信サイクル時間となります。

第 7 章 異常時の処置

7-1 LED 表示と異常時の処置

DeviceNet PCI ボード（以下、「本ボード」）では、本ボードの状態を表す MS LED と、ネットワークの状態を表す NS LED の 2 つの LED により、異常の有無や、異常内容を知ることができます。

ここでは、本ボードの LED 表示の意味と、異常時の処置について説明します。ただし、ここでは本ボードのセットアップが正常に完了していることを前提にしています。

■正常動作中の LED 表示

本ボードが正常に動作している場合の LED の状態を、下表に示します。

LED の状態	状態	備考
MS ● 消灯 NS ● 消灯	初期処理開始待ち	ブートプログラムの初期処理を実行中です。
MS 点滅 緑点滅 NS ● 消灯	スキャナプログラム開始待ち	スキャナファームウェアの初期処理を実行中です。
MS 点滅 緑点灯 NS ● 消灯	MAC ID 重複チェック中	オンライン要求待ち、および要求受付後、自ノードと他ノードのノードアドレス重複チェック中です。
MS 点滅 緑点灯 NS 点滅 緑点滅	リモート I/O 通信停止中で、メッセージ通信未確立	オンライン状態になっていても、まだ他ノードとの通信が確立していない状態です。 メッセージ通信未確立とは、自ノードから他ノードへのメッセージを送信しておらず、また他ノードからのメッセージも受信していない状態です。
MS 点滅 緑点灯 NS 点滅 緑点灯	リモート I/O 通信中またはメッセージ通信中	リモート I/O 通信、メッセージ通信のどちらか、または両方が実行中の状態です。 正常に通信している状態です。

点滅：点灯 点滅：点滅 ●：消灯

■異常発生時の LED 表示と処置

LED の状態	状態	異常内容と処置
MS  赤点滅 NS — 変化なし	ハードウェア異常	以下のハードウェア異常が発生しました。 ・EEPROM 異常 本ボードを交換してください。
MS  赤点滅 NS ● 消灯	PC-WDT 異常	本ボードの PC ウォッチドッグタイマ管理機能により、PC ウォッチドッグタイマ異常が検出された状態（パソコン上のアプリケーションが停止している状態）です。 パソコン上のアプリケーション、またはパソコン本体を再起動してください。
MS  赤点灯 NS ● 消灯	ハードウェア異常	以下のいずれかのハードウェア異常が発生しました。 ・BD-WDT 異常 ・RAM 異常 本ボードを交換してください。
MS — 変化なし NS  赤点灯	ノードアドレス重複	本ボードとノードアドレスが重複している他ノードがあります。 全ノードのノードアドレス設定を確認の上、パソコン本体の再起動、または本ボードのリセットを行ってください。
	Busoff 検知	データ異常が多発し、通信が停止しました。 以下の項目を検討してください。 ・全ノードの通信速度設定が同一か ・ケーブル長（幹線/支線）は適切か ・ケーブルの断線・ゆるみがないか ・終端抵抗が幹線の両端のみにあるか ・ノイズが多くないか
MS — 変化なし NS ● 消灯	ネットワーク電源異常（送信異常）	通信用の電源が正常に供給されていません。 ネットワーク電源と、ネットワークケーブルの配線を確認してください。
	送信タイムアウト（送信異常）	以下のいずれかが原因で、送信が正常終了できませんでした。 ・スレーブが 1 台もない ・同一ネットワーク上に他のマスタが存在する ・CAN コントローラが異常 ・マスタ/スレーブの通信速度が不一致 以下の項目を検討してください。 ・全ノードの通信速度設定が同一か ・ケーブル長（幹線/支線）は適切か ・ケーブルの断線・ゆるみがないか ・終端抵抗が幹線の両端のみにあるか ・ノイズが多くないか

LED の状態	状態	異常内容と処置
MS — 変化なし NS 〇 赤点減	スレーブ不在 (照合異常)	スキャンリストに登録されているスレーブが存在していません。 以下の項目を検討してください。 ・全ノードの通信速度設定が同一か ・ケーブル長（幹線/支線）は適切か ・ケーブルの断線・ゆるみがないか ・終端抵抗が幹線の両端のみにあるか ・ノイズが多くないか
	未サポートスレーブあり	1 スレーブの I/O サイズが 200 バイトを超えているスレーブが接続されています。 I/O サイズが 200 バイトを超えるスレーブを、ネットワークから外してください。
	スレーブ I/O サイズ不一致（照合異常）	スレーブの I/O サイズが、スキャンリストの登録内容と一致していません。 スレーブを点検後、スキャンリストに登録し直してください。
	I/O 通信異常	I/O 通信でタイムアウトが発生しました。 以下の項目を検討してください。 ・全ノードの通信速度設定が同一か ・ケーブル長（幹線/支線）は適切か ・ケーブルの断線・ゆるみがないか ・終端抵抗が幹線の両端のみにあるか ・ノイズが多くないか
MS ● 消灯 NS ● 消灯	システム異常	本ボードを交換してください。

〇：点灯 〇：点滅 ●：消灯

7-2 関数で検知される異常と処置

ここでは、本ボードのマスタ機能用 API 関数を使用することで調べることができる異常の内容と、異常に対する処置について説明します。

3-4 「■関数の戻り値によるエラーのチェック」 (P.3-6 参照)

■エラーコード

本ボードのマスタ機能用の API 関数は、すべて BOOL 型となっていて、異常終了時には、戻り値として FALSE を返します。FALSE が返された場合は、詳細なエラー情報 (エラーコード) を、GetLastError()関数で読み出すことができます。

ここでは、エラーコードの内容と処置について説明します。

【参考】

エラーコードによるエラーのチェック方法については、3-4 「■関数の戻り値によるエラーのチェック」 (P.3-6) を参照してください。

エラーコード	値	エラーの内容と処置
SCAN_INVALID_HANDLE	0x20000001	ドライバハンドルが無効です。正しいドライバハンドルを指定して再度実行してください。
SCAN_ALREADY_USED	0x20000002	指定ドライバは他のアプリケーションで使用中です。他のアプリケーションを終了して再度実行してください。
SCAN_ERROR_DRIVER_FUNCTION	0x20000003	ドライバファンクションでエラーが発生しました。パラメータを確認して再度実行してください。
SCAN_NOT_EXIST_DEVICE	0x20000004	指定されたボード ID のボードは存在しません。存在するボードのボード ID を指定してください。
SCAN_TIMEOUT	0x20000005	コマンドがタイムアウトしました。ボードをリセットし、再度実行してください。
SCAN_NOT_MODULE_PATH	0x20000006	通信モジュールパスが見つかりません。再度インストールしてください。
SCAN_NOT_EXIST_MODULE	0x20000007	通信モジュールが見つかりません。再度インストールしてください。
SCAN_NOT_ALLOCATE_MEMORY	0x20000008	メモリのアロケートができません。他のアプリケーションを終了させて再度実行してください。
SCAN_NOT_OPEN_DRIVER	0x20000009	ドライバがオープンできません。正しくドライバがインストールされていることを確認して再度実行してください。
SCAN_NOT_CREATE_EVENT	0x2000000A	割り込みイベントを作成することができません。
SCAN_INVALID_PARAMETER	0x2000000B	無効なパラメータです。正しいパラメータを指定して再度実行してください。
SCAN_NOT_EXECUTECOMMAND	0x2000000C	コマンド実行中にエラーが発生しました。パラメータを確認して再度実行してください。
SCAN_NOT_DOWNLOAD	0x2000000D	通信モジュールダウンロード中にエラーが発生しました。ハードウェアに異常がないか確認して再度実行してください。
SCAN_NOT_ENOUGH_BUFFER	0x20001000	データを格納するのに十分なバッファではありません。バッファサイズを大きくしてください。
SCAN_RES_NOT_RECEIVED	0x20001001	クライアントレスポンスを受信していません。
SCAN_REQ_NOT_RECEIVED	0x20001002	サーバリクエストを受信していません。
SCAN_INVALID_REGISTHANDLE	0x20001003	登録するウィンドウあるいはスレッドハンドルが正しくありません。正しいハンドルを指定してください。
SCAN_INVALID_REGISTMSG	0x20001004	登録する通知メッセージが正しくありません。正しいメッセージを指定してください。
SCAN_NOT_EVENTDATA	0x20001005	イベントデータがありません。
SCAN_NOT_EVENT	0x20001006	該当イベントは存在しません。
SCAN_FILE_IO_ERROR	0x20001007	ファイルアクセス中にエラーが発生しました。ファイルがあるか、壊れていないか確認し、再度実行してください。
SCAN_INVALID_CMD	0x20010010	コマンドコードが正しくありません。正しいコードを指定してください。

エラーコード	値	エラーの内容と処置
SCAN_SYSTEM_BUSY	0x20010011	システムがビジー状態です。
SCAN_OFF_LINE	0x20010020	オフラインのため、実行できません。オンラインにして再度実行してください。
SCAN_PROCESSING_ON_LINE	0x20010021	コネクション初期化中のため、実行できません。しばらく待って再度実行してください。
SCAN_ON_LINE	0x20010022	オンラインのため、実行できません。オフラインにして再度実行してください。
SCAN_SCANNING	0x20010023	スキャン中のため、実行できません。スキャンを停止して再度実行してください。
SCAN_NOT_SCANNING	0x20010024	スキャン停止中のため、実行できません。スキャンを開始し再度実行してください。
SCAN_AUTO_SCANNING	0x20010025	無効モードで動作中のため実行できません。有効モードにし再度実行してください。
SCAN_FIXED_SCANNING	0x20010026	既に無効モードで動作中です。
SCAN_BUS_OFF	0x20010029	バスオフ発生中のため、実行できません。エラーを解除して再度実行してください。
SCAN_CONNECTED	0x2001002A	既にコネクションが確立されています。
SCAN_DISCONNECTED	0x2001002B	既にコネクションが切断されています。
SCAN_NO_NET_POWER	0x2001002C	ネットワーク電源が供給されていません。ネットワーク電源を供給してください。
SCAN_TX_TIME_OUT	0x2001002D	送信でタイムアウトが発生しました。送信先のデバイスが存在しないか、優先度が高いメッセージでネットワークが占有されています。少し間隔をおいてから、再度、再送してください。
SCAN_DUP_MAC_ERROR	0x2001002E	指定したノードアドレスが他のデバイスと重複しています。他のノードアドレスを指定してください。
SCAN_INVALID_BAUD_RATE	0x20010040	通信速度の指定が正しくありません。正しい通信速度を指定してください。
SCAN_INVALID_MAC_ID	0x20010041	無効なノードアドレスです。正しいノードアドレスを指定して再度実行してください。
SCAN_INVALID_SCAN_TYPE	0x20010042	スキャンタイプが不正です。正しいスキャンタイプを指定して再度実行してください。
SCAN_INVALID_IO_SIZE	0x20010043	データサイズが正しくありません。正しいサイズを指定して再度実行してください。
SCAN_INVALID_CLASS_ID	0x20010044	指定されたクラス ID は登録されていません。
SCAN_INVALID_INSTANCE_ID	0x20010045	インスタンス ID が正しくありません。正しい ID を指定してください。
SCAN_INVALID_ATTRIBUTE_ID	0x20010046	アトリビュート ID が正しくありません。正しい ID を指定してください。
SCAN_INVALID_SELF_SLAVE	0x20010047	有効スレーブスキャンリストが未登録です。スレーブスキャンリストに登録してください。
SCAN_TOO_MANY_CLASS	0x20010048	クラス ID の登録数をオーバーしました。
SCAN_INVALID_SCAN_LIST	0x20010049	スキャンリストに登録されているデバイスがありません。デバイスの登録を行ってください。
SCAN_NO_COS_CNXN	0x20010060	COS コネクションが確立されていません。コネクションを確立して再度、実行してください。
SCAN_NO EMC_CNXN	0x20010061	Explicit コネクションが確立されていません。コネクションを確立して再度、実行してください。
SCAN EMC_CNXN_TIME_OUT	0x20010062	Explicit コネクションの確立でタイムアウトが発生しました。Explicit コネクション先を確認し、再度、実行してください。
SCAN_SND_MSG_TOO_LONG	0x20010063	送信データサイズがオーバーしました。
SCAN_RSP_MSG_TOO_LONG	0x20010064	受信データサイズがオーバーしました。
SCAN_DEST_DEVICE_OVERFLOW	0x20010065	送信バッファがオーバーフローしました。少し間隔をおいてから、再度、再送してください。
SCAN_MEMORY_ACCESS_CONFLICT	0x20010080	I/O 割り付けが重複しています。重複しないように設定してください。
SCAN_MEMORY_ERROR	0x20010081	EEPROM 書き込み異常が発生しました。
SCAN_SUM_CHECK_ERROR	0x20010082	サムチェック異常が発生しました。

7-3 異常履歴機能

本ボードには、発生した異常の内容を表す「異常コード」を最大 64 件まで記録できる「異常履歴機能」があります。

本ボード内に記録されている異常履歴データは、SCAN_GetErrorLog()関数によって読み出すことができます (P.4-56 参照)。また、SCAN_ClearErrorLog()関数によってクリアすることもできます (P.4-56 参照)。

参 考

- ・異常履歴の中でも重要な履歴は、不揮発性メモリに保存されます。不揮発性メモリに保存されない履歴は、オープン状態中のみ保持されます。ボードクローズや、電源 OFF、リセットにより、すべてクリアされます。
- ・不揮発性メモリに保存された異常履歴は、再オープンや再起動の後でも読み出すことができます。
- ・本ボードのソフトウェアに添付されている「Scanner Error History Viewer」によっても、不揮発性メモリに保存された異常履歴を読み出すことができます。

■異常履歴データ

1 件 (レコード) の異常履歴データは、次のように構成されています。本ボード内には、最大 64 件の異常履歴データを記録でき、64 件を超えた場合は、古いデータから破棄されていきます。

変数名	型	内容(BCD 形式)
ErrCode	WORD	異常コード
DetailCode	WORD	詳細コード
Second	BYTE	異常発生時刻 (秒)
Minute	BYTE	異常発生時刻 (分)
Hour	BYTE	異常発生時刻 (時)
Day	BYTE	異常発生時刻 (日)
Month	BYTE	異常発生時刻 (月)
Year	BYTE	異常発生時刻 (年)

参 考

異常履歴データは、ERROR_INFO 構造体で定義します (P.4-55 参照)。

■異常コード一覧

以下に、異常履歴機能で記録される、異常コードとその内容を示します。処置については、「異常発生時の LED 表示と処置」(P.7-2)を参照してください。

異常コード (HEX)	異常内容	詳細情報		EEP-ROM
		1 バイト目 (オフセット=0002h)	2 バイト目 (オフセット=0003h)	
0001	PC_WDT 異常 (PC_WDT 監視機能におけるタイムアウト)	00 Hex	00 Hex	○
0101	オフライン中のためレスポンスメッセージ送信不可	80 Hex	bit0-bit5: クライアントデバイス MAC ID bit6: OFF bit7: ON	×
0106	重複 MAC ID 異常発生中のためレスポンスメッセージ送信不可	80 Hex	bit0-bit5: クライアントデバイス MAC ID bit6: OFF bit7: ON	×
0107	コネクション未確立のためレスポンスメッセージ送信失敗	80 Hex	bit0-bit5: クライアントデバイス MAC ID bit6: OFF bit7: ON	×
0109	相手先バッファフル	80 Hex	bit0-bit5: クライアントデバイス MAC ID bit6: OFF bit7: ON	×
0111	サービスデータサイズオーバ	80 Hex	bit0-bit5: クライアントデバイス MAC ID bit6: OFF bit7: ON	×
0117	内部バッファフル	00 Hex	bit0-bit5: サーバデバイス MAC ID bit6: OFF bit7: ON	×
0112	メッセージ ID 不正のためレスポンスメッセージ廃棄	80 Hex	bit0-bit5: クライアントデバイス MAC ID bit6: OFF bit7: ON	×
0211	重複 MAC ID 異常	00 Hex	自 MAC ID	×

異常コード (HEX)	異常内容	詳細情報		EEP-ROM
		1 バイト目 (オフセット=0002h)	2 バイト目 (オフセット=0003h)	
021A	設定テーブル論理異常	00 Hex	0A Hex: スキャンリスト 0B Hex: スレーブスキャンリスト 0C Hex: メッセージ監視タイマリスト	○
0340	バスオフ検知	00 Hex	00 Hex	×
0341	ネットワーク電源異常	00 Hex	00 Hex	×
0342	送信タイムアウト異常	00 Hex	00 Hex	×
0343	I/O 構成異常	03 Hex: 未サポートスレーブ	異常スレーブ MAC ID	×
0344	照合異常	01 Hex: スレーブ不在 (自 MAC ID 使用含む) 02 Hex: ベンダ ID 不正 03 Hex: プロダクトタイプ 不正 04 Hex: プロダクトコード 不正 05 Hex: コネクション 未サポート 06 Hex: I/O サイズ不一致 07 Hex: コネクションパス 不正	スレーブ MAC ID	×
0345	I/O 通信異常	01 Hex マスタ機能 02 Hex スレーブ機能	マスタ機能: スレーブ MAC ID スレーブ機能: マスタ MAC ID	×
0346	I/O 通信異常によりスキャン停止	01 Hex: 通信異常 02 Hex: ネットワーク電源 異常 03 Hex: 送信タイムアウト	通信異常: 異常スレーブ MAC ID ネットワーク電源異常: 自 MAC ID 送信タイムアウト: 自 MAC ID	×
0348	新規リクエスト受信によりレスポンスメッセージ廃棄	80 Hex	bit0-bit5: クライアントデバイス MAC ID bit6: OFF bit7: ON	×

異常コード (HEX)	異常内容	詳細情報		EEP-ROM
		1 バイト目 (オフセット=0002h)	2 バイト目 (オフセット=0003h)	
0601	システム異常	不定		○
0602	メモリ異常	01 Hex: 読み出しエラー 02 Hex: 書き込みエラー	06 Hex: 異常履歴 09 Hex: Identity 情報 0A Hex: スキャンリスト 0B Hex: スレーブスキャン リスト 0C Hex: メッセージ監視タイマ リスト 0D Hex: 通信サイクル時間 設定値	○*注

注：異常履歴エリア（EEP-ROM）がメモリ異常の場合は、EEP-ROM への異常登録を行わない。

●本誌に記載のない条件や環境での使用、および原子力制御・鉄道・航空・車両・燃焼装置・医療機器・娯楽機械・安全機器、その他人命や財産に大きな影響が予測されるなど、特に安全性が要求される用途に使用される際には、当社の意図した商品用途の場合や特別の合意がある場合を除き、当社は当社商品に対して一切保証をいたしません。
●本製品の内、外国為替及び外国貿易法に定める輸出許可、承認対象貨物(又は技術)に該当するものを輸出(又は非居住者に提供)する場合は同法に基づく輸出許可、承認(又は役務取引許可)が必要です。

オムロン株式会社 インダストリアルオートメーションビジネスカンパニー

●製品に関するお問い合わせ先

お客様相談室

 フリー通話 **0120-919-066** クイック オムロン

携帯電話・PHS・IP電話などではご利用いただけませんので、下記の電話番号へおかけください。

電話 **055-982-5015** (通話料がかかります)

■営業時間：8:00～21:00 ■営業日：365日

●FAXやWebページでもお問い合わせいただけます。

FAX **055-982-5051** / www.fa.omron.co.jp

●その他のお問い合わせ

納期・価格・サンプル・仕様書は貴社のお取引先、または貴社担当オムロン販売員にご相談ください。
オムロン制御機器販売店やオムロン販売拠点は、Webページでご案内しています。

オムロン制御機器の最新情報をご覧ください。

www.fa.omron.co.jp

緊急時のご購入にもご利用ください。

オムロン商品のご用命は