YASKAWA

MOTOCOM32
操作要領書

本説明書は、最終的に本製品をお使いになる方のお手元に確実に届けられるよう、
お取り計らい願います。

資料番号
HW9481253 ⑯ 1/310

危険

- 本説明書は、MOTOCOM32 について詳しく説明しています。必ずご一読を願い、十分にご理解いただいたうえで、お取り扱いいただくようお願いします。
- また、安全についての一般事項は、コントローラの取扱説明書の「1 安全について」に記載しています。必ず熟読していただき、正しくお使いいただきますようお願いいたします。

注意

- お客様による製品の改造は、当社の保証範囲外ですので責任を負いません。
- 本マニュアルに記載されているソフトウェアは、ライセンスの所有者に対してのみ供給され、同ライセンスの許可する条件のもとでのみ使用または複写することが許されています。

通知

- 説明書に掲載している図及び写真は、代表事例であり、お届けした製品と異なる場合があります。
- 説明書は、製品の改良や仕様変更、及び説明書自身の使いやすさの向上のために適宜変更されることがあります。
  この変更は改訂版として表紙右下の説明書番号の更新によって行われます。
- 損傷や紛失などにより、説明書を注文される場合は、当社代理店または説明書の裏表紙に記載している最寄りの営業所に表紙の資料番号を連絡してください。

重要

本説明書は、小形機種用制御盤「FS100」及び中・大形機種用制御盤「FS100L」に共通の説明書です。
特に断りが無い限り、本説明書において “FS100” の表記は「FS100」及び「FS100L」の双方を指します。

安全上のご注意

ご使用の前に、必ずこの説明書とその他の付属書類をすべて熟読し、機器の知識、安全の知識そして注意事項のすべてについても習熟してから、正しくご使用ください。
本説明書は、安全注意事項のランクを「危険」、「警告」、「注意」、「通知」に区分して掲載しています。

回避しないと死亡または重症、火災を招く差し迫った危険な状態を示す。

回避しないと死亡または重症、火災を招く恐れがある危険な状態を示す。

回避しないと軽症または中程度の障害、火災を招くかもしれない危険な状態を示す。

回避しないと人身事故、火災以外の限定した損害（物損等）を引き起こす危険性がある状態を示す。

なお、「注意」に記載した事項でも、状況によっては重大な結果に結びつく可能性があります。いずれも重要な内容を記載していますので、必ず守ってください。

「危険」、「警告」と「注意」には該当しませんが、ユーザーに必ず守っていただきたい事項を、関連する個所に併記しています。

メニュー、ボタンの表記

メニュー、ボタンの表記については以下のように表します。

機　器	本書での表記
メニュー	メニューは【　】で囲んで表します。 例：【ツール】
ボタン	画面に表示されるボタンは［　］で囲んで表します。 例：［座標］

操作手順の表現についての定義

操作手順の説明において、「＊＊を選択」という表現は、

　　マウスで対象項目にカーソルを移動させ、左クリックを押す。

または、

　　［Tab］キーで対象項目を移動させ、［Enter］キーを押す。

　　（メニューは、矢印キーで移動させ、［Enter］キーを押す。）

という操作を表します。

商標の表記について

本書で使用するシステム名、製品名は、それぞれ各社の商標、または登録商標です。これらの記述にあたり、本文中での明示的な表示は行っておりません。

1 はじめに

2 インストールの方法

3 High Speed JobExchanger 機能

4 ホストコントロール機能

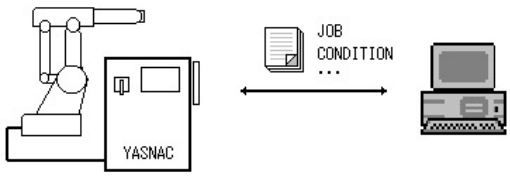
1 はじめに

1.1 MOTOCOM32 について

MOTOCOM32（モートコム３２）はパーソナルコンピュータと、当社産業用ロボット MOTOMAN 用制御盤 YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、YASNAC XRC、MRC、MRC2、ERC、ERC2 とのデータ伝送を行うソフトウェアです。YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、YASNAC とパーソナルコンピュータ間をシリアルケーブル（RC232C ケーブル）で接続、あるいは YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、YASNAC XRC、MRC に Ethernet ケーブルを接続 (YASNAC XRC、MRC では Ethernet I/F 基板の取り付けが必要) し、パーソナルコンピュータと LAN を構成することによって高速なデータ伝送を実現します。本ソフトウェアはパーソナルコンピュータ上で動作し、以下の機能を持っています。

High Speed JobExchanger 機能

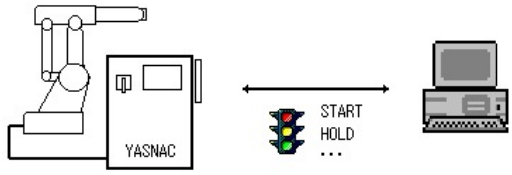
パソコンからの指令によって、ジョブや条件データなどのファイルをロード・セーブします。



ホストコントロール機能

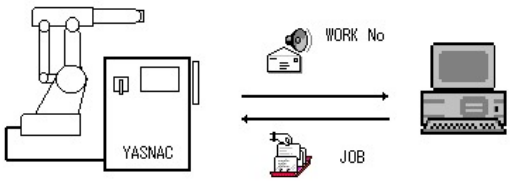
パソコンからの指令によって、以下の作業を簡単に行えます。

・ロボットのステータス（現在位置、アラーム・エラー・サーボ等の状態）のリードや、システムのコントロール（スタート・ホールド、ジョブの呼び出し等）を行います。
・Ｉ／Ｏ信号のリード・ライトを行います。



作業ジョブ自動段取り替え機能

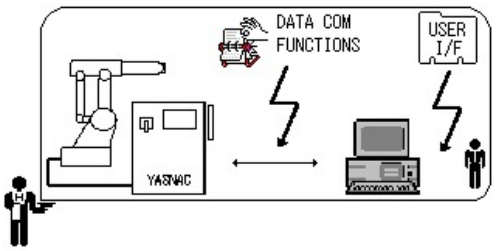
DCI(Data Communication by Instruction) 伝送機能を用いて、簡単に作業ジョブの自動段取り替えを実現します。
ロボットのメモリー不足を補うためにパソコンを外部記憶装置とし、ロボットから伝送してくる作業番号情報に対応する作業ジョブをロボットへ伝送します。



伝送アプリケーション作成手順説明機能

ユーザがロボットとパソコン間の伝送アプリケーションを開発できるようにするため、以下の情報提供を行います。

・ロボットとパソコンとのデータ伝送関数ライブラリを提供します。(Motocom32.dll)
・上記関数を含むサンプルプログラムを用いて、アプリケーションの作成手順を説明します。



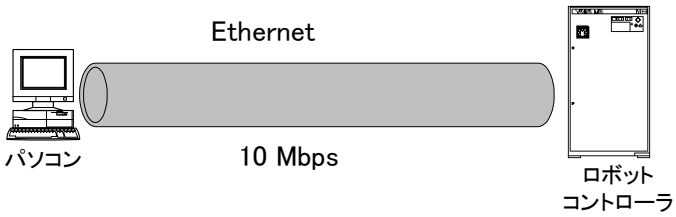
1.2 Ethernet を用いた通信の特徴

Ethernet I/F 基板と本製品 MOTOCOM32 の Ethernet 機能を使用することにより、従来のデータ伝送機能をさらに高速に実現することができます。以下に Ethernet の特徴を示します。

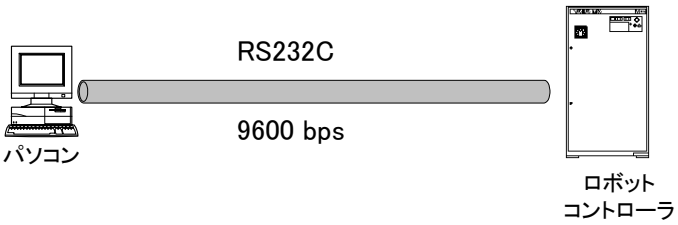
■ 高速な伝送速度

RS-232C を使用した場合に比べ、高速な通信が可能です。

（Ethernetを使用した場合）



（RS-232Cを使用した場合）



重要 ここで表記している伝送速度は、単純に通信媒体上の通信速度のことであり、通信を行うデータのフォーマットチェックなどの時間は含まれていません。

■ 複数のホスト間での通信

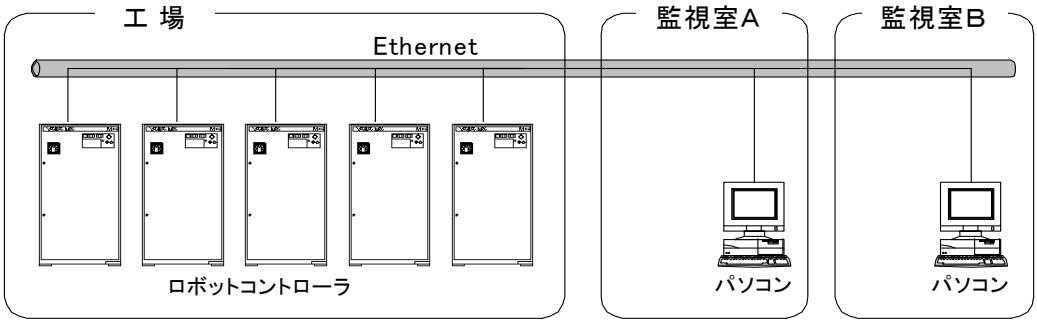
Ethernet では、一本のケーブルで N:N の通信が行えることにより、以下に示すようなシステムの構築が可能となります。

重要 以下の構築例とあわせて「2.4 制限事項」をお読みください。

【構築例１】

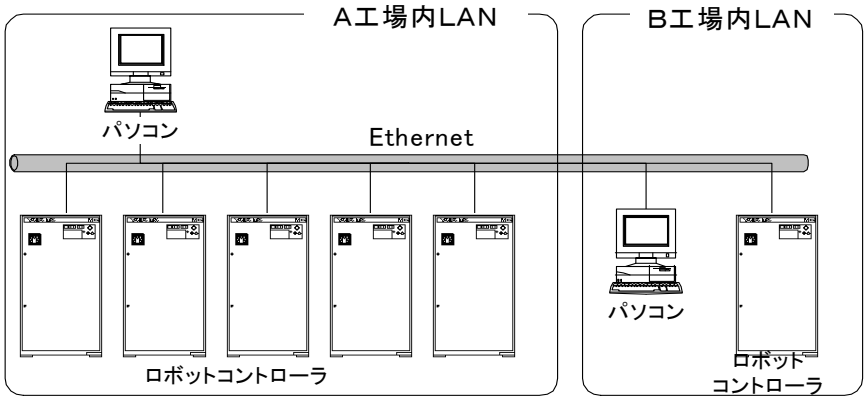
一本の Ethernet ケーブルで複数のネットワーク機器を接続できますので、工場の稼動状況、

アラーム状況などを複数の異なる場所から監視できます。



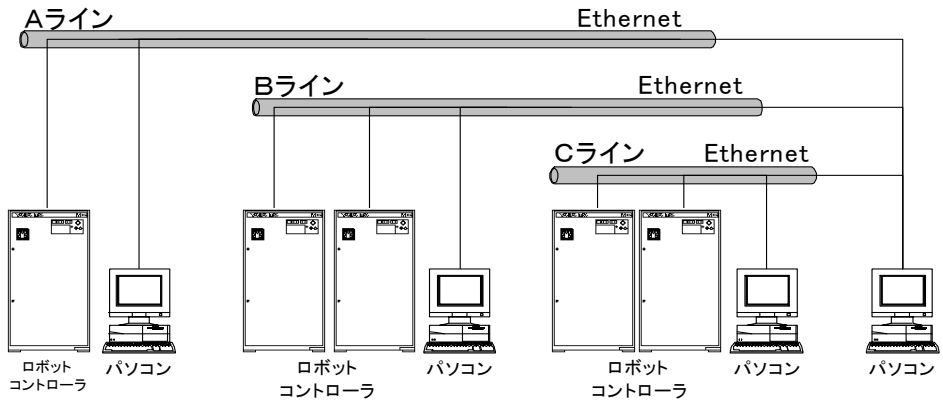
【構築例２】

一本の Ethernet ケーブルで複数工場の LAN を接続し、各工場内での伝送をそれぞれ同時に行なえます。A 工場でのロボットコントローラとパソコン間の伝送と B 工場でのロボットコントローラとパソコンとの伝送が干渉することはありません。また、異なる工場間でのデータ伝送を行なうことができます。（いずれの場合も、正しく設定されていることが必要です。）



【構築例３】

各生産ライン毎にパソコンを置き、パソコン上のジョブを伝送してロボットコントローラで実行できます。また、全体を監視することも可能で Ethernet で接続されたパソコンの状態監視やデータのバックアップなどができます。



1.3 MOTOCOM32 を実行するのに必要なもの

OS	Microsoft Windows 10 （64 ビット版） Microsoft Windows 7 ServicePack1 （32 ビット版 /64 ビット版） 日本語版 / 英語版のみ対応 *1
メモリ	128M バイト以上
ハードディスク	50M バイト以上
ディスプレイ	MS-Windows がサポートするもの（256 色以上）
ロボット コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、 YASNAC XRC、MRC、MRC2、ERC、ERC2
伝送ケーブル	RS-232C ケーブルまたは Ethernet ケーブル
ハードウェア キー	シングルユーザ環境で使用。 詳細については、「1.4 ハードウェアキーについて」を参照してください。

1 MS-Windows10、MS-Windows 7 は、Microsoft Coporation の米国およびその他の国における登録商標です。

重要

- 本パッケージにはコントローラのデータ伝送機能、Ethernet機能、Ethernet基板、伝送ケーブル、パーソナルコンピュータおよびＯＳ は含まれておりません。
- 伝送経路がEthernetケーブル、RS-232Cケーブルのいずれになるかは、ロボットコントローラに設定されているデータ伝送機能仕様に依存します。このソフトウェアを動作させる前に、通信したいコントローラのハード及びソフトウェアの仕様の確認を行なってください。
- YASNAC MRC2,、ERC、ERC2ではEthernetを用いた伝送はできません。（ロボットコントローラが機能をサポートしていないため）また、MRCについては、Ver 4.111以降でのみ対応可能です。
- YRC1000microではRS-232Cケーブルを用いた伝送はできません。（ロボットコントローラが機能をサポートしていないため)）
- ユーザが伝送アプリケーションを作成する場合は、作成ツールMicrosoft Visual Basic Ver6.0以上または、Microsoft Visual C++ Ver6.0以上が必要です。

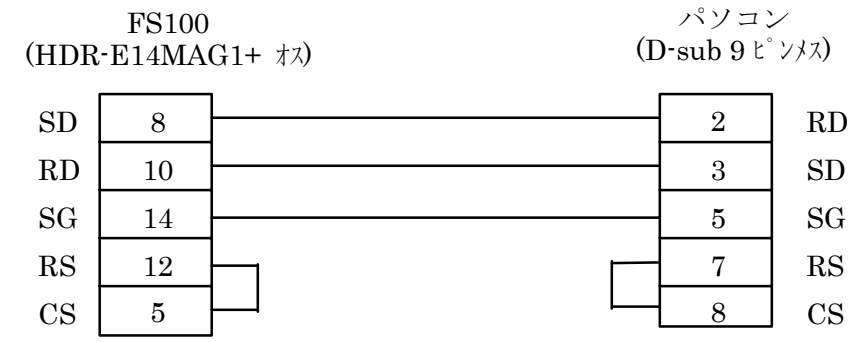
なお伝送ケーブルの接続及び必要になる機器の詳細については、ロボットコントローラの『データ伝送機能 操作説明書』および『Ethernet I/F 基板 取扱説明書』(YRC1000、YRC1000micro、DX200、DX100、FS100、NX100 の場合は『Ethernet 機能説明書』)、MS-Windows のマニュアルなどをあわせてご参照ください。

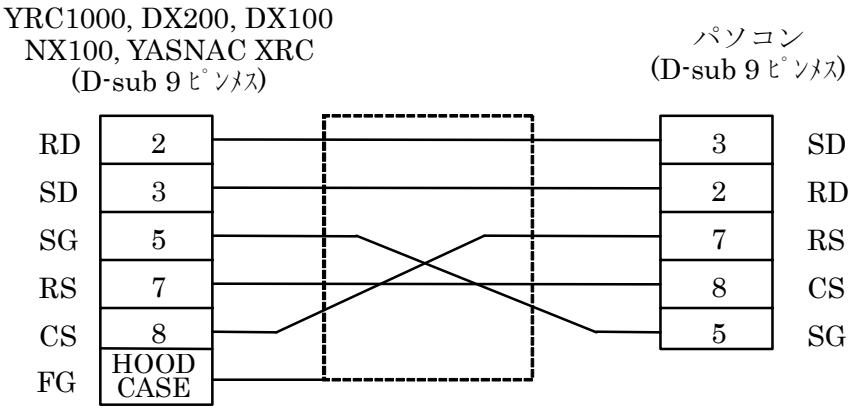
1.3.1 RS-232C 伝送ケーブル仕様

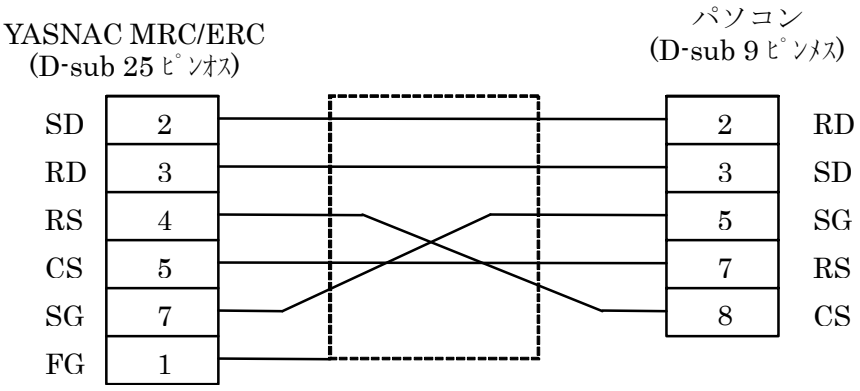
RS-232C 伝送ケーブル仕様は、以下のようになります。

重要 Ethernet を使用する場合、本ケーブルは不要です。

【PC/AT 版】







1.4 ハードウェアキーについて

本ソフトウェアを正常に動かすには、必ずパッケージに同封されているハードウェアキー（USB タイプ）をパソコンに取り付けてから使用してください。

キーをパソコンへ取り付ける前に必ず以下の《パソコンに関する確認》《ドライバのインストール》を行い、その後、キーを USB ポートに取り付けます。

《パソコンに関する確認》

USB タイプキーは、ハードウェアの構造上１ USB ポートに対してキーを連結して複数接続することができません。従って、１ USB ポートにキー１つを接続していただくことになります。

同一のパソコンに複数のオフラインソフトウェアを導入し、USB キーを複数接続する場合は、そのソフトウェア数の USB ポートが準備されたパソコンをご使用ください。

《ドライバのインストール》

重要

必ず、パソコンに取り付けている全ての Sentinel のハードウェアキーを取り外した状態でドライバをインストールしてください。

インストールは、インストール CD-ROM の「\SentinelDriver\Sentinel System Driver Installer 7.6.0.exe」を実行します。インストールの詳細については、「\SentinelDriver\Manual\Sentinel System Driver ReadMe.pdf」をご覧ください。

重要

- 必ずドライバのインストールが必要です。
- ドライバのインストールを行う際には、必ず管理者権限でログオンしてください。
- ドライバをインストールする前にキーをパソコンに取り付けると「デバイス ドライバー ソフトウェアは正しくインストールされませんでした。」が表示されます。
  この場合は、キーをパソコンより取り外して、ドライバをインストールしてください。
  Windows Vista/7以外の場合は、Windows ウィザード（新しいハードウェアの追加ウィザード）が立ち上がります。この場合は、［キャンセル］し、キーをパソコンより取り外して、ドライバをインストールしてください。
- Windows NT4.0、2000にドライバをインストールする場合は、インストールCD-ROMの「\SentinelDriver\SSD5411\SSD5411-32bit.EXE」を使ってインストールしてください。インストール手順は、「\SentinelDriver\SSD5411\Manual\jp\SystemDriver_Installation_J_Internet.pdf」をご覧ください。

その他のハードウェアロックキーに関する対処方法については、「8.1 よくあるお問い合わせ」を参照してください。

2 インストールの方法

2.1 MOTOCOM32 のセットアップ

1. 全てのアプリケーションを終了します。

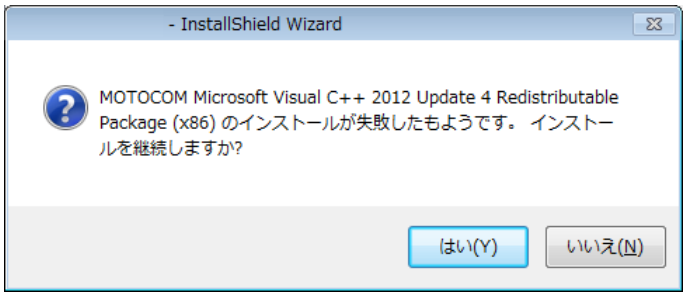
重要

Windows 7、Windows 10 にインストールする際は、必ず管理者権限でログオンしてください。管理者権限でログオンしていない場合に Windows の System 関係の DLL ファイルが更新されない場合があります。

2. インストール CD を CD-ROM ドライブに挿入すると、[MOTOCOM32-InstallShieldWizard] が自動的に表示されます。
   Windows 7 、Windows 10 をご使用の場合は、[ユーザーアカウント制御] ダイアログが表示されるので、[はい] をクリックします。
3. 画面に表示されている説明に従って操作します。

重要

- Windows10の場合は［完了］後、「VB6.0 comm control」をインストール します。画面に表示される説明に従って操作してください。
- インストール途中で以下のメッセージが表示された場合は、［はい］を押し、インストールを継続してください。
  このメッセージは、Visual C++ 2012 の再配布パッケージのインストールが実行済みではあるがインストールが必要な場合に表示されます。メッセージが表示された場合は、MOTOCOM32のインストール後にインストール CDの
  \ISSetupPrerequisites\{BF2F04CD-3D1F-444e-8960-D08EBD285C3F}\vcredist_x86.exe
  を手動にて実行して Visual C++ 2012 の再配布パッケージのインストールまたは、修復を行ってください。



4. セットアップが終了すると［プログラム］－［Motoman］－［MOTOCOM32］グループ内に［High Speed JobExchanger］、［作業ジョブ自動段取り替え］、［ﾎｽﾄｺﾝﾄﾛｰﾙ］が登録されます。（Windows10 の場合は、［Motoman］直下に表示されます。）
5. ハードウェア ロックのドライバをインストールしてからパソコンに接続します。詳細については、「1.4 ハードウェアキーについて」を参照してください。

2.2 ロボットコントローラの環境設定

MOTOCOM32 を用いてコントローラと伝送を行なう場合には、RS-232C、Ethernet のいずれを使用する場合にも以下のロボットコントローラの環境設定が必要になります。

■ パラメータの設定

ロボットコントローラとパソコンの伝送をできるようにするには、ロボットコントローラのいくつかのパラメータを設定する必要があります。

■ 伝送プロトコル指定

RS000 パラメータ: 標準ポート 1 プロトコル指定

RS001 パラメータ: 標準ポート 2 プロトコル指定（MRC のみ）

RS000 / 001 パラメータの値と意味
- 0：未使用
- 1：システム予約
- 2：BSC LIKE（データ伝送機能）
- 3：FC1 プロトコル

これらのパラメータは、ロボットコントローラの標準ポート 1、標準ポート 2 または、Ethernet ボードの伝送プロトコルを指定するために使われます。もし、Ethernet 通信機能が使用されないのであれば、RS000、RS001 は、それぞれ標準ポート 1、標準ポート 2 に該当します。
Ethernet 通信機能＋標準ポート 1 または、標準ポート 2 が使用される場合、そのポート番号に従ったパラメータが設定されなければなりません。それ以外のパラメータは、Ethernet 通信のために使われます。
MOTOCOM32 を使うためには、RS000 または、RS001 のどちらかの値を「2」に設定する必要があります。例えば、ポート 1 が既に FC1 または、FC2 で使用されていて、RS000 が「3」に設定されているのであれば、RS001 は、MOTOCOM32 を使うために「2」を設定する必要があります。

重要 RS000、RS001 パラメータは、同じ値を設定することはできません。Ethernet 通信は、BSC LIKE プロトコルのみをサポートしています。

■ コマンドリモートの設定

ホストコントロール機能を用いて伝送を行なうには、コマンドリモートを有効にする必要があります。また、DCI 機能、スタンドアローン機能を使用する場合にはコマンドリモートを無効にしなければなりません。

MRC, ERC の場合

プログラミングペンダントを用いて、メンテナンスモードからカストマオプションを設定します。

メンテナンスモードの操作法は、各コントローラの操作マニュアルを参照してください。
コマンドリモートを有効にするには、カストマオプションを以下のように設定します。

カストマオプション

I/O ＝ 使用しない

コマンドモード ＝ 使用する

PP/PBOX ＝ 使用しない

YRC1000、YRC1000micro、DX200、DX100、FS100、NX100, XRC の場合

プログラミングペンダントの「入出力」→「擬似入力」より「コマンドリモート選択」を設定します。（●：有効、○：無効）

これらの環境設定の詳細については、以下のマニュアルを参照してください。

『YRC1000 データ伝送機能操作説明書』

『YRC1000micro データ伝送機能操作説明書』

『DX200 データ伝送機能操作説明書』

『DX100 データ伝送機能操作説明書』

『FS100 データ伝送機能操作説明書』

『NX100 データ伝送機能操作説明書』

『YASNAC XRC データ伝送機能操作説明書』

『YASNAC MRC データ伝送機能操作説明書』

『YASNAC ERC データ伝送機能ユーザーズマニュアル』

2.3 Ethernet を使用する場合の環境設定

MOTOCOM32 で Ethernet を使用して伝送をする場合には、以下の環境設定が必要になります。RS-232C ケーブルを使用して伝送を行う場合には、この環境設定を行う必要はありません。各アプリケーションにて「RS-232C 伝送パラメータ」を設定してください。

2.3.1 MOTOCOM32 に含まれる各アプリケーションの設定

■ パラメータの設定

ロボットコントローラと伝送するために IP アドレスなどの伝送パラメータを各アプリケーションで設定する必要があります。

2.3.2 パソコンの設定

Ethernet を使用して伝送をする場合には、インストールを行ったパソコンに伝送に関する設定を行う必要があります。

■ ハードウェアの設定

MOTOCOM32 を使用する前にパソコン本体に Ethernet ボードを接続し、Ethernet ボードが正常に動作することを確認してください。
接続方法は、ご使用になる Ethernet ボードの説明書をお読みください。

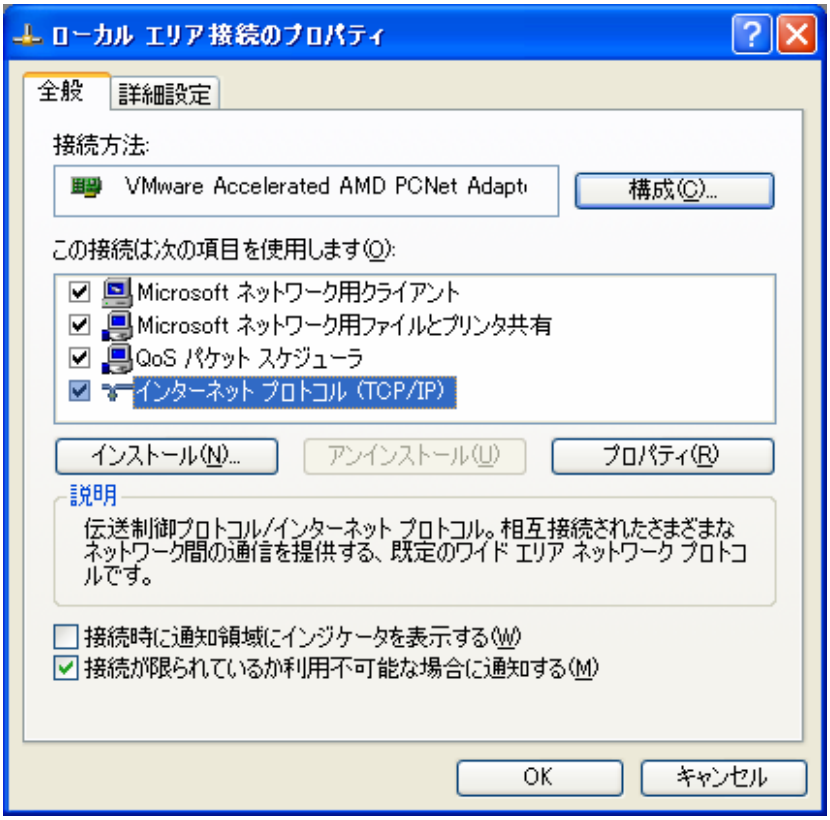
■ Windows のネットワーク設定

Ethernet で伝送を行うためには、Windows のネットワークに関する設定を行います。(以下の例は、Windows XP での設定を例に説明しています。)

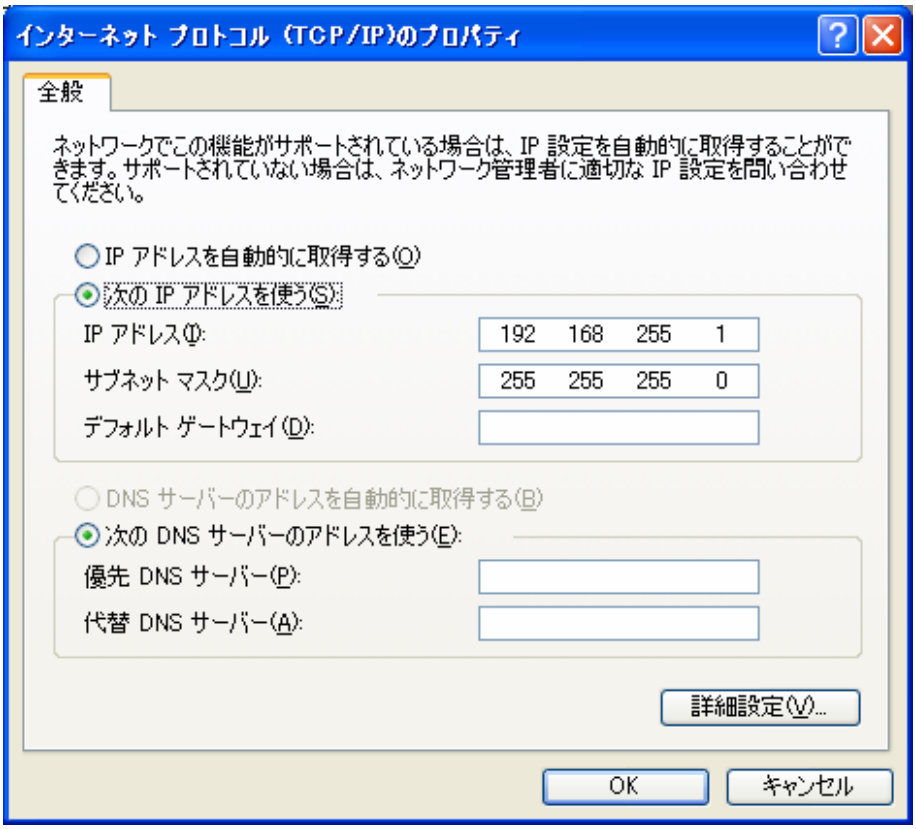
1. タスクバーの [スタート] ボタンをクリックし、【設定】から【コントロールパネル】を開きます。【コントロールパネル】がカテゴリ表示の場合は、[ネットワークとインターネット接続] のカテゴリを選択し、そこから [ネットワーク接続] を選択します。【コントロールパネル】がクラシック表示の場合は、[ネットワーク接続] を選択します。

2. 伝送に使用するネットワーク接続に対して、右クリックメニューから「プロパティ」を選択します。

3. パソコンの IP アドレス、サブネットマスクを設定します。プロパティダイアログで「インターネットプロトコル (TCP/IP)」を選択し、[プロパティ] ボタンをクリックします。



4. [インターネットプロトコ (TCP/IP)] ダイアログで IP アドレス、サブネットマスクを設定します。デフォルトゲートウェイ、DNS サーバなどの詳細については、Windows の説明書をお読みになり、ご使用になる環境に合わせて適切な設定を行ってください。



重要

IP アドレス、サブネットマスクを設定する際には、ネットワーク管理者の指示に従い、正しい数値を入力してください。同一の IP アドレスを複数の機器に設定するなどの誤った設定を行った場合には伝送が正常に行えなくなります。

2.3.3 ロボットコントローラの設定

■ ハードウェアの設定

Ethernet による伝送を行う場合には、YRC1000 用 ACP01 基板、YRC1000micro 用 ACP30 基板、DX200 用 YCP21 基板、DX100 用 YCP01 基板、FS100 用 CPU201R 基板、NX100 用 NCP01 基板の Ethernet コネクタ、または XRC 用／MRC 用 Ethernet I/F 基板を使用する必要があります。以下の説明書をよく読み、IP アドレス、サブネットマスクを設定してください。

『YRC1000 Ethernet 機能説明書』
『YRC1000micro Ethernet 機能説明書』
『DX200 Ethernet 機能説明書』
『DX100 Ethernet 機能説明書』
『FS100 Ethernet 機能説明書』
『NX100 Ethernet 機能説明書』
『YASNAC XRC Ethernet I/F 基板 取扱説明書』

『YASNAC MRC Ethernet I/F 基板 取扱説明書』

■ 通信パラメータの設定

コントローラで使用する IP アドレスなどの通信パラメータの設定をプログラミングペンダントを用いてメンテナンスモードで行ないます。メンテナンスモードの操作法は、各コントローラの操作マニュアルを参照してください。

Ethernet

Ethernet ＝ 使用する
IP アドレス ＝ 192.168.10.10(*)
サブネットマスタ ＝ 255.255.255.0(*)
デフォルトゲートウェイ ＝ 192.168.10.1(*)
サーバーアドレス ＝ 0.0.0.0(*)
(*) は使用例です。ネットワーク環境にあわせて適切な値を設定してください。
ホストコントロール機能を使用する場合は、サーバーアドレスを使用する必要はありません。これは、DCI 機能やスタンドアローン機能の場合に使用されます。これらの情報については、以下のマニュアルを参照してください。

『YRC1000 Ethernet 機能説明書』
『YRC1000micro Ethernet 機能説明書』
『DX200 Ethernet 機能説明書』
『DX100 Ethernet 機能説明書』
『FS100 Ethernet 機能説明書』
『NX100 Ethernet 機能説明書』
『YASNAC XRC Ethernet I/F 基板 取扱説明書』
『YASNAC MRC Ethernet I/F 基板 取扱説明書』

2.3.4 ネットワークの構成

Ethernet を使用して、ロボットコントローラと伝送を行うためには、ネットワークを適切に構成する必要があります。
ネットワークの詳細な構成方法については、

『YRC1000 Ethernet 機能説明書』
『YRC1000micro Ethernet 機能説明書』
『DX200 Ethernet 機能説明書』
『DX100 Ethernet 機能説明書』
『FS100 Ethernet 機能説明書』
『NX100 Ethernet 機能説明書』
『YASNAC XRC Ethernet I/F 基板 取扱説明書』
『YASNAC MRC Ethernet I/F 基板 取扱説明書』

を参照してください。

2.3.5 High Speed Link Server

High Speed Link Server は、Ethernet で伝送を行うときにに使用します。RS-232C の場合は、使用しません。

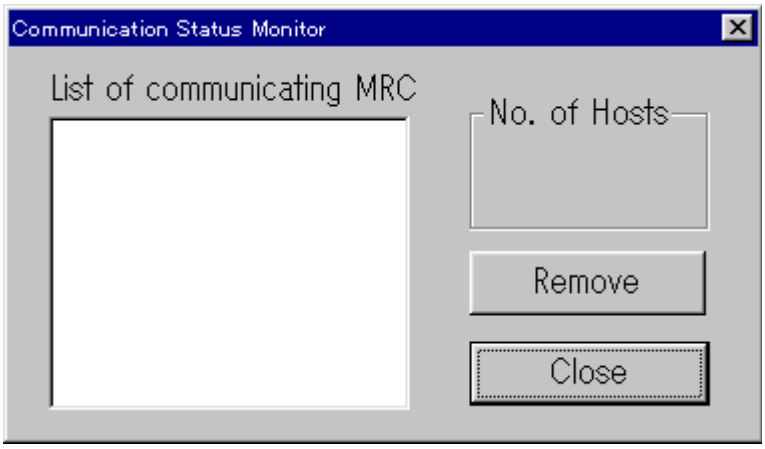
以下に、High Speed Link Server について説明します。

■ Server について

Ethernet でロボットコントローラと通信するためには、サーバプログラム（HSLSR32.EXE）が必要です。
このサーバは、MOTOCOM32 を使用するアプリケーションからの要求でデータの送受信を行い、アプリケーションとロボットコントローラが Ethernet で伝送を始める時に、サーバプログラムは自動で起動されます。
サーバプログラムを終了する方法は、タスクバーの［High Speed Link Server］にマウスを移動し、右ボタンクリックします。メニューが表示されるので、［閉じる］を選択し、サーバプログラムを終了します。
パソコンとロボットコントローラと通信するためには、サーバプログラムと関連する DLL ファイルがアプリケーションと同じフォルダに入っている必要があります。
サーバプログラム以外のアプリケーション (MOTOCOM32 を使用するアプリケーションを含む) の優先度がサーバプログラムより高いと、データの送受信が正しく行われないことがあります。
サーバプログラムの優先度は、他のアプリケーションと同じか高くなるようにする必要があります。

■ 伝送ステータスモニタ

High Speed JobExchager の立ち上がり時に伝送ステータスモニタを表示します。



何らかの理由（例えば、別の操作の割り込みをかける場合）で伝送ステータスモニタを隠す場合は、タスクバーの［High Speed Link server］を選択し、マウスの右のボタンをクリックします。メニューが表示されるので、［Monitoring］を選択することで、伝送ステータスモニタの表示の切り換えができます。

■ 伝送情報の削除

Ethernet で伝送を行い、伝送の誤りが以下の何れかの理由によって発生した場合の通信情報は、ロボットコントローラとパソコンの両方またはどちらかに残る場合があります。
伝送がうまくいかなかった原因には、以下の２つがあります。

(a) Ethernet ケーブルが、伝送中に取り除かれた場合。
(b) ロボットコントローラの PLAYBACK BOX の［REMOVE］ボタンが、伝送中にオフになった場合。

伝送が続けられない時は、以下の手順で伝送情報をリセットします。
(ロボットコントローラ側)

(1) プログラミングペンダントにエラーメッセージが、表示されている時に［RESET］ボタンを押します。
(2) 一度、PLAYBACK BOX の［REMOVE］ボタンをオフにして、再びオンにします。

(パソコン側)

(1) 伝送ステータスモニタの［Remove］ボタンを押します。
(2) 情報をリセットするために、エラーが発生したロボットコントローラの IP アドレスをインプットし、［OK］ボタンを押します。

2.4 制限事項

MOTOCOM32 を使用するにあたり、以下の点に注意してご利用ください。

2.4.1 ロボットコントローラ／パソコン共通

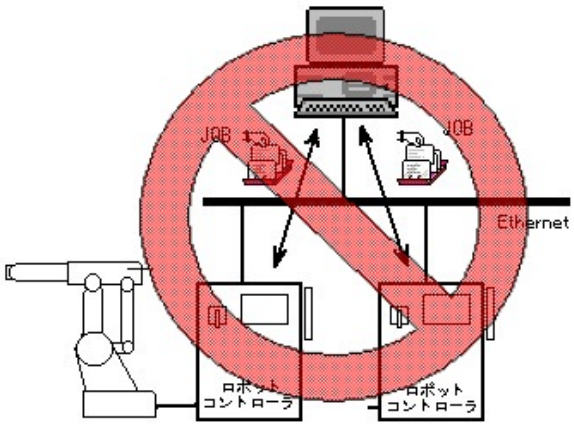
■ TCP/IP での使用ポート

MOTOCOM32 では通信のプロトコル（伝送規約）として TCP/IP を使用しています。
TCP/IP で通信をするためには、内部で「ポート番号」と呼ばれるサービス識別番号を使用しますが、MOTOCOM32 ではデータ伝送のために 10000 ～ 10008 のポート番号を利用します。この番号が他のネットワーク機器と重複している場合は、正常に通信が行なえなくなります。MOTOCOM32 をご使用になる場合は、同一ネットワーク上に上記の範囲のポート番号を使用するネットワーク機器が存在しないことをあらかじめ確認してください。

2.4.2 パソコン

■ 同一ファイルへのアクセス

パソコン内の同一ファイルを異なるロボットコントローラから同時にアクセスすることはできません。（同時でなければ問題ありません。）



■ 同一 Window ハンドルの使用禁止

Ethernet 通信を実施する場合、BscSetEther() 関数の引数として Window のハンドルを指定します。この Window のハンドルは実態のある Window ハンドルである必要があります。また複数のロボットに対し、同一の Window ハンドルを指定することはできません。

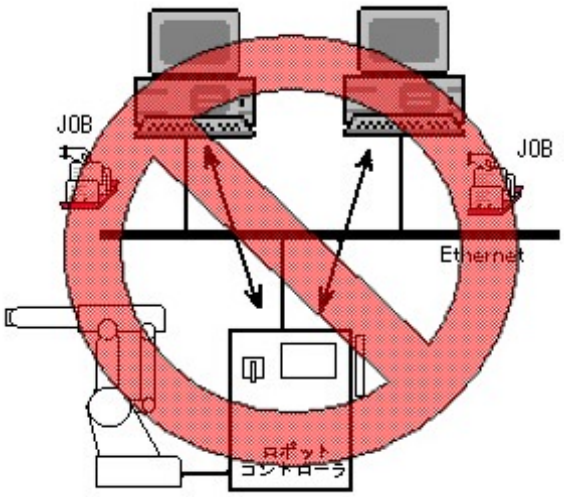
■ マルチスレッドでの使用禁止

Ethernet 通信を実施する場合、MOTOCOM32 の関数は必ずメインスレッドの中で実行してください。

2.4.3 ロボットコントローラ

■ 複数のパソコンからのアクセス

MOTOCOM32 を使用する場合、1 台のロボットコントローラにつき、1 台のパソコンだけが伝送できます。<u>同時に複数のパソコンとの伝送を行なうことはできません。</u>
（1 台のパソコンと複数のロボットコントローラが、同時に伝送を行なうことは可能です。）
YRC1000、YRC1000micro、DX200、DX100、FS100、NX100 では「イーサネットサーバ機能」を用いて接続することで、1 台のコントローラで複数のパソコンとの接続を管理することができます。



■ CMOS 一括などの保存

ロボットコントローラには、外部機器と伝送を行なうプロトコル（伝送規約）として BSC 準拠プロトコルと FC1 プロトコルがあります。MOTOCOM32 では BSC 準拠プロトコルを使用して伝送を行ないます。CMOS 一括の保存などは FC1 プロトコルを利用するため MOTOCOM32 では実行できません。CMOS 一括の保存などを行なう場合には、YASNAC FC1 / FC2 または、PC カード / CF への保存をご利用ください。

2.4.4 文字列変数送受信のサポート

「文字列変数の通信」は、YRC1000、YRC1000micro、DX200、DX100、FS100、NX100 バージョン NS3.00-00 以降でのみサポートされています。「文字列変数の通信」を行うためには、MOTOCOM32 バージョン 3.10 以降をご使用いただく必要があります。

2.5 MOTOCOM32 各プログラムの実行

MOTOCOM32 のプログラムは

「High Speed JobExchanger」

「ホストコントロール」

「作業ジョブ自動段取り替え」

の 3 つのプログラムからなっています。各プログラムを実行する場合は、スタートメニューからを実行したいアプリケーションを選択してください。

3 High Speed JobExchanger 機能

3.1 High Speed JobExchanger とは

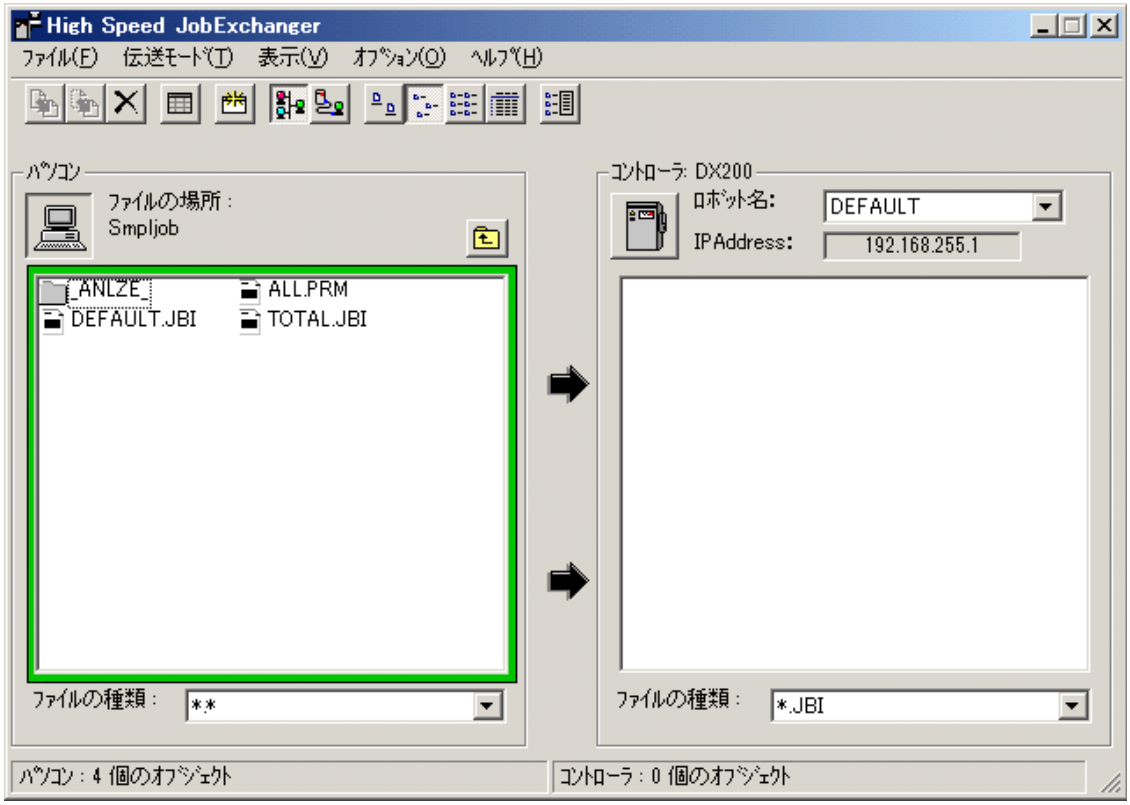
High Speed JobExchanger（ハイ スピード ジョブ イクスチェンジャー）は、ロボットコントローラとパソコンを Ethernet ケーブルまたは、RS232C ケーブルで接続し、パソコンからの指令によって以下のファイルのロード・セーブを行うアプリケーションソフトウェアです。ファイルデータ伝送機能がサポートしている伝送コマンドは以下のとおりです。

ジョブの伝送	UPLOAD/ DOWNLOAD	単独ジョブ
		関連ジョブ
各種条件データ・システム情報ファイルの伝送	UPLOAD/ DOWNLOAD	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC、ERC のサポートする各種条件データファイルの伝送

重要

- High Speed JobExchanger を使用してロボットコントローラとの通信を行うには、コントローラの設定を「コマンドリモート有効」に設定しておく必要があります。
- Ethernet で伝送を行う場合のロボットコントローラは、YRC1000、YRC1000micro, DX200, DX100, FS100, NX100, YASNAC XRC / MRC のみとなります。YASNAC MRC2,ERC,ERC2 は、Ethernet での伝送は行えません。

3.2 画面の操作


High Speed JobExchanger
ファイル(F) 伝送ﾓｰﾄﾞ(T) 表示(V) ｵﾌﾟｼｮﾝ(O) ﾍﾙﾌﾟ(H)
ﾊﾟｿｺﾝ
ファイルの場所：
Smpljob
ANLZE
ALL.PRM
DEFAULT.JBI
TOTAL.JBI
ファイルの種類：
**
ｺﾝﾄﾛｰﾗ: DX200
ﾛﾎﾞｯﾄ名:
DEFAULT
IPAddress:
192.168.255.1
ファイルの種類：
*.JBI
ﾊﾟｿｺﾝ：4 個のｵﾌﾞｼﾞｪｸﾄ
ｺﾝﾄﾛｰﾗ：0 個のｵﾌﾞｼﾞｪｸﾄ

3.2.1 メニュー

■ ［ファイル］メニュー

コピー	ロボットコントローラからパソコンへ、またはパソコンからロボットコントローラへファイルをコピーします。
移動	ロボットコントローラからパソコンへ、またはパソコンからロボットコントローラへファイルを移動します。
削除	ロボットコントローラまたは、パソコンのファイルを削除します。
全て選択	現在、選択されているウィンドウの全てのファイルを選択します。
ファイル内容の表示	選択したファイルの内容をコントローラ側のウィンドウに表示します。
ファイル内容の印刷	選択したファイルの内容をコントローラ側のウィンドウに表示し、印刷します。
High Speed JobExchanger の終了	High Speed JobExchanger を終了します。 【オプション】－【環境設定】で「今回の設定値を保存しない」が選択されていなければ、現在の設定状況が初期ファイルに保存され、次回起動時には同じ状況で使い始めることができます。

■ ［伝送モード］メニュー

Ethernet 伝送	Ethernet 経由で伝送を行います。
RS232C 伝送	RS232C ケーブル経由で伝送を行います。

■ ［表示］メニュー

大きいアイコン	現在選択されている対象（パソコン、またはロボットコントローラ）のファイルリストの表示形式を大きいアイコンで表示します。
小さいアイコン	現在選択されている対象（パソコン、またはロボットコントローラ）のファイルリストの表示形式を小さいアイコンで表示します。
一覧	現在選択されている対象（パソコン、またはロボットコントローラ）のファイルリストの表示形式を一覧形式で表示します。
詳細	パソコン側のファイルリストの表示形式を詳細で表示します。 詳細表示では、「ファイル名」「ファイルのサイズ」「更新日時」を表示します。 重要 詳細表示ができるのは、パソコン側のファイルリストのみです。コントローラ側のファイルリストは詳細表示できません。

アイコンの整列	現在選択されている対象（パソコン、またはロボットコントローラ）のファイルリストに表示されているファイル（アイコン）を並べ替えます。並べ替えは、「名前順」「サイズ順」「時間順」のいずれかで行うことができます。 ファイルリストが「詳細」で表示されている場合、見出しの部分をクリックすることで並べ替えを行うこともできます。
最新の情報に更新	現在選択されている対象（パソコン、またはロボットコントローラ）のファイルリストを最新の情報に更新します。 以下のような操作を行なった場合には、必ず【最新の情報に更新】を選択して下さい。 ・High Speed JobExchanger 以外のアプリケーションで、パソコン側ファイルのコピー、移動、削除を行なった場合 ・コントローラ側のファイルを、FC1/FC2 などの外部記憶装置を使用して移動、削除を行なった場合

■ ［オプション］メニュー

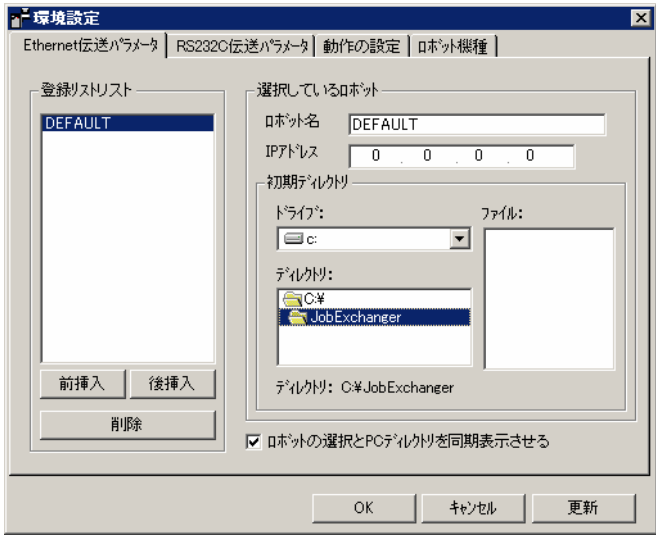
環境設定	High Speed JobExchanger を動作させるためのさまざまな環境の設定を行います。このメニューを選択すると、［環境設定］ダイアログが表示され、このダイアログで各項目の設定を行います。 詳細については、「環境設定」を参照してください。
ジョブの一括アップロード	ジョブを一括でアップロードするときに使用します。 ファイルの名前を入力することによって、複数ファイルのアップロードが行えます。詳細については、「3.3.9 ジョブの一括アップ／ダウンロード」を参照して下さい。
ジョブの一括ダウンロード	ジョブを一括でダウンロードするときに使用します。 ファイルの名前を入力することによって、複数ファイルのダウンロードが行えます。詳細については、「3.3.9 ジョブの一括アップ／ダウンロード」を参照してください。
言語選択	High Speed JobExchanger で表示する言語を選択します。 詳細については、「言語選択」を参照してください。

環境設定

High Speed JobExchanger を動作させるためのさまざまな環境の設定を行います。

Ethernet 伝送パラメータ

Ethernet 伝送パラメータで設定されるロボットに関する情報は、「robot.ini」ファイルに格納され、ロボットコントローラとの伝送に使用されます。
ロボット名称、IP アドレス、初期ディレクトリは、各ロボットごとに登録されます。

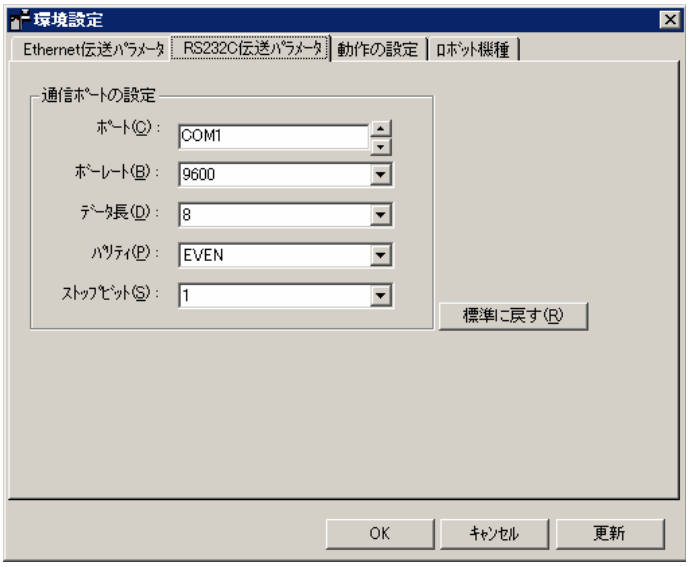


登録リスト	登録されているロボットのロボット名一覧を表示します。 ロボット名をクリックすると、対応する設定値が［選択しているロボット］に表示されます。
前挿入	登録リストより選択しているロボットの前に新規ロボットを挿入します。
後挿入	登録リストより選択しているロボットの後に新規ロボットを挿入します。
削除	登録リストより選択しているロボットを削除します。 重要 現在、使用しているロボットは、削除できません。
ロボット名	ロボットを識別するための名前を入力します。
IP アドレス	Ethernet で伝送をするために必要な IP アドレスを入力します。 この値は、伝送相手となるロボットコントローラに設定した値と同じ値を入力します。
初期ディレクトリ	各ロボットの初期ディレクトリを設定します。 伝送するロボットを変更した場合、初期ディレクトリで設定したディレクトリが最初に開かれます。
ロボットの選択とPCディレクトリを同期表示させる	この項目を選択した場合、コントローラ側の［ロボット名］を変更するとパソコン側のファイルリストの表示が設定したロボット名の［初期ディレクトリ］のディレクトリに切り替わります。

RS232C 伝送パラメータ

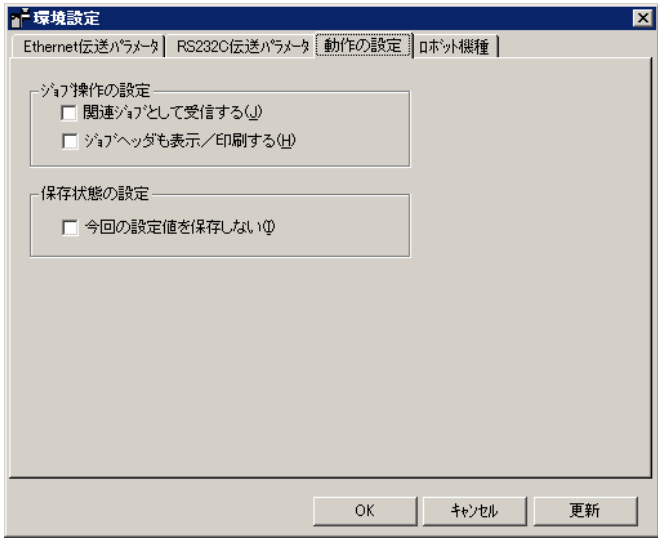
RS232C 伝送パラメータで、ロボットコントローラとの伝送の通信ポートの設定を行います。ポート、ポートレート、データ長、パリティ、ストップビットをロボットコントローラの設定に合わせて選択します。

［標準に戻す］ボタンを選択すると、デフォルトのデータが設定されます。



通信ポートの設定	シリアル通信を行うための通信パラメータを設定します。 ロボットコントローラとの伝送は、ここで設定したパラメータで行われます。
標準に戻す	通信ポートの設定値をデフォルトの状態に戻します。

動作の設定



関連ジョブとして受信する	この項目を選択すると、ロボットコントローラからのジョブを関連ジョブとして受信します。
ジョブヘッダも表示／印刷する	この項目を選択すると、ジョブを表示または、印刷する際に、ジョブヘッダを表示または、印刷します。 重要　関連ジョブや条件ファイルは、このパラメータの対象となりません。

今回の設定値を保存しない	通常、プログラムの終了時に「伝送パラメータ」「ロボット機種」「ジョブ操作の設定」「ウィンドウの大きさ／位置の情報」を初期値ファイルとして自動的に保管し、次回の起動時に反映させます。 環境を一時的に変更したなどの理由で次回の起動に影響を及ぼしたくない場合は、この項目を選択します。

ロボット機種

コントローラの設定	High Speed JobExchanger と接続するロボットコントローラを「YRC1000」「YRC1000micro」「DX200」「DX100」「FS100」「NX100」「XRC」「MRC」「ERC」より選択します。

言語選択

High Speed JobExchanger で表示する言語を選択します。
各言語の修正や変更、新規の言語の作成も行えます。詳細については、「3.4 言語ファイルの編集」を参照してください。

プログラム開始時に言語選択を行なわない	本項目をチェックすると、次回から起動時にこのダイアログを表示しません。また、チェックをはずすと、起動するたびに［言語選択］ダイアログが表示され、言語切り替えが簡単に行えます。

重要 言語によっては正しいフォントの設定を行なわないと、うまく字が表されない場合があります。

■ ［ヘルプ］メニュー

トピックの検索	各機能についてのヘルプを表示します。
バージョン情報	製品のバージョンを表示します。 この表示を閉じる場合は、［OK］ボタンを押します。

3.2.2 ツールバー

以下に、ツールバーについての説明をします。

	ロボットコントローラからパソコンへ、またはパソコンからロボットコントローラへファイルをコピーします。 メニュー：【ファイル】－【コピー】
	ロボットコントローラからパソコンへ、またはパソコンからロボットコントローラへファイルを移動します。 メニュー：【ファイル】－【移動】
	ロボットコントローラ、またはパソコンのファイルを削除します。 メニュー：【ファイル】－【削除】
	現在、選択されているウィンドウの全てのファイルを選択します。 メニュー：【ファイル】－【全ての選択】
	ファイルリストを最新の情報に更新します。 メニュー：【表示】－【最新の情報に更新】
	Ethernet 経由での伝送を行います。 メニュー：【伝送モード】－【Ethernet 伝送】
	RS232C ケーブル経由での伝送を行います。 メニュー：【伝送モード】－【RS232C 伝送】
	ファイルリストの表示形式を「大きいアイコン」で表示します。 メニュー：【表示】－【大きいアイコン】

	ファイルリストの表示形式を「小さいアイコン」で表示します。 メニュー：【表示】－【小さいアイコン】
	ファイルリストの表示形式を「一覧形式」で表示します。 メニュー：【表示】－【一覧】
	パソコン側のファイルリストの表示形式を「詳細」で表示するように変更します。 メニュー：【表示】－【詳細】  詳細表示ができるのは、パソコン側のファイルリストのみです。
	選択したファイルの内容をコントローラ側のウィンドウに表示します。 メニュー：【ファイル】－【ファイル内容の表示】

3.3 操作手順

3.3.1 High Speed JobExchanger の起動

1. タスクバー［スタート］ボタンの【プログラム】－【Motoman】－【MOTOCOM32】－【High Speed JobExchanger】（Windows10 の場合：【Motoman】－【High Speed JobExchanger】）を押し、High Speed JobExchanger を起動します。
2. ロボットコントローラに接続するかどうかのメッセージ（「コントローラにジョブ一覧を問い合わせます。よろしいですか？」）が表示されるので、［はい］または、［いいえ］ボタンを押します。［いいえ］を選択した場合は、コントローラに接続しません。
3. High Speed JobExchanger のメイン画面が表示されます。左側ウィンドウにパソコンの情報が表示され、右側ウィンドウには、ロボットコントローラの情報を表示します。

3.3.2 ファイルのコピー

1. コピーするファイルをリストから選択します。（ファイルを複数選択する場合は、「3.3.5 ファイルの複数選択」を参照してください。）
2.  または、【ファイル】－【コピー】を選択します。
3. 画面に表示されるメッセージに従い操作を進めます。

重要

- ロボットコントローラ側のジョブを関連ジョブとして受信する場合は、【オプション】－【環境設定】で「関連ジョブとして受信」をあらかじめ選択しておきます。
- パソコン側の関連ジョブを送信する場合は、拡張子が「JBR」のファイルを選択します。ロボットコントローラに同名のジョブが存在する場合は、事前に削除しておきます。

■ メニュー、ツールバー以外のファイル操作方法

ポップアップメニュー

ファイルリストにマウスを移動し、マウスの右ボタンをクリックすると、ポップアップメニューが表示され、ファイルの操作が行えます。

3.3.3 ファイルの移動

1. 移動するファイルをリストから選択します。（ファイルを複数選択する場合は、「3.3.5 ファイルの複数選択」を参照してください。）
2.  または、【ファイル】－【移動】を選択します。
3. 画面に表示されるメッセージに従い操作を進めます。

重要

- ロボットコントローラ側のジョブを関連ジョブとして受信する場合は、[オプション]－［環境設定］で「関連ジョブとして受信」をあらかじめ選択しておきます。
- ロボットコントローラ側の条件ファイルは、移動することができません。

3.3.4 ファイルの削除

1. 削除するファイルをリストから選択します。(ファイルを複数選択する場合は、「3.3.5 ファイルの複数選択」を参照してください。)
2.  または、【ファイル】－【移動】を選択します。
3. 画面に表示されるメッセージに従い操作を進めます。

ロボットコントローラ側の条件ファイルは、移動することができません。

3.3.5 ファイルの複数選択

ファイルを複数選択するには、以下の５つの方法があります。

a) ファイルリスト内でマウスをドラッグ＆ドロップすると選択枠が表示され、その枠で選択したいファイルを囲みます。囲んだ範囲のファイルが全て選択されます。

b) まず１つのファイルを選択し、[SHIFT] キーを押しながら選択した場所の最後のファイルをマウスでクリックすると、最後のファイルまでがリスト順で選択されます。

c) ファイルを選択した後、[CTRL] キーを押しながら選択したいファイルをマウスでクリックすると、選択ファイルを１つずつ追加することができます。

d)b) と同じ方法で、マウスではなく [SHIFT] キーを押しながらカーソルキーを上下左右に動かすことで実現できます。

e)c) と同じ方法で、マウスではなく [CTRL] キーを押しながらカーソルキーで選択したいファイルにカーソルをあわせて、スペースキーを押すことで実現できます。

3.3.6 ファイル内容の表示

■ ファイル内容の表示

ファイル内容の表示するには、以下の２つの方法があります。ただし、表示できるファイルは、ロボット関連ファイルのみで他のファイルを選択しても表示できません。


ロボット関連ファイルのアイコン


その他のファイルのアイコン

a) 表示するファイルにマウスを移動し、ダブルクリックをすると、Windows のメモ帳を起動し、ファイルの内容を表示します。

b) パソコン側のファイル一覧よりファイルを選択し、  または、【ファイル】－【ファイル内容の表示】を選択すると、コントローラ側 Window にファイルの内容を表示します。
ファイルの内容が表示された状態で、他のファイルを選択すると、選択されたファイルの

内容が表示されます。もう一度  を選択すると、元の画面に戻ります。

重要 ファイルの内容を表示できるのはパソコン側のみで、ロボットコントローラ側のファイルの内容は表示できません。

■ ジョブヘッダ情報の表示

ジョブヘッダ情報を表示する場合は、【オプション】－【環境設定】の［動作の設定］の「ジョブヘッダも表示／印刷する」を選択します。

3.3.7 ファイルの種類の設定

パソコン側の［ファイルの種類］より表示するファイルの種類を設定します。

3.3.8 印刷

1. パソコン側のファイル一覧よりファイルを選択し、  または、、【ファイル】－【ファイル内容の表示】を選択すると、コントローラ側にファイルの内容を表示します。
2.  または、【ファイル】－【ファイル内容の印刷】を選択すると、ファイルを印刷します。

重要
- ファイルの内容を印刷する際にジョブヘッダも表示する場合は、【オプション】－【環境設定】で［ジョブヘッダも表示／印刷］をあらかじめ選択しておきます。
- ファイルの内容を印刷できるのはパソコン側のみで、ロボットコントローラ側のファイルの内容は印刷できません。

3.3.9 ジョブの一括アップ／ダウンロード

ジョブリストを作成すると、複数ファイルのアップロード、ダウンロードを行えます。
ジョブリストは、Windows のメモ帳などを使用して作成して下さい。
ジョブリストの形式は、以下のようになります。

【ジョブリストの形式】
- １行１ファイルとします。
- 大文字、小文字の両方を使用できます。
- 拡張のないファイルは、JBI と見なされます。
- ジョブリストの記述途中に空白行がある場合は、その行を読み飛ばします。

【ジョブリストの例】

```
1.JBI
2.JBI
:          （拡張子を省略すると、拡張子は、JBI となります。）
:
5.jbi
```

■　ジョブの一括アップロード

1. 【オプション】－【ジョブの一括アップロード】を選択すると、［ジョブの一括アップロード］ダイアログを表示します。現在のディレクトリに存在しているジョブのリストファイル名を設定します。

2. ［OK］ボタンを押すとリストファイルに書かれたファイルを１行ずつ順番にアップロードしていきます。

■　ジョブの一括ダウンロード

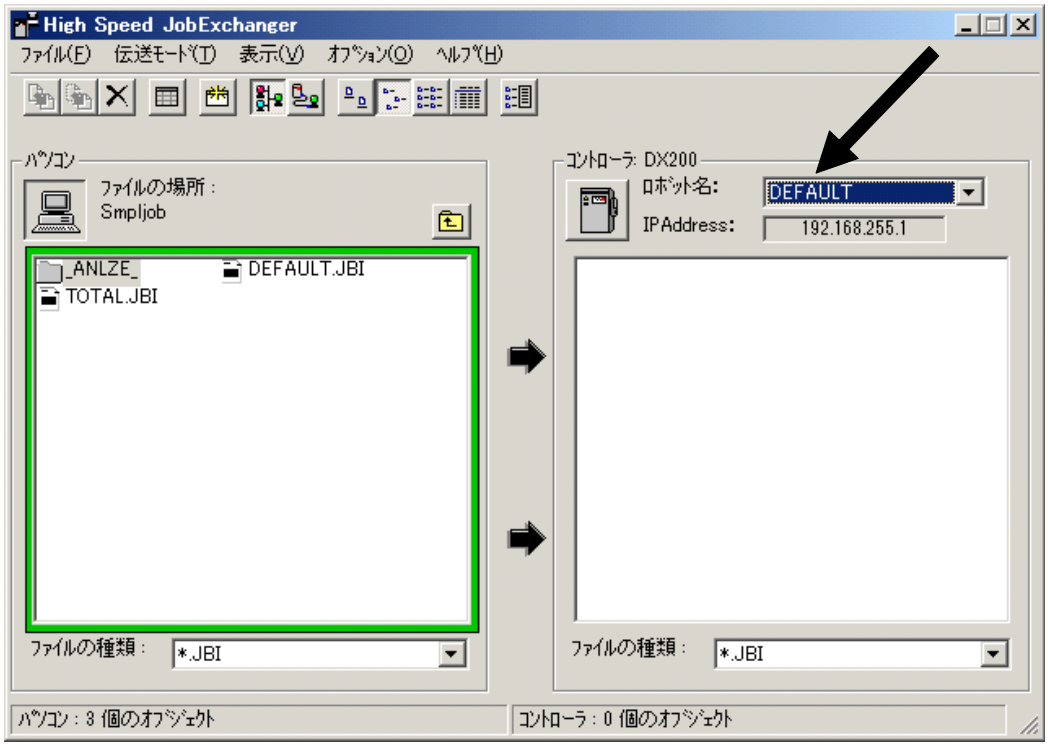
1. 【オプション】－【ジョブの一括ダウンロード】を選択すると、［ジョブの一括ダウンロード］ダイアログを表示します。現在のディレクトリに存在しているジョブのリストファイル名を設定します。

2. ［OK］ボタンを押すとリストファイルに書かれたファイルを１行ずつ順番にダウンロードしていきます。

3.3.10　伝送対象ロボットの設定

Ethernet で伝送する時にコントローラ側の［ロボット名］コンボボックスより、ロボットを

選択し、伝送対象ロボットの切り替えを行います。



重要

- 伝送対象ロボットの設定はEthernetで伝送する時のみ使用できます。
- ロボットの切り替え時にウィンドウは自動時で更新されません。ファイルの操作を行う前に、コントローラ側の［最新の情報に更新］（【表示】メニュー）を行ってください。

3.3.11　伝送可能なファイル

伝送可能なファイルは、以下の通りです。
ただし、「I/O データファイル（.dat .lst）」「カストマデータファイル（.dat .sys）」「パラメータファイル（.prm）」は、ロボットコントローラへのロードまたは、削除はできません。

ジョブファイル（.jbi .jbr）
条件データファイル（.cnd）
汎用データファイル（.dat）
I/O データファイル（.dat .lst）(*)
カストマデータファイル（.dat .sys）(*)
パラメータファイル（.prm）(*)

重要

- システムパラメータのロボットコントローラへのロードは、安全上できないようにしています。（FC1、FC2または、FC1 Emulator for Windowsを使用してください。）
- コントローラのバージョンが、Ethernet機能をサポートしていない場合は、(*)の伝送はできません。

3.3.12 INI ファイル

High Speed JobExchanger には、環境に関する情報の２つの INI ファイルがあります。

JobExchanger.ini	フォント、スタート時の言語、ウィンドウのサイズなどの環境に関する情報を記述しています。
robot.ini	ロボットの登録に関する情報を記述しています。

INI ファイルは MOTOCOM32 をインストールした OS によって格納先が異なります。それぞれ以下の場所に格納しています。

【Windows XP】

両 INI ファイル共に High Speed JobExchanger インストールフォルダに格納しています。

【Windows 7 / 10】

JobExchanger.ini	C:\Users\< ユーザ名 >\Documents\MOTOMAN\MOTOCOM32\HighSpeedJobExchanger" に格納しています。
robot.ini	C:\ProgramData\Motoman\MOTOCOM32\HighSpeedJobExchanger\" に格納しています。

3.3.13 言語ファイル

High Speed JobExchanger では、英語と日本語の２言語をサポートしています。
言語ファイルの編集や新規の言語ファイルを作成することもできます。詳しくは、「3.4 言語ファイルの編集」を参照してください。

3.4 言語ファイルの編集

3.4.1 言語ファイルの編集

［言語選択］ダイアログの［編集］ボタンを押すと、現在、選択している言語の言語ファイルを Windows のメモ帳で開きます。言語ファイルの編集を行います。

重要 言語ファイルは、ソフトウェアの重要な部分なので変更を行う時は十分な注意が必要です。

3.4.2 新規言語ファイルの作成

言語ファイルは、Windows のメモ帳で変更ができます。
以下の手順で新規言語ファイルの作成を行います。
言語ファイルは MOTOCOM32 をインストールした OS によって格納先が異なります。それぞれ以下の場所に格納しています。
【Windows XP】
High Speed JobExchanger インストールフォルダに格納しています。
【Windows 7 / 10】
"C:\ProgramData\Motoman\MOTOCOM32\HighSpeedJobExchanger\" に格納しています。

【例：中国語の場合】
1. 「English.lng」ファイルをコピーし、ファイル名を「User13.lng」にリネームします。

重要 「Chinese.lng」にリネームしてはいけません。

2. 「User13.lng」ファイルの「English」を「Chinese（中国語）」に変更し、「User13.lng」ファイルの各言語を中国語に翻訳します。
3. より多くの言語を作成する場合は、「User14.lng」などを同じ手順で作成します。

重要 新規で作成された言語は、「システム」言語では表示されません。「User13」を選択すると中国語を表示します。

4 ホストコントロール機能

4.1 ホストコントロールとは

ホストコントロール機能は
　ロボット制御機能
　I/O 信号のリード・ライト機能
２つの機能からなります。いずれもパソコンからの指令によって伝送を行います。
詳細については、「4.3 ロボット制御」「4.4 I/O 信号のリード・ライト」を参照してください。

4.2 起動と終了

■ ホストコントロールの起動

［スタート］ボタンの【プログラム】－【Motoman】－【MOTOCOM32】－【ホストコントロール】（Windows10 の場合：【Motoman】－【ホストコントロール】）を押し、「Host Control32」を起動します。［言語選択］が表示されます。任意の言語を選択して［OK］ボタンを押します。言語は、メニューからいつでも変更できます。

■ ホストコントロールの終了

メニューの【ファイル】－【ホストコントロールの終了】を選択し、ホストコントロールを終了します。

4.3 ロボット制御

ロボットのステータス（現在位置、アラーム・エラー・サーボ等の状態）のリードやシステムのコントロール（スタート・ホールド、ジョブの呼び出しなど）を行います。コントローラの伝送コマンドを個別に実行できます。（コントローラは、リモート状態にしておきます。）

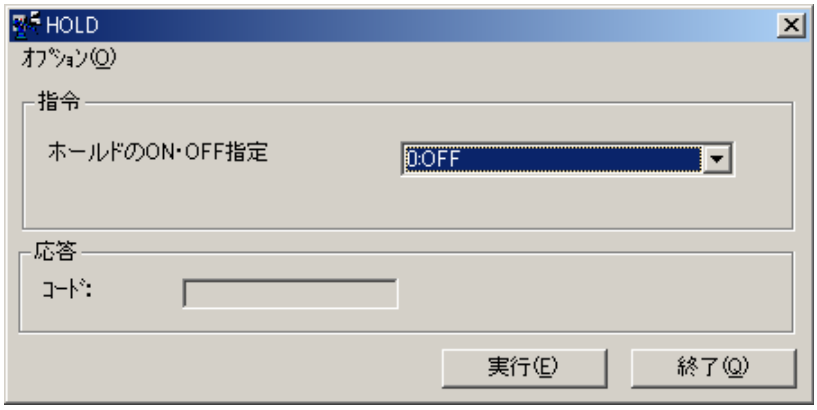
1.「ホストコントロール」を起動すると、ロボット制御画面が表示されます。

2.［指令］ボタンを押すと、使用できるコマンドの一覧を［コマンド］に表示します。

重要

- コントローラがERCの場合は、一部のコマンドが淡色表示となり、実行できません。
- コントローラがYRC1000、YRC1000micro、DX200、DX100、FS100、NX100でイーサネットサーバ機能を利用する場合は、一部のコマンドが淡色表示となり、実行できません。

3. コマンドボタンを押すと、コマンドごとの画面を表示します。



4. 画面表示に従って指令パラメータを入力します。(指令パラメータがない場合は不要です。)

5. [実行] ボタンを押すとコマンドが発行され、コントローラからの応答コードと応答データを表示します。
伝送コマンドは、以下のようになります。詳細については、コントローラのデータ伝送機能説明書
『YRC1000 データ伝送機能操作説明書』
『YRC1000micro データ伝送機能操作説明書』
『DX200 データ伝送機能操作説明書』
『DX100 データ伝送機能操作説明書』
『FS100 データ伝送機能操作説明書』
『NX100 データ伝送機能操作説明書』
『YASNAC XRC データ伝送機能操作説明書』
『YASNAC MRC データ伝送機能操作説明書』
『YASNAC ERC データ伝送機能ユーザーズマニュアル』
を参照してください。

			ERC	MRC	XRC	NX100/DX100/FS100/DX200 YRC1000/YRC1000micro
ステータスのリード	リード・監視系	RALARM	○	○	○	○
		RPOS *2	○	○	—	—
		RPOSJ	○	○	○	○
		RSTATS	○	○	○	○
		RJSEQ	○	○	○	○
		RPOSC	○	○	○	○
		JWAIT	○	○	○	○
		RGROUP *1	—	○	○	○
		RALARMS *3	—	—	—	○
	リード・データアクセス系	RJDIR	○	○	○	○
		RUFRAME	○	○	○	○
		UPLOAD	○	○	○	○
		SAVEV *1	—	○	○	○
		SAVEVP *3	—	—	—	○

システムのコントロール	操作系	HOLD	○	○	○	○
		RESET	○	○	○	○
		CANCEL	○	○	○	○
		MODE *1	—	○	○	○
		CYCLE	○	○	○	○
		HLOCK	○	○	○	○
		MDSP	○	○	○	○
		SVON	○	○	○	○
		CGROUP *1	—	○	○	○
		CTASK *1	—	○	○	○
	編集系	DELETE	○	○	○	○
		WUFRAME	○	○	○	○
		CVTRJ	○	○	○	○
		DOWNLOAD	○	○	○	○
		CVTSJ *1	—	○	○	○
		LOADV *1	—	○	○	○
		LOADVP *3	—	—	—	○
	ジョブ選択系	SETMJ	○	○	○	○
		JSEQ	○	○	○	○
	起動系	START	○	○	○	○
		MOVJ	○	○	○	○
		MOVL	○	○	○	○
		IMOV	○	○	○	○
		PMOVJ	○	○	○	○
		PMOVL	○	○	○	○

1 YRC1000, YRC1000micro, DX200, DX100, FS100, NX100, YASNAC XRC, MRC, MRC2 のみ有効なコマンド

2 YASNAC MRC, MRC2, ERC, ERC2 のみ有効なコマンド

3 YRC1000, YRC1000micro, DX200, DX100, FS100, NX100 のみ有効なコマンド

4.4 I/O 信号のリード・ライト

ロボットコントローラの I/O 信号のリード（読み込み）・ライト（書き込み）を行います。
リード・ライト可能な信号と画面は、以下のようにコントローラにより異なります。

重要 コントローラは、リモート状態にしておきます。

■ I/O 信号のリード・ライト可能な信号一覧

MRC

信 号	信号の範囲	分類名	読み込み	書き込み
0xxx	0010-0167（128 個）	ロボット汎用入力	○	×
1xxx	1010-1167（128 個）	ロボット汎用出力	○	×
2xxx	2010-2187（144 個）	外部入力	○	×
3xxx	3010-3187（144 個）	外部出力	○	×
4xxx	4010-4167（128 個）	ロボット専用入力	○	×
5xxx	5010-5247（192 個）	ロボット専用出力	○	×
6xxx	6010-6047（32 個）	タイマ／カウンタ	×	×
7xxx	7010-7327（256 個）	補助リレー	○	×
8xxx	8010-8087（64 個）	制御状態信号	○	×
82xx	8210-8247（32 個）	擬似入力信号	○	×
9xxx	9010-9167（128 個）	DL 入力	○	○

XRC

信 号	信号の範囲	分類名	読み込み	書き込み
0xxx	0010-0247（192 個）	ロボット汎用入力	○	×
1xxx	1010-1247（192 個）	ロボット汎用出力	○	×
2xxx	2010-2327（256 個）	外部入力	○	×
3xxx	3010-3327（256 個）	外部出力	○	×
4xxx	4010-4287（224 個）	ロボット専用入力	○	×
5xxx	5010-5387（304 個）	ロボット専用出力	○	×
6xxx	―	―	×	×
7xxx	7010-7887（704 個）	補助リレー	○	×
8xxx	8010-8127（96 個）	制御状態信号	○	×
82xx	8210-8247（32 個）	擬似入力信号	○	×
9xxx	9010-9167（128 個）	ネットワーク入力	○	○

NX100

信 号	信号の範囲	分類名	読み込み	書き込み
0xxxx	00010-01287（1024 個）	汎用入力	○	×
1xxxx	10010-11287（1024 個）	汎用出力	○	×
2xxxx	20010-21287（1024 個）	外部入力	○	×
22xxx	22010-23287（1024 個）	ネットワーク入力	○	○
3xxxx	30010-31287（1024 個）	外部出力	○	×
32xxx	32010-33287（1024 個）	ネットワーク出力	○	×
4xxxx	40010-40807（640 個）	専用入力 (システム)	○	×
5xxxx	50010-51007（800 個）	専用出力 (システム)	○	×
6xxxx	―	―	×	×
7xxxx	70010-79997（7992 個）	補助リレー (システム)	○	×
8xxxx	80010-80647（572 個）	制御状態信号 (システム)	○	×
82xxx	82010-82127（96 個）	擬似入力信号 (システム)	○	×

DX100

信 号	信号の範囲	分類名	読み込み	書き込み
0xxxx	00010-02567（2048 個）	汎用入力	○	×
1xxxx	10010-12567（2048 個）	汎用出力	○	×
2xxxx	20010-22567（2048 個）	外部入力	○	×
25xxx	25010-27567（2048 個）	ネットワーク入力	○	○
3xxxx	30010-32567（2048 個）	外部出力	○	×
35xxx	35010-37567（2048 個）	ネットワーク出力	○	×
4xxxx	40010-41607（1280 個）	専用入力 (システム)	○	×
5xxxx	50010-52007（1600 個）	専用出力 (システム)	○	×
6xxxx	―	―	×	×
7xxxx	70010-79997（7992 個）	補助リレー (システム)	○	×
8xxxx	80010-80647（512 個）	制御状態信号 (システム)	○	×
82xxx	82010-82207（160 個）	擬似入力信号 (システム)	○	×

FS100

信 号	信号の範囲	分類名	読み込み	書き込み
0xxxx	00010-01287（1024 個）	汎用入力	○	×
1xxxx	10010-11287（1024 個）	汎用出力	○	×
2xxxx	20010-21287（1024 個）	外部入力	○	×
25xxx	25010-26287（1024 個）	ネットワーク入力	○	○
3xxxx	30010-31287（1024 個）	外部出力	○	×
35xxx	35010-36287（1024 個）	ネットワーク出力	○	×
4xxxx	40010-41607（1280 個）	専用入力 (システム)	○	×
5xxxx	50010-52007（1600 個）	専用出力 (システム)	○	×
6xxxx	―	―	×	×
7xxxx	70010-79997（7992 個）	補助リレー (システム)	○	×
8xxxx	80010-80647（512 個）	制御状態信号 (システム)	○	×
82xxx	82010-82207（160 個）	擬似入力信号 (システム)	○	×

DX200

信 号	信号の範囲	分類名	読み込み	書き込み
0xxxx	00010-05127（4096 個）	汎用入力	○	×
1xxxx	10010-15127（4096 個）	汎用出力	○	×
2xxxx	20010-25127（4096 個）	外部入力	○	×
27xxx	27010-29567（2048 個）	ネットワーク入力	○	○
3xxxx	30010-35127（4096 個）	外部出力	○	×
37xxx	37010-39567（2048 個）	ネットワーク出力	○	×
4xxxx	40010-41607（1280 個）	専用入力 (システム)	○	×
5xxxx	50010-53007（2400 個）	専用出力 (システム)	○	×
6xxxx	―	―	×	×
7xxxx	70010-79997（7992 個）	補助リレー (システム)	○	×
8xxxx	80010-80647（512 個）	制御状態信号 (システム)	○	×
82xxx	82010-82207（160 個）	擬似入力信号 (システム)	○	×

YRC1000

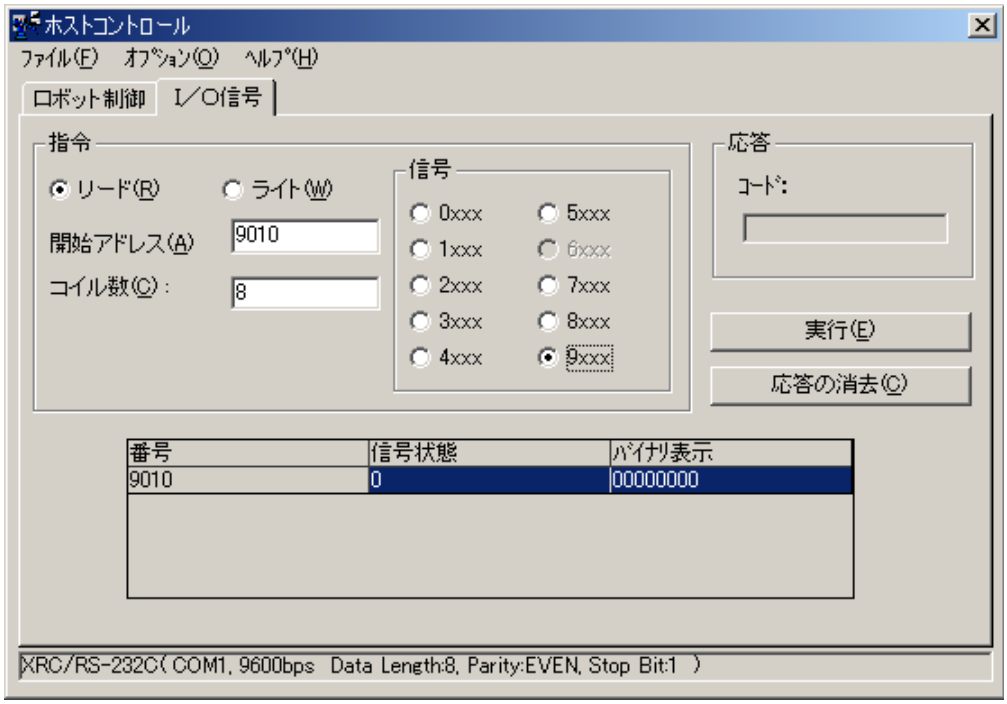
信 号	信号の範囲	分類名	読み込み	書き込み
0xxxx	00010-05127（4096 個）	汎用入力	○	×
1xxxx	10010-15127（4096 個）	汎用出力	○	×
2xxxx	20010-25127（4096 個）	外部入力	○	×
27xxx	27010-29567（2048 個）	ネットワーク入力	○	○
3xxxx	30010-35127（4096 個）	外部出力	○	×
37xxx	37010-39567（2048 個）	ネットワーク出力	○	×
4xxxx	40010-42567（2048 個）	専用入力 (システム)	○	×
5xxxx	50010-55127（4096 個）	専用出力 (システム)	○	×
6xxxx	―	―	×	×
7xxxx	70010-79997（7992 個）	補助リレー (システム)	○	×
8xxxx	80010-85127（4096 個）	制御状態信号 (システム)	○	×
87xxx	87010-87207（160 個）	擬似入力信号 (システム)	○	×

YRC1000micro

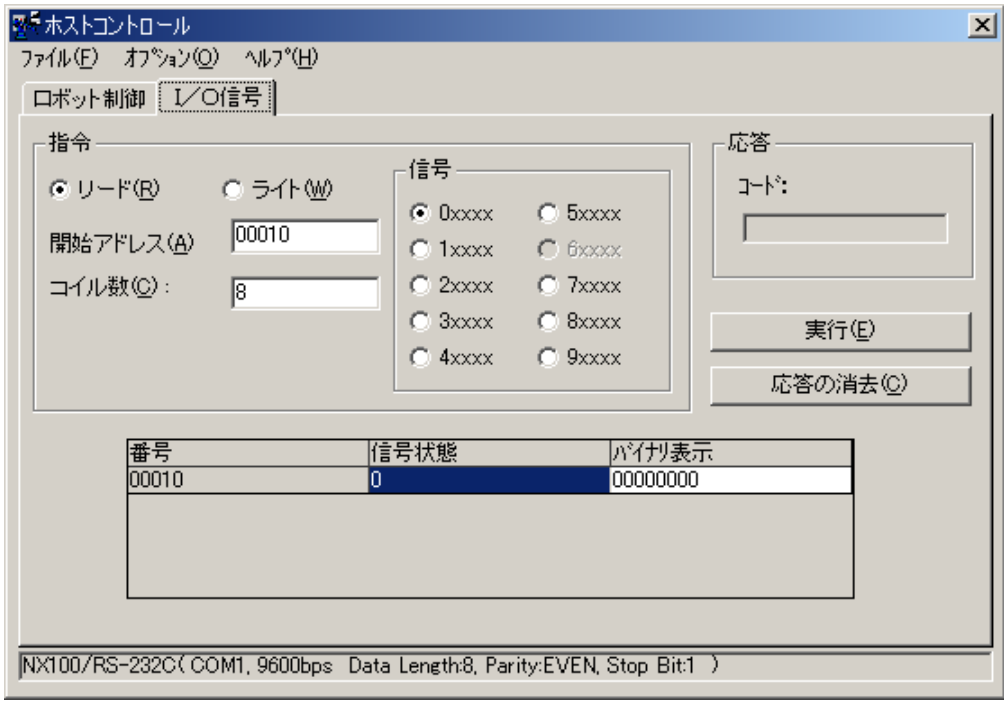
信 号	信号の範囲	分類名	読み込み	書き込み
0xxxx	00010-05127（4096 個）	汎用入力	○	×
1xxxx	10010-15127（4096 個）	汎用出力	○	×
2xxxx	20010-21287（1024 個）	外部入力	○	×
27xxx	27010-29567（2048 個）	ネットワーク入力	○	○
3xxxx	30010-31287（1024 個）	外部出力	○	×
37xxx	37010-39567（2048 個）	ネットワーク出力	○	×
4xxxx	40010-42567（2048 個）	専用入力 (システム)	○	×
5xxxx	50010-55127（4096 個）	専用出力 (システム)	○	×
6xxxx	―	―	×	×
7xxxx	70010-79997（7992 個）	補助リレー (システム)	○	×
8xxxx	80010-85127（4096 個）	制御状態信号 (システム)	○	×
87xxx	87010-87207（160 個）	擬似入力信号 (システム)	○	×

■ I/O 信号のリード・ライト画面

XRC、MRC



YRC1000、YRC1000micro、DX200、DX100、FS100、NX100



I/O 信号のライトを行う場合は、指令モード［ライト］を選択し、信号を一覧表より選択します。I/O 編集画面が表示され、信号状態を入力します。

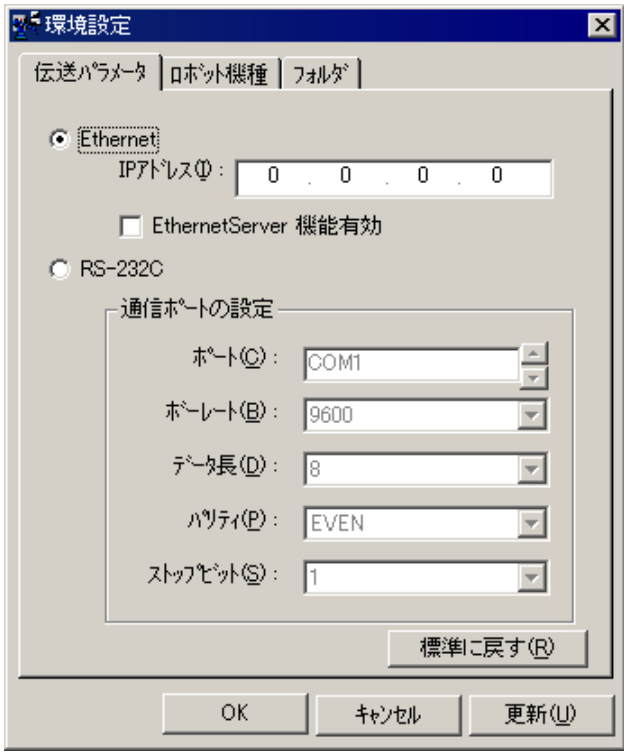
重要 本機能は、MRC のバージョン Ver4.111 以降と XRC、NX100、DX100、FS100、DX200、YRC1000、YRC1000micro で有効です。

4.5 環境設定

環境設定では、ホストコントロールを動作させるためのさまざまな設定を行います。【オプション】－【環境設定】を選択し、［環境設定］ダイアログで各項目の設定を行います。

［伝送パラメータ］の設定

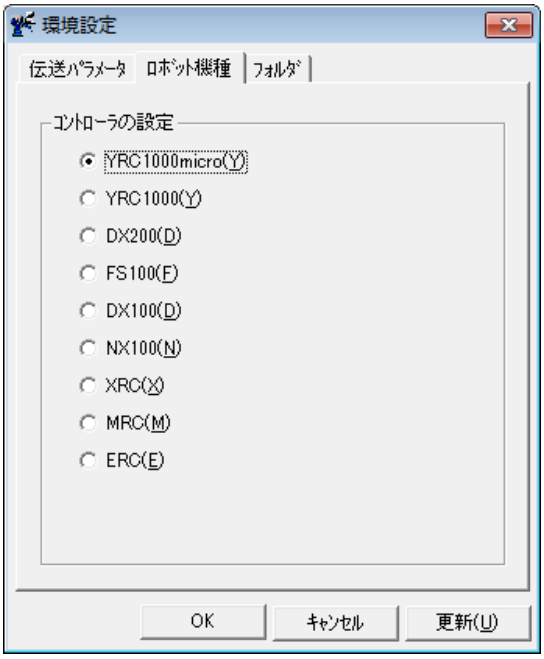
コントローラとの通信プロトコルの選択、設定を行います。
通信は、Ethernet 伝送または RS-232C 伝送があり、オプションボタンで選択します。



Ethernet		Ethernet 伝送の設定項目は、［IP アドレス］のみです。 ネットワークに接続されたコントローラに割り当てられている IP アドレスを入力します。
	EthernetServer 機能有効	コントローラの設定が「YRC1000」「YRC1000micro」「DX200」「DX100」「FS100」「NX100」の場合のみ表示されます。 コントローラのイーサネットサーバ機能が有効な場合で本機能を選択することで､イーサネットサーバ機能を用いて通信を行ないます。
RS-232C		［ポート番号］、［ボーレート］、［データ長］、［パリティ］、［ストップビット］をコントローラの設定にあわせて選択します。 ［標準に戻す］ボタンをクリックすると、デフォルトのデータが設定されます。

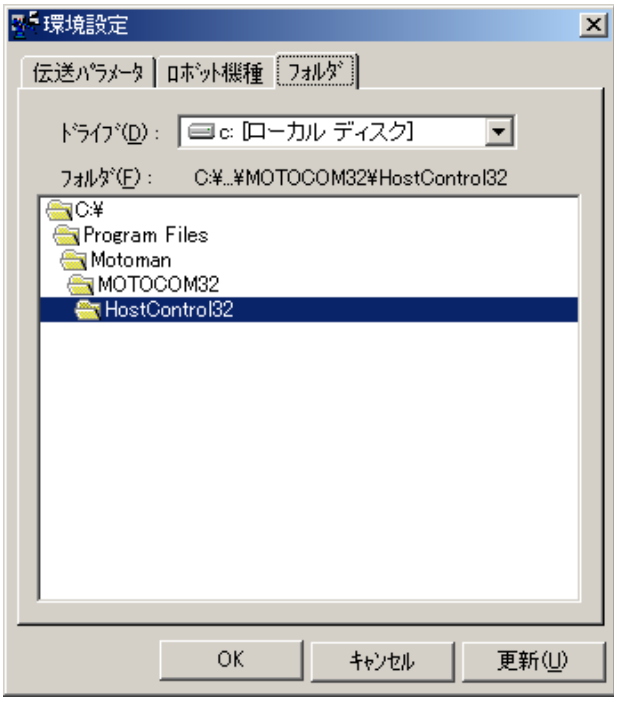
［ロボット機種］の設定

伝送を行うコントローラに設定します。



［フォルダ］の設定

通信対象となるフォルダを設定します。
ここでのドライブおよびフォルダの設定が、「DOWNLOAD」「UPLOAD」などのファイル操作のコマンドを扱う時の伝送対象となります。



4.6 言語選択

ホストコントロールで表示する言語を選択します。
【オプション】－【言語選択】を選択すると、言語選択ダイアログが表示され、言語を選択します。

プログラム開始時に言語選択を行なわない	本項目をチェックすると、次回から起動時にこのダイアログを表示しません。また、チェックをはずすと、起動するたびに［言語選択］ダイアログが表示され、言語切り替えが簡単に行えます。

重要 言語によっては正しいフォントの設定を行なわないと、うまく字が表されない場合があります。

4.7 バージョン情報

ホストコントロールのバージョン情報を表示します。

5 作業ジョブ自動段取り替え機能

5.1 作業ジョブ自動段取り替えとは

DCI(Data Comunication by Instruction) 伝送機能を用いて、コントローラから伝送要求されてくる作業番号のジョブを伝送します。簡単に作業ジョブの自動段取り替えを実現する機能です。
「ワーク品種が多い」あるいは「作業が複雑なためにジョブが長くなる。」などの原因でコントローラのメモリーが足りない状況になった場合にパソコンを外部メモリーとして利用する方法があります。このような場合にコントローラ側からパソコン側へ作業番号を伝送し、必要なジョブをアップロードし、そのジョブを実行します。
コントローラのメモリが足りない状況を想定し、パソコンからアップロードする作業ジョブは全て同一の名称（名称 :WORKJOB）に名称変更してロードします。
本アプリケーションでは、作業番号の 1 番から 5 番までを続けて作業する構成になっています。（必要に応じ、作業ジョブの追加も可能です。）

5.2 「自動運転」を実行の前に

作業ジョブ自動段取り替えの「自動運転」を実行するためには、コントローラに以下のジョブが必要です。（このジョブはコントローラでは、作業ジョブ自動段取り替えのマスタージョブとして位置付けてください。）このジョブと 1 番から 5 番までのジョブは、MOTOCOM32 をインストールした時にサンプルで登録します。作業ジョブ自動段取り替えを行う前にホストコントロールを用いてこのジョブをコントローラに伝送してください。

ジョブ名称： ROBJOB

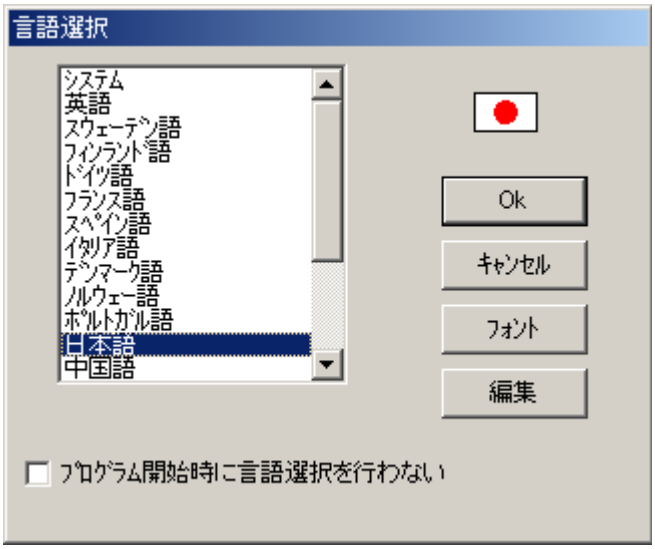
ジョブ内容：

```
NOP
'DATA INITIALIZE
SET I000 0
'
*START
TIMER T=5.00
'DATA SET
INC I000
'
'DCI INSTRUCTION
SAVEV I000
TIMER T=1.00
LOADJ JOB:WORKJOB JBI
CALL JOB:WORKJOB
JUMP *START IF I000<5
'
END
```

5.3 起動と終了

■ 作業ジョブ自動段取り替えの起動

［スタート］ボタンの【プログラム】－【Motoman】－【MOTOCOM32】－【作業ジョブ自動段取り替え】（Windows10 の場合 :【Motoman】－【作業ジョブ自動段取り替え】）を押し、「作業ジョブ自動段取り替え」を起動します。［言語選択］が表示されます。任意の言語を選択して［OK］ボタンを押します。言語は、メニューからいつでも変更できます。



■ 作業ジョブ自動段取り替えの終了

メニューの【ファイル】－【終了】を選択し、作業ジョブ自動段取り替えを終了します。

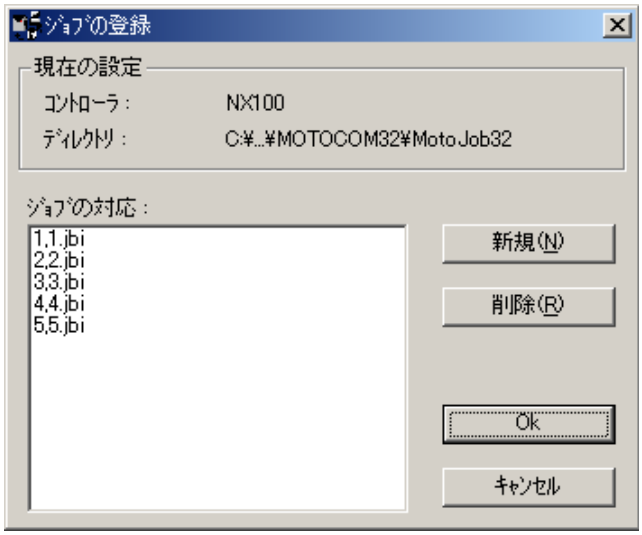
5.4 操作手順

5.4.1 ジョブの登録

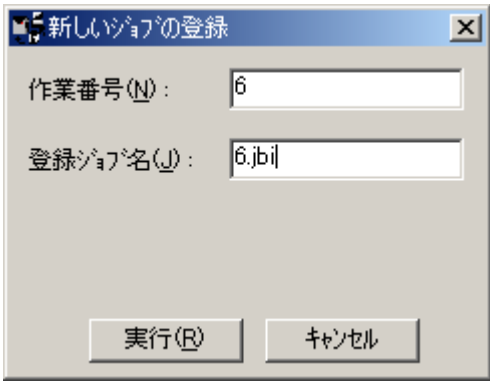
作業番号とジョブ名の対応は、あらかじめ登録しておく必要があります。

■ 新規登録

1. ［ジョブの登録］ダイアログで［新規］ボタンを押します。



2. ［新しいジョブの登録］ダイアログを表示しますので、［作業番号］、［登録ジョブ名（拡張子まで含む)］を入力し、「実行」ボタンを押します。



3. すでに登録ジョブが存在する場合は、そのジョブと対応付けます（プログラムから問合わせします)。そうでない場合は、コントローラから登録ジョブを受信します。［はい］、［いいえ］いずれかのボタンを押します。

4. ［ジョブの登録］ダイアログで、［OK］ボタンを押します。これで新規に登録した情報が全て登録されます。［キャンセル］ボタンを押すと、全ての登録操作はキャンセルされます。

■ 削除

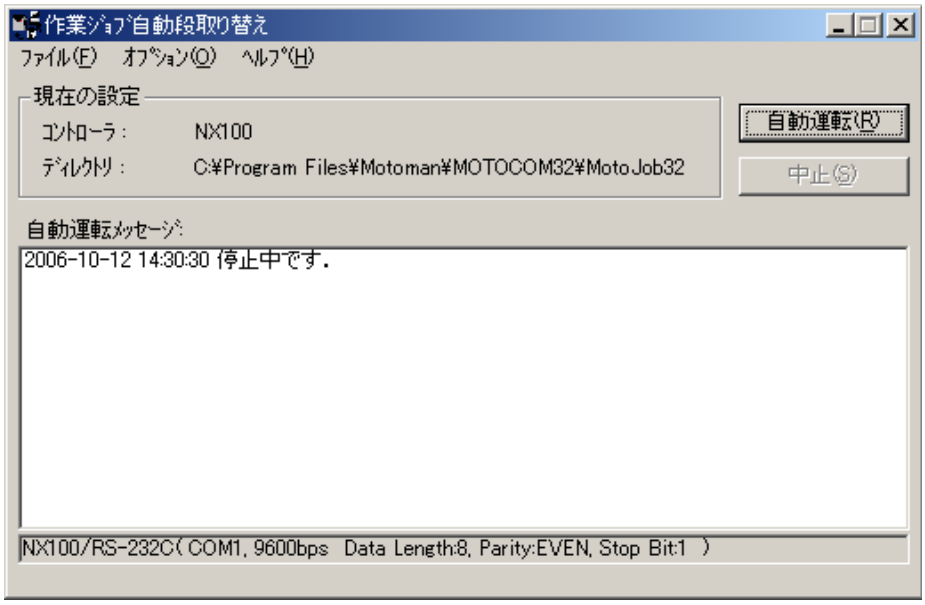
1. ［ジョブの登録］ダイアログの［ジョブの対応］リストから削除したいジョブを選択します。（複数選択可能。）
2. ［削除］ボタンを押します。
3. ［ジョブの登録］ダイアログの［OK］ボタンを押します。これで削除した情報は全て登録されます。［キャンセル］ボタンを押すと、全ての登録操作はキャンセルされます。

5.4.2 自動運転

［作業ジョブ自動段取り替え］ダイアログの［自動運転］ボタンを押すと、コントローラからの作業番号受信待ちになります。［中止］ボタンが押されるまでは、以下のシーケンスを繰り返します。

1. コントローラから作業番号を受信すると、あらかじめその番号に対して登録されていたジョブを「WORKJOB」という名前に変えます。
2. コントローラからの「WORKJOB」の受信要求を待ちます。
3. 受信要求を受けるとジョブを送信し、再び作業番号受信待ちになります。

重要 自動運転中は、ロボットの変更などの淡色表示となっている機能は使用できません。



5.4.3　中止

自動運転を終了する場合には、［作業ジョブ自動段取り替え］ダイアログの［中止］ボタンを押します。
通常は、コントローラから先に DCI 運転の停止を行ってください。（コントローラにアラームが発生することがあります。）

重要　［中止］ボタンを押した後、実際に停止状態になるまでしばらく時間を要する場合があります。

5.4.4　ログファイルの表示

［オプション］－［ロボファイルの表示］でログファイルの内容を表示します。

5.4.5　メッセージの消去

［オプション］－［メッセージの消去］で自動運転メッセージを消去します。
メッセージの消去は、プログラムで定期的に行っていますので特に意識する必要はありません。

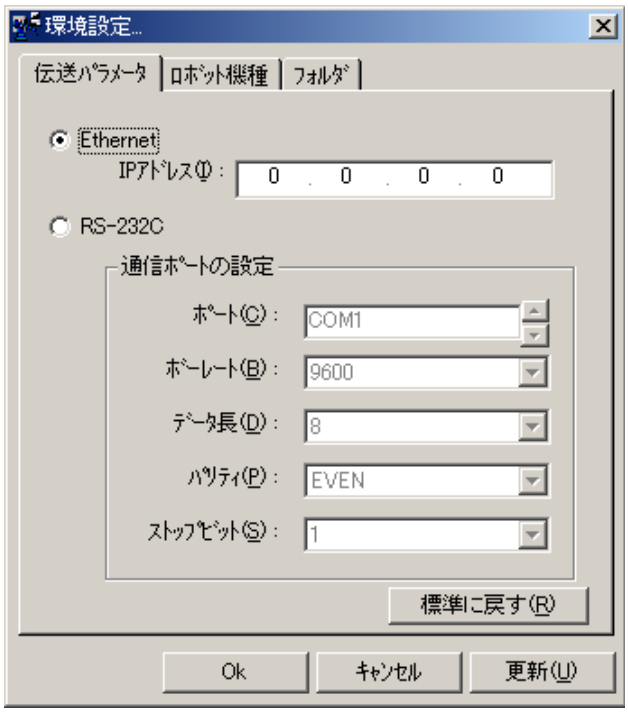
5.5 環境設定

5.5.1 環境設定

環境設定では、作業ジョブ自動段取り替えを動作させるためのさまざまな設定を行います。
【オプション】－【環境設定】を選択し、［環境設定］ダイアログで各項目の設定を行います。

［伝送パラメータ］の設定

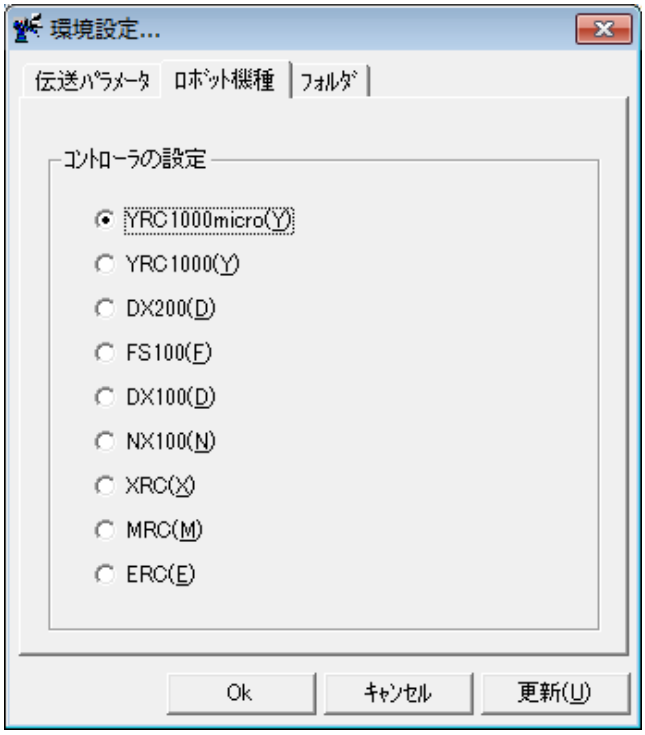
コントローラとの通信プロトコルの選択、設定を行います。
通信は、Ethernet 伝送または RS-232C 伝送があり、オプションボタンで選択します。



Ethernet	Ethernet 伝送の設定項目は、［IP アドレス］のみです。 ネットワークに接続されたコントローラに割り当てられている IP アドレスを入力します。
RS-232C	［ポート番号］、［ボーレート］、［データ長］、［パリティ］、［ストップビット］をコントローラの設定にあわせて選択します。 ［標準に戻す］ボタンをクリックすると、デフォルトのデータが設定されます。

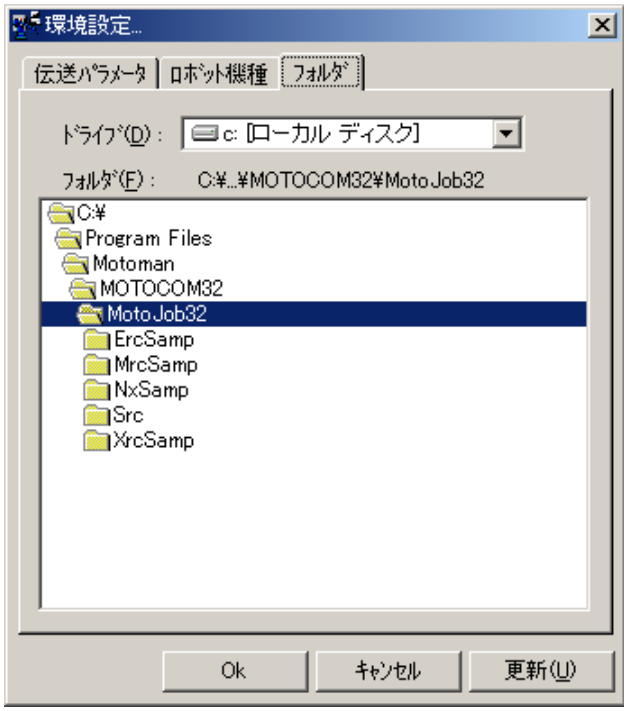
［ロボット機種］の設定

伝送を行うコントローラに設定します。



［フォルダ］の設定

通信対象となるフォルダを設定します。
ここでのドライブ及びフォルダの設定が、「DOWNLOAD」「UPLOAD」などのファイル操作のコマンドを扱う時の伝送対象となります。



5.5.2 起動時に自動運転

［オプション］－［起動時に自動運転］を設定すると、作業ジョブ自動段取り替えの起動と

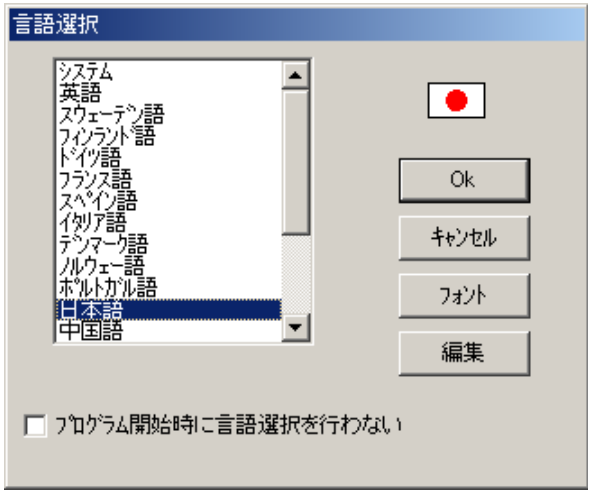
同時に自動運転に入るか否かを決めます。メニューの左にチェックマークがある場合は、自動運転に入ることを意味します。

5.5.3 ログファイルを採取

［オプション］－［ログファイルを採取］を設定すると、自動運転時の伝送ログを採取するか否かを決めます。メニューの左にチェックマークがある場合は、ログを採ることを意味します。
ログファイルは、MOTOCOM32 をインストールしたフォルダに「ONLINE.LOG」というファイル名で作成されます。プログラムは、このファイルが一定の大きさを超えると上書きをしますが、「ONLINE.LO$」のファイル名で 1 世代前のログファイルを残します。

5.6 言語選択

作業ジョブ自動段取り替えで表示する言語を選択します。
【オプション】－【言語選択】を選択すると、言語選択ダイアログが表示され、言語を選択します。



プログラム開始時に言語選択を行なわない	本項目をチェックすると、次回から起動時にこのダイアログを表示しません。また、チェックをはずすと、起動するたびに［言語選択］ダイアログが表示され、言語切り替えが簡単に行えます。

重要 言語によっては正しいフォントの設定を行なわないと、うまく字が表されない場合があります。

5.7 バージョン情報

作業ジョブ自動段取り替えのバージョン情報を表示します。

6 伝送アプリケーション作成手順

6.1 概要

ユーザがコントローラとパソコン間の伝送アプリケーションを簡単に作成できるようにするためにデータ伝送関数（Windows DLL ファイル形式、32 ビット版 伝送ライブラリ「Motocom32.DLL」）を含むサンプルプログラム（Windows のアプリケーション開発ツール「Visual Basic」、「Visual C++」、「Visual C#」を使用）を用いて、伝送アプリケーションの作成手順を説明します。（「Visual Basic」、「Visual C++」、「Visual C#」以外の言語でも同じように作成することができます。）

サンプルプログラムのプログラムリストは、パブリックのドキュメントフォルダ内の「MOTOMAN\MOTOCOM32\MOTOCOM32DLL\Sample」フォルダに入っています。また、作業ジョブ自動段取り替えソフトのプログラムリストは「MOTOMAN\MOTOCOM32\MotoJob32\Src」フォルダに入っています。アプリケーション作成の際、参照ください。

重要

- サンプルプログラムを実行する場合は、MOTOCOM32 をインストールしたフォルダで実行してください。
- サンプルプログラムを運用した結果の影響については、いっさい責任を負いかねますのでご了承ください。

6.2 Visual Basic での作成方法

6.2.1 前準備

伝送アプリケーションを作成するためには、あらかじめ以下のシステムをパソコンにインストールしておく必要があります.

(1) Microsoft Windows 7 / 10 *1

(2) Visual Basic Ver6.0 以上 *2

*1 Microsoft Windows 7 / 10 は米国マイクロソフト社の登録商標です。

*2 Visual Basic は米国マイクロソフト社の登録商標です。

6.2.2 作成手順

ここでは、簡単な例として、テキストボックスに入力されたジョブをコントローラと送受信するプログラムを説明します。

■ コードモジュールの作成

Visual Basic から「Motocom32.DLL」を呼び出すためには、使用する「Motocom32.DLL」の I/F 関数を宣言する必要があります。方法として次の 2 つの方法があります。

自分で DLL 関数の宣言を記述する。

MOTOCOM32 パッケージ付属の定義ファイルを利用する。

自分で DLL 関数の宣言を記述する。

コードモジュールを追加し、Declaration 部に、以下のような宣言を記述します。

```
Declare Function BscOpen Lib "MotoCom32" Alias "_BscOpen@8" _
        (ByVal Path As String, ByVal mode As Integer) As Integer
Declare Function BscClose Lib "MotoCom32" Alias "_BscClose@4" _
        (ByVal nCid As Integer) As Integer
Declare Function BscSetCom Lib "MotoCom32" Alias "_BscSetCom@24" _
        (ByVal nCid As Integer, ByVal Port As Integer, ByVal Baud As Long, ByVal
        Parity As Integer, ByVal clen As Integer, ByVal stp As Integer) As Integer
Declare Function BscSetEther Lib "MotoCom32" Alias "_BscSetEther@16" _
        (ByVal nCid%, ByVal IPaddr$, ByVal mode%, ByVal hWnd&) As Integer
Declare Function BscConnect Lib "MotoCom32" Alias "_BscConnect@4" _
        (ByVal nCid As Integer) As Integer
Declare Function BscDisConnect Lib "MotoCom32" Alias "_BscDisConnect@4" _
        (ByVal nCid As Integer) As Integer
Declare Function BscDownLoad Lib "motocom32.dll" Alias "_BscDownLoad@8" _
        (ByVal nCid As Integer, ByVal fName As String) As Integer
Declare Function BscUpLoad Lib "motocom32.dll" Alias "_BscUpLoad@8" _
        (ByVal nCid As Integer, ByVal fName As String) As Integer
```

BscOpen() で接続回線種別を選択するためのパラメータとして

```
Public Const PACKETCOM = (1)
Public Const PACKETETHERNET = (16)
```

を定義します。

MOTOCOM32 パッケージ付属の定義ファイルを利用する。

「Motocom32.DLL」パッケージには DLL の I/F 関数を定義した「Motocom32.BAS」ファイルが用意されています。そのファイルを Visual Basic のプロジェクトに追加します。

(1) 「Motocom32.DLL」パッケージの「Motocom32.BAS」ファイルを作成したいアプリケーションのソースディレクトリへコピーします。
(2) ［プロジェクト］－［ファイルの追加］メニューから「Motocom32.BAS」ファイルを指定してプロジェクトへ追加します。

さらに、以下のようなポートをオープン／クローズするためのサブモジュールを作成します。

```
Function Ms_BscOpenComm(TransMode%, FuncMode%)
Function Ms_BscCloseComm()
```

「Function Ms_BscOpenComm()」で上記関数の内容部分（プログラムリスト）をコピーし、Ms_BscOpenComm () 関数に貼り付けてください。同様の操作を Ms_BscCloseComm（「Function Ms_BscCloseComm()」）で実行してください。

■　フォームモジュールの作成

以下のコントロールを作成します。

(1) プログラムの画面となるフォーム
さらに、このフォームの上に、以下のコントロールを作成します。
(2) ジョブ名を入力するテキストボックス（コントロール名を "TxtJobName"、テキスト名を "0.JBI" とします。）
(3) 送信ボタン（コントロール名を "CmdDownLoad"、キャプション名を " 送信 " とします。）
(4) 受信ボタン（コントロール名を "CmdUpLoad"、キャプション名を " 受信 " とします。）
(5) 終了ボタン（コントロール名を "CmdExit"、キャプション名を " 終了 " とします。）

コントロールを作成したら、各ボタンにイベントプロシジャを記述します。

```
Sub CmdDownload_Click ()
Sub CmdUpLoad_Click ()
Sub CmdExit_Click ()
```

「Sub CmdDownload_Click ()」で上記関数の内容部分（プログラムリスト）をコピーし、CmdDownload_Click () 関数に貼り付けてください。同様の操作を CmdUpLoad（「Sub CmdUpLoad_Click ()」）、CmdExit（「Sub CmdExit_Click ()」）で実行してください。

■　ＥＸＥ ファイルの作成と実行

Visual Basic の【ファイル】メニューから【EXE ファイルの作成】を選択すると、実行可能モジュールが作成されます。このモジュールを送受信するジョブと同じフォルダに置いて実行すると、ジョブの送受信を行うことができます。

重要

データ伝送関数（Windows DLL ファイル形式、**ファイル名：「Motocom32.DLL」および「Motolk.DLL」「Motolkr.Dll」「Vrp32.DLL」**）は MOTOCOM32 をインストールしたフォルダに入っています。アプリケーション実行時は、実行可能モジュールを作成したフォルダにコピーしてご利用ください。Ethernet 経由で伝送を行なう場合は、更に「HslSrv32.exe」も「Motocom32.DLL」と同じフォルダにコピーしてください。

6.3 Visual C++ での作成方法

6.3.1 準備

伝送アプリケーションを作成するためには、あらかじめ以下のシステムをパソコンにインストールしておく必要があります。

(1) Microsoft Windows 7 / 10 *1
(2) Visual C++ Ver6.0 以上 *2

*1 Microsoft Windows 7 / 10 は米国マイクロソフト社の登録商標です。
*2 Visual C++ は米国マイクロソフト社の登録商標です。

6.3.2 作成手順

ここでは、簡単な例として、テキストボックスに入力されたジョブをコントローラと送受信するプログラムを説明します。

■ スケルトンの作成

以下の説明では、Visual C++ Ver6.0 を用いて作成手順を説明しています。

(1) Microsoft development Studio を起動し、【ファイル】－【新規作成】を選択して［新規作成］画面を表示します。
(2) ［プロジェクト］タブを選択し、「MFC AppWizard（exe）」を選択します。
(3) プロジェクト名（ここでは Test と入力）を入力し、プロジェクトを作成するフォルダを指定したら［OK］ボタンを押します。
(4) 「ステップ 1」で作成するアプリケーションの種類を「ダイアログベース」とし、［終了］ボタンを押します。

以上の操作を行うと、ウィンドウに［OK］［キャンセル］ボタンのみが存在するダイアログを表示するソースコードが作成されます。

■ DLL 呼び出しの定義

(1) ダイアログクラスのソースファイル（TestDlg.cpp）で、MOTOCOM32 アプリケーションに付属の「motocom.h」をインクルードします。また、以降の説明でサンプルソースで使用するヘッダーファイル「direct.h」もインクルードしておきます。

```
#include "stdafx.h"
#include "Test.h"
#include "TestDlg.h"
#include <direct.h>        <------ この行を追加
#include "motocom.h"       <------ この行を追加
```

(2) MOTOCOM32 アプリケーションに付属の「motocom32.lib」「motocom.h」および

データ伝送関数（Windows DLL ファイル形式、**ファイル名：「Motocom32.DLL」および「Motolk.DLL」「Motolkr.Dll」「Vrp32.DLL」**）をプロジェクトのあるフォルダにコピーします。

(3) 「ビルド」－「設定」ボタンを押し、「プロジェクトの設定」ダイアログの「リンク」タブを開きます。「オブジェクト／ライブラリ モジュール」設定欄に「motocom32.lib」を指定し、［OK］ボタンを押します。

以上の操作を行うと、「motocom.h」をインクルードしたファイルで MOTOCOM32 の関数が使用可能となります。

重要

ライブラリファイル (**ファイル名：「Motocom32.Lib」**）およびインクルードファイル (**ファイル名：「Motocom.h」**）は MOTOCOM32 をインストールしたフォルダに入っています。

■ ダイアログの編集

作成したダイアログで以下のような編集を行います。
IDD_TEST_DIALOG ダイアログを開いてください。

(1) デフォルトで作成される［キャンセル］ボタンを削除します。
(2) デフォルトで作成される［OK］ボタンのキャプションを「終了」に変更します。
(3) ジョブ名を入力するエディットコントロールを作成し、ID 名を「IDC_JOBNAME」とします。
(4) 送信用プッシュボタンを作成し、キャプションを「送信」、ID 名を「IDC_DOWNLOAD」とします。
(5) 受信用プッシュボタンを作成し、キャプションを「受信」、ID 名を「IDC_UPLOAD」とします。

■ 関数及び変数の追加

(1) 「TestDlg.h」を開き、次のような関数宣言を追加します。

```
short TestOpenComm( int TransMode, int FuncMode );
short TestCloseComm( short ncid );
```

(2) 通信ポートをオープンする関数「CTestDlg::TestOpenComm」を作成します。
(3) 通信ポートをクローズする関数「CTestDlg::TestCloseComm」を作成します。
(4) ClassWizard にて、［送信］ボタン（IDC_DOWNLOAD）で BN_CLICKED メッセージ発生時の関数「CTestDlg::OnDownload」を作成します。
(5) ClassWizard にて、［受信］ボタン（IDC_UPLOAD）で BN_CLICKED メッセージ発生時の関数「CTestDlg::OnUpload」を作成します。
(6) ClassWizard にて、ジョブ名エディットコントロール (IDC_JOBNAME) の入力文字用の変数「m_jobname」を CEdit 型で追加します。
各関数を追加したら、各関数内にコードを記述します。

CTestDlg::TestOpenComm 関数
CTestDlg::TestCloseComm 関数

CTestDlg::OnDownload 関数
CTestDlg::OnUpload 関数

TestOpenComm() では、IP アドレス（変数 : IPAddress)、通信モード（変数：mode)、アプリケーションの格納パス（変数：cur_dir）を指定しています。お客様の環境に合わせて変更してください。

「CTestDlg::TestOpenComm 関数」で上記関数の内容部分（プログラムリスト）をコピーし、CTestDlg::TestOpenComm() 関数に貼り付けてください。同様の操作を CTestDlg::TestCloseComm（「CTestDlg::TestCloseComm 関数」)、CTestDlg::OnDownload（「CTestDlg::OnDownload 関数」)、CTestDlg::OnUpload（「CTestDlg::OnUpload 関数」）で実行してください。

■ EXE ファイルの作成と実行

Visual C++ のビルドメニューにてビルドを行うと、実行可能モジュールが作成されます。このモジュールを、送受信するジョブと同じディレクトリに置いて実行すると、ジョブの送受信を行うことができます。

データ伝送関数（Windows DLL ファイル形式、**ファイル名：「Motocom32.DLL」および「Motolk.DLL」「Motolkr.Dll」「Vrp32.DLL」**）は MOTOCOM32 をインストールしたフォルダに入っています。アプリケーション実行時は、実行可能モジュールを作成したフォルダにコピーしてご利用ください。Ethernet 経由で伝送を行なう場合は、更に「HslSrv32.exe」も「Motocom32.DLL」と同じフォルダにコピーしてください。

6.4 Visual Studio C# での作成方法

6.4.1 前準備

伝送アプリケーションを作成するためには、あらかじめ以下のシステムをパソコンにインストールしておく必要があります。

(1) Microsoft Windows 7 / 10 *1

(2) Visual Studio 2012, .NETFramework 4.0 以上 *2

*1 Microsoft Windows 7 / 10 は米国マイクロソフト社の登録商標です。
*2 Visual Studio 2012, .NETFramework は米国マイクロソフト社の登録商標です。

6.4.2 作成手順

ここでは、簡単な例として、テキストボックスに入力されたジョブをコントローラと送受信するプログラムを説明します。

■ プロジェクトの作成

Visual Studio で「Visual C#」「Windows アプリケーション」のプロジェクトを新規作成します。

■ ライブラリ参照設定

Visual Studio C# で「Motocom32.DLL」を使用するために、「Motocom32CS.DLL」を参照設定する必要があります。
作成したプロジェクトで、参照の追加を行い、「Motocom32」インストールフォルダ内の「MOTOCOM32DLL」フォルダにある「Motocom32CS.DLL」を選択します。

[プロジェクト] ー [参照の追加] メニューから [参照の追加] 画面で、
[参照] タグを選択し、「Motocom32」インストールフォルダ内の
「MOTOCOM32DLL」フォルダにある「Motocom32CS.DLL」を選択して
プロジェクトへ追加します。

「Motocom32CS.DLL」の名前空間で定義されている型をインポートするために、次の using ディレクティブを記述します。

```
using MotoCom32CS;
```

さらに、以下のようなポートをオープン／クローズするためのメソッドを作成します。

```
private short Ms_BscOpenComm( int mode )
private short Ms_BscCloseComm( short nCid )
```

「private short Ms_BscOpenComm()」で上記関数の内容部分（プログラムリスト）をコピーし、Ms_BscOpenComm() 関数に貼り付けてください。同様の操作を Ms_BscCloseComm（「private

short Ms_BscCloseComm()」）で実行してください。

■ フォームモジュールの作成

以下のコントロールを作成します。

(1) プログラムの画面となるフォーム
さらに、このフォーム上に、以下のコントロールを作成します。
(2) ジョブ名を入力するテキストボックス（コントロール名を "TxtJobName"、テキスト名を "0.JBI" とします。）
(3) 送信ボタン（コントロール名を "CmdDownLoad"、キャプション名を " 送信 " とします。）
(4) 受信ボタン（コントロール名を "CmdUpLoad"、キャプション名を " 受信 " とします。）
(5) 終了ボタン（コントロール名を "CmdExit"、キャプション名を " 終了 " とします。）

コントロールを作成したら、各ボタンのイベントを記述します。

```
private void CmdDownLoad_Click( object sender, EventArgs e )
private void CmdUpLoad_Click( object sender, EventArgs e )
private void CmdExit_Click( object sender, EventArgs e )
```

「private void CmdDownLoad_Click()」で上記関数の内容部分（プログラムリスト）をコピーし、CmdDownLoad_Click() 関数に貼り付けてください。同様の操作を CmdUpLoad_Click（「private void CmdUpLoad_Click()」）、CmdExit_Click（「private void CmdExit_Click()」）で実行してください。

■ EXE ファイルの作成と実行

Visual Studio の [ビルド] メニューから [ソリューションのビルド] を選択すると、実行可能モジュールが作成されます。このモジュールのあるフォルダに送受信するジョブを置いて実行すると、ジョブの送受信を行うことが出来ます。

重要

データ伝送関数（Windows DLL ファイル形式、**ファイル名：「Motocom32.DLL」および「Motolk.DLL」「Motolkr.Dll」「Vrp32.DLL」**）は MOTOCOM32 をインストールしたフォルダに入っています。アプリケーション実行時は、実行可能モジュールを作成したフォルダにコピーしてご利用ください。Ethernet 経由で伝送を行なう場合は、更に「HslSrv32.exe」も「Motocom32.DLL」と同じフォルダにコピーしてください。

6.5 作業ジョブ自動段取り替えソフト作成手順説明

ここでは、［自動運転］ボタンが押された時に呼ばれるプロシジャ（プロシジャ名：Sub DciOnline）を例に、DCI アプリケーションの作成手順を説明します。
なお、作業ジョブ自動段取り替えソフトは Visual Basic で作成しているため、説明は Visual Basic で行っていますが、Visual C++ または他の言語でも同じように作成することができます。
処理は大きくは以下の流れになっています。

(1) 伝送ポートのオープン　（関数名：Ms_BscOpenComm()）
(2) ジョブ番号の受信　（関数名：DciGetJobNo()）
(3) 送信ジョブの準備　（関数名：GetJobNameByNo()、JobCopy()）
(4) ジョブの送信　（関数名：DciLoadSave()）
(5) 伝送ポートのクローズ　（関数名：Ms_BscCloseComm()）

以降、各処理のリストを記述します。

■ Sub DciOnline

```
Sub DciOnline (CvtName As String, lst As Control, LogFile As String)
'input     CvtName      コピーするジョブ名
'          Lst          メッセージ出力用リスト名
'          LogFile      ログファイル名
'output    なし
    Dim nCid As Integer
    Dim JobNo As Integer
    Dim JobName As String
    Dim rc As Integer
    Dim Cycle As Long

    Cycle = 0
    ' 通信ハンドラの獲得
    nCid = Ms_BscOpenComm(mode, 1)  ' mode=0 or 1
    If nCid >= 0 Then
    ' 中止ボタンが押される（F_QUIT フラグが真になる）まで、作業番号の受信とジョブの送信を繰り返し行う.
        Do While Not F_QUIT
        DispLogMsg lst, "***** 作業番号受信待ちです．*****", ""
            ' 作業番号を受信する.
             If Not DciGetJobNo(nCid, JobNo, lst, LogFile) Then Exit Do
            DispLogMsg lst, " 作業番号 (" + Format$(JobNo) + ") を受信しました．",
LogFile
            ' 作業番号に対応するジョブ名を獲得する.
           JobName = GetJobNameByNo(JobNo)
            If JobName = "" Then
                MsgBox " 対応するジョブが登録されていません。"
                Exit Do
```

```
            End If
            ' 対応するジョブを送信用の名前にコピーする。
             If Not JobCopy(JobName, CvtName) Then
                MsgBox " ジョブコピーができません．(" + JobName + ")"
                Exit Do
            End If
            DispLogMsg lst, JobName + " を " + CvtName + " にコピーしました。",
LogFile

            DispLogMsg lst, "***** ジョブ伝送要求待ちです。*****", ""
            'YASNAC からの指示に応じて、ジョブを送信する。
            If Not DciLoadSave(nCid, lst, LogFile) Then Exit Do
            Cycle = Cycle + 1
            DispLogMsg lst, " ジョブ送信を完了しました (" + Format$(Cycle) + " 巡 ).",
LogFile
        Loop

        ' 通信ハンドラの解放
        rc = Ms_BscCloseComm(nCid)
         If rc <> 0 Then
            MsgBox "BscCloseComm がエラー終了しました " + "(" + Format$(rc) + ")."
        End If
    Else
        MsgBox " オープンできません． "
    End If
End Sub
```

参考

- プログラム中の1重下線は、プログラムリストが記述されている関数です。
  - Ms_BscOpenComm ： 「Function Ms_BscOpenComm()」
  - Ms_BscCloseComm ： 「Function Ms_BscCloseComm()」
  - DciGetJobNo ： 「Function DciGetJobNo」
  - DciLoadSave ： 「Function DciLoadSave」

- DispLogMsg ：別途定義された関数
  メッセージをリストボックスとログファイルに出力します。
- GetJobNameByNo ：別途定義された関数
  作業番号に対応したジョブ名を返します。
- JobCopy ：別途定義された関数
  ジョブを指定された名前のファイルにコピーします。

- MsgBox：Visual Basicの関数
  MsgBoxはダイアログボックスにメッセージを表示し、ボタンが選択されるのを待ちます。MsgBox関数はどのボタンが選択されたかを示す値を返しますが、MsgBoxステートメントは値を返しません。
- Format$：Visual Basicの関数
  指定した書式にしたがって、数値、日付、時間、文字列などを表示する文字列処理関数です。

■ Function DciGetJobNo

```
Function DciGetJobNo (nCid As Integer, JobNo As Integer, lst As Control, LogFile As String)
As Integer
'input     nCid        通信ハンドラ
'          Lst         メッセージ出力用リスト名
'          LogFile     ログファイル名
'output    JobNo       受信したジョブ番号
' リターン値 TRUE     送信完了
'                FALSE    キャンセルまたはエラー発生
    Dim rc As Integer
    Dim rc0 As Integer
    'BscDciGetPos の戻り値を宣言
    ReDim axis6(5) As Double
    Dim datatype As Integer
    Dim RConf As Integer
    rc = False
    rc0 = -1
    ' 中止ボタンが押される（F_QUIT フラグが真になる）か、作業番号を受信するまで
受信要求を繰り返す.
    Do While Not F_QUIT
        rc0 = BscDCIGetPos(nCid, datatype, RConf, axis6(0))
        If rc0 >= 0 Then Exit Do  ' 作業番号を受信.
    Loop
    If Not F_QUIT Then
         If datatype <= 2 Then ' バイト型か整数型のみ受付け
            ' 受信した作業番号を設定.
            JobNo = axis6(0)
            rc = True
        Else
            DispLogMsg lst, " 予期しないデータタイプを受信しました．(" + Str$(datatype) +
")", LogFile
        End If
    Else
        DispLogMsg lst, " キャンセルされました．", ""
    End If
    DciGetJobNo = rc
End Function
```

- プログラム中の２重下線は、MOTOCOM32の伝送関数です。
- DispLogMsg　：MOTOCOMの関数
  メッセージをリストボックスとログファイルに出力します。

■ Function DciLoadSave

```
Function DciLoadSave (nCid As Integer, lst As Control, LogFile As String) As Integer
'input     nCid        通信ハンドラ
'           Lst         メッセージ出力用リスト名
'           LogFile     ログファイル名
'output    なし
' リターン値 TRUE        送信完了
'                FALSE       キャンセルまたはエラー発生
    Dim rc As Integer
    Dim rc0 As Integer
    rc = False
    ' 中止ボタンが押される（F_QUIT フラグが真になる）か、送信を完了するまで繰り返す.
    Do While Not F_QUIT
        rc0 = BscDCILoadSave(nCid, 1)
        If rc0 > 0 Then ' 送信完了
            rc = True
            Exit Do
        ElseIf rc0 = 0 Then  'YASNAC からの受信要求なし．再度受信要求待ちへ
        Else  ' ジョブ伝送異常が発生.
            MsgBox " ジョブ伝送異常が発生しました．(" + Format$(rc0) + ")"
            Exit Do
        End If
    Loop
    If F_QUIT = True Then
        DispLogMsg lst, " キャンセルされました．", ""
    End If
End Function
```

参考

- プログラム中の２重下線は、MOTOCOM32の伝送関数です。
- DispLogMsg　：別途定義された関数
  メッセージをリストボックスとログファイルに出力します。
- MsgBox：Visual Basicの関数
  MsgBoxはダイアログボックスにメッセージを表示し、ボタンが選択されるのを待ちます。MsgBox関数はどのボタンが選択されたかを示す値を返しますが、MsgBoxステートメントは値を返しません。

6.6 各関数のプログラムリスト

■ Function Ms_BscOpenComm()

```
'TransMode:  0...RS-232C   1...Ethernet
'FuncMode:  0...Host Control (Client), 1...DCI/Stand Alone (Server)
Function Ms_BscOpenComm(TransMode%, FuncMode% ) as Integer
    Dim ncid As Integer
    Dim rc As Integer
    Dim IPAddrress As string
    Ms_BscOpenComm = -1
    if TransMode=0 then
        ' ポートをオープンします。
        ncid = BscOpen(CurDir$, 1)
        If ncid < 0 Then GoTo Ms_BscOpenComm_Exit

        ' シリアル通信のパラメータを設定します。
        rc = BscSetCom(ncid, 1, 9600, 2, 8, 0)
    else
        'Ethernet 回線をオープンします。
        ncid = BscOpen(CurDir$, PACKETETHERNET)
        If ncid < 0 Then GoTo Ms_BscOpenComm_Exit

        'Ethernet 通信のパラメータを設定します。
        IPAddrress = "999.999.99.99"  '<--- 任意の IP アドレスを指定します
        rc = BscSetEther( ncid , IPAddrress , FuncMode, frmMain.hWnd )
    end if
        If rc <> 1 Then
        rc = BscClose(ncid)
        ncid = -1
        GoTo Ms_BscOpenComm_Exit
    End If

    ' 通信回線を接続します。
    rc = BscConnect(ncid)
    If rc <> 1 Then
        rc = BscClose(ncid)
        ncid = -1
        GoTo Ms_BscOpenComm_Exit
    End If

Ms_BscOpenComm_Exit:
    Ms_BscOpenComm = ncid

End Function
```

参考
- プログラム中の２重下線は、MOTOCOM32の伝送関数です。
- CurDir$ ：Visual Basicの関数
  指定したドライブの現在のパスを返す関数

この関数はCOMポートまたはEthernet回線をオープンし、接続が完了したら、戻り値としてハンドル値を返します。以降のMotocom32.DLLに対する操作はこのハンドル値を用いて行ないます。

■ Function Ms_BscCloseComm()

```
Function Ms_BscCloseComm(ncid as integer) as Integer
    Dim rc As Integer

    ' 通信回線を切断します。
    rc = BscDisConnect(ncid)

    ' ポートをクローズします。
    rc = BscClose(ncid)

    Ms_BscCloseComm = rc

End Sub
```

参考 プログラム中の２重下線はMOTOCOM32の伝送関数です。

■ Sub CmdDownload_Click ()

```
Sub CmdDownload_Click ()
    Dim nCid As Integer
    Dim rc As Integer
    Dim JobName As String

    ' テキストボックスからジョブ名を獲得します.
    JobName = UCase$(TxtJobName.Text)

    ' ジョブ名が指定されているかチェックします.
    If JobName <> "" Then
        ' 通信ハンドラを獲得します.
        nCid = Ms_BscOpenComm( mode, 0 )  ' mode=0 or 1
        ' 通信ハンドラを獲得できたかチェックします.
        If nCid >= 0 Then
            ' ジョブを送信します.
            rc = BscDownLoad(nCid, JobName)
            ' 戻り値を確認します.
            If rc = 0 Then
                MsgBox " ジョブ送信完了. "
            Else
                MsgBox " ジョブ送信失敗 :" + Format$(rc)
            End If

            ' 通信ハンドラを解放します.
            rc = Ms_BscCloseComm(nCid)
        Else
            MsgBox " ポートをオープンできません. "
        End If
    Else
        MsgBox " ジョブ名を指定して下さい. "
    End If

End Sub
```

参考

- プログラム中の２重下線はMOTOCOM32の伝送関数です。
- プログラム中の1重下線は、プログラムリストが記述されている関数です。
  Ms_BscOpenComm ： 「Function Ms_BscOpenComm()」
  Ms_BscCloseComm ： 「Function Ms_BscCloseComm()」

- UCase$：Visual Basicの関数
  指定した文字列中のアルファベットの小文字を大文字に変換する文字列処理関数
- MsgBox：Visual Basicの関数
  MsgBoxはダイアログボックスにメッセージを表示し、ボタンが選択されるのを待ちます。MsgBox関数はどのボタンが選択されたかを示す値を返しますが、MsgBoxステートメントは値を返しません。
- Format$：Visual Basicの関数
  指定した書式にしたがって、数値、日付、時間、文字列などを表示する文字列処理関数です。

■ Sub CmdUpLoad_Click ()

```
Sub CmdUpLoad_Click ()
    Dim nCid As Integer
    Dim rc As Integer
    Dim JobName As String

    ' テキストボックスからジョブ名を獲得します.
    JobName = UCase$(TxtJobName.Text)

    ' ジョブ名が指定されているかチェックします.
    If JobName <> "" Then
        ' 通信ハンドラを獲得します.
        nCid = Ms_BscOpenComm( mode, 0 )  ' mode=0 or 1
        ' 通信ハンドラを獲得できたかチェックします.

        If nCid >= 0 Then
            ' ジョブを受信します.
            rc = BscUpLoad(nCid, JobName)
            ' 戻り値を確認します.
            If rc = 0 Then
                MsgBox " ジョブ受信完了. "
            Else
                MsgBox" ジョブ受信失敗 :" + Format$(rc)
            End If

            ' 通信ハンドラを解放します.
            rc = Ms_BscCloseComm(nCid)

        Else
            MsgBox " ポートをオープンできません. "
        End If
    Else
        MsgBox " ジョブ名を指定して下さい. "
    End If

End Sub
```

参考

• プログラム中の２重下線はMOTOCOM32の伝送関数です。
• プログラム中の1重下線は、プログラムリストが記述されている関数です。
Ms_BscOpenComm ： 「Function Ms_BscOpenComm()」
Ms_BscCloseComm ： 「Function Ms_BscCloseComm()」

• UCase$：Visual Basicの関数
指定した文字列中のアルファベットの小文字を大文字に変換する文字列処理関数
• MsgBox：Visual Basicの関数
MsgBoxはダイアログボックスにメッセージを表示し、ボタンが選択されるのを待ちます。MsgBox関数はどのボタンが選択されたかを示す値を返しますが、MsgBoxステートメントは値を返しません。
• Format$：Visual Basicの関数
指定した書式にしたがって、数値、日付、時間、文字列などを表示する文字列処理関数です。

■ Sub CmdExit_Click ()

```
Sub CmdExit_Click ()
    'フォームをアンロードしてプログラムを終了します.
    Unload Me
End Sub
```

参考

• Unload ：Visual Basicの関数
フォームまたはコントロールをメモリからアンロードします。

■ CTestDlg::TestOpenComm 関数

```
// TransMode : 0...RS-232C   1...Ethernet
// FuncMode : 0...HostControl (Client)   1...DCI/Stand Alone (Server)
short CTestDlg::TestOpenComm( int TransMode, int FuncMode )
{
    short ncid;
    short rc;
    char cur_dir[_MAX_DIR ];
    char *IPAddress="999.999.99.99"; // 任意の IP アドレス文字列

    _getcwd( cur_dir, _MAX_DIR );
    if( TransMode==0 )
    {
        //COM ポートをオープンします。
        ncid = BscOpen( cur_dir, PACKETCOM );
        if( ncid < 0 )
         {
            return( ncid );
         }
        // シリアル通信のパラメータを設定します。
        rc = BscSetCom( ncid, 1, 9600, 2, 8, 0 );
        if( rc != 1 )
        {
            rc = BscClose( ncid );
            return( -1 );
        }
    }else
    {
        //Ethernet 回線をオープンします。
        ncid = BscOpen( cur_dir, PACKETETHERNET );
        if( ncid < 0 )
        {
            return( ncid );
        }
        //Ethernet 通信のパラメータを設定します。
        rc = BscSetEther( ncid, IPAddress, FuncMode, GetSafeHwnd() );
        if( rc != 1 )
        {
             rc = BscClose( ncid );
            return( -1 );
        }
    }
    // 通信回線を接続します。
    rc = BscConnect( ncid );
    if( rc != 1 )
    {
        rc = BscClose( ncid );
        return( -1 );
    }
```

```
    return( ncid );
}
```

参考

- プログラム中の２重下線はMOTOCOM32の伝送関数です。
- _getcwd：Visual C++の関数
  カレントパスを返す関数

■ CTestDlg::TestCloseComm 関数

```
short CTestDlg::TestCloseComm( short ncid )
{
    short rc;

    // 通信回線を切断します。
    rc = BscDisConnect( ncid );

    // ポートをクローズします。
    rc = BscClose( ncid );

    return( rc );
}
```

参考

プログラム中の２重下線は MOTOCOM32 の伝送関数です。

■ CTestDlg::OnDownload 関数

```
void  CTestDlg::OnDownload()
{
    short  nCid;
    short  rc;
    CString  JobName;

    // エディットコントロールからジョブ名を獲得します.
    m_jobname.GetWindowText(JobName);
    JobName.MakeUpper();

    // ジョブ名が指定されているかチェックします.
    if(!JobName.IsEmpty() )
    {
       // 通信ハンドラを獲得します.
      nCid = TestOpenComm ( mode, 0 ); // mode=0  or  1
      // 通信ハンドラを獲得できたかチェックします.
      if( nCid >= 0 )
      {
          // ジョブを送信します.
          rc = BscDownLoad( nCid, (char*)(LPCSTR)JobName);
          // 戻り値を確認します.
          if( rc == 0 )
              AfxMessageBox ( " ジョブ送信完了 " );
          else
              AfxMessageBox ( " ジョブ送信失敗 " );

          // 通信ハンドラを解放します.
          rc = TestCloseComm( nCid );
       }
       else
          AfxMessageBox (" ポートをオープンできません．" );
    }
    else
    {
        AfxMessageBox ( " ジョブ名を指定して下さい．" );
     }
}
```

参考

- プログラム中の２重下線はMOTOCOM32の伝送関数です。
- プログラム中の1重下線は、プログラムリストが記述されている関数です。
  TestOpenComm ： 「CTestDlg::TestOpenComm関数」
  TestCloseComm ： 「CTestDlg::TestCloseComm関数」

- MakeUpper ：Visual C++の関数
  指定した文字列中のアルファベットの小文字を大文字に変換する文字列処理関数
- AfxMessageBos：Visual C++の関数
  ダイアログボックスにメッセージを表示し、ボタンが選択されるのを待ちます。

■ CTestDlg::OnUpload 関数

```
void  CTestDlg::OnUpload()
{
    short   nCid;
    short   rc;
    CString JobName;

    // エディットコントロールからジョブ名を獲得します.
    m_jobname.GetWindowText(JobName);
    JobName.MakeUpper();

    // ジョブ名が指定されているかチェックします.
    if(!JobName.IsEmpty() )
    {
        // 通信ハンドラを獲得します.
        nCid = TestOpenComm ( mode, 0 ); // mode=0  or  1
        // 通信ハンドラを獲得できたかチェックします.
        if( nCid >= 0 )
         {
            // ジョブを受信します.
            rc = BscUpLoad(nCid, (char*)(LPCSTR)JobName);
            // 戻り値を確認します.
            if( rc == 0 )
                AfxMessageBox ( " ジョブ受信完了 " );
            else
                AfxMessageBox(" ジョブ受信失敗 " );

            // 通信ハンドラを解放します.
            rc = TestCloseComm( nCid );
        }
        else
            AfxMessageBox ( " ポートをオープンできません．" );
    }
    else
    {
        AfxMessageBox ( " ジョブ名を指定して下さい．" );
    }
}
```

参考

- プログラム中の２重下線はMOTOCOM32の伝送関数です。
- プログラム中の1重下線は、プログラムリストが記述されている関数です。
  TestOpenComm ： 「CTestDlg::TestOpenComm関数」
  TestCloseComm ： 「CTestDlg::TestCloseComm関数」

- MakeUpper ：Visual C++の関数
  指定した文字列中のアルファベットの小文字を大文字に変換する文字列処理関数
- AfxMessageBos：Visual C++の関数
  ダイアログボックスにメッセージを表示し、ボタンが選択されるのを待ちます。

■ private short Ms_BscOpenComm()

```
// mode: 0...RS-232C 1...Ethernet
private short Ms_BscOpenComm( int mode )
{
   short nCid;
   short rc;
   byte[] IPAddress;

   // バイト型配列へ変換
   byte[] path = Encoding.Default.GetBytes( Application.StartupPath );

   if( mode == 0 )
   {
      // COM ポートをオープンします。
      nCid = MotoCom32.BscOpen( ref path[ 0 ], (short)MotoCom32.PACKET.COM );

      if( nCid < 0 )
      {
         return nCid;
      }

      // シリアル通信のパラメータを設定します。
      rc = MotoCom32.BscSetCom( nCid, 1, 9600, 2, 8, 0 );
   }
   else
   {
      // Ethernet 回線をオープンします。
      nCid = MotoCom32.BscOpen( ref path[ 0 ], (short)MotoCom32.PACKET.ETHERNET );

      if( nCid < 0 )
      {
         return nCid;
      }

      // Ethernet 通信のパラメータを設定します。
      // 任意の IP アドレスを指定します。
      IPAddress = Encoding.Default.GetBytes( "192.168.0.10" );

      rc = MotoCom32.BscSetEther( nCid, ref IPAddress[ 0 ], 0, this.Handle );
   }

   if( rc != 1 )
   {
      rc = MotoCom32.BscClose( nCid );
      return -1;
   }

   // 通信回線を接続します。
   rc = MotoCom32.BscConnect( nCid );
```

```
    if( rc != 1 )
    {
      rc = MotoCom32.BscClose( nCid );
      return -1;
    }

    return nCid;
  }
```

参考
- プログラム中の２重下線はMOTOCOM32の伝送関数です。
- Encoding.Default.GetBytes() ：Visual Studio C# の関数
  カッコ内の文字列をバイト配列に変換します。
- Application.StartupPath ： Visual Studio C# の関数
  アプリケーションを開始した実行可能ファイルの存在するフォルダを取得します。

この関数は COM ポートまたは Ethernet 回線をオープンし、接続が完了したら、戻り値としてハンドル値を返します。以降の Motocom32.DLL に対する操作はこのハンドル値を用いて行います。

■ private short Ms_BscCloseComm()

```
private short Ms_BscCloseComm( short nCid )
{
    short rc;

    // 通信回線を切断します。
    rc = MotoCom32.BscDisConnect( nCid );

    // ポートをクローズします。
    rc = MotoCom32.BscClose( nCid );

    return rc;
}
```

参考 プログラム中の２重下線は MOTOCOM32 の伝送関数です

■ private void CmdDownLoad_Click()

```
private void CmdDownLoad_Click( object sender, EventArgs e )
{
  short nCid;
  short rc;
  byte[] jobname;

  // テキストボックスからジョブ名を獲得します.
  jobname = Encoding.Default.GetBytes( TxtJobName.Text.ToUpper() );

  // ジョブ名が指定されているかチェックします.
  if( jobname.Length > 0 )
  {
```

```
        // 伝送モードを指定します mode: 0:RS-232C, 1:Ethernet
        nCid = Ms_BscOpenComm( 1 );

        // 通信ハンドラを獲得できたかチェックします.
        if( nCid != -1 )
        {
          // ジョブを送信します.
          rc = MotoCom32.BscDownLoad( nCid, ref jobname[ 0 ] );

          // 戻り値を確認します.
          if( rc == 0 )
          {
            MessageBox.Show( " ジョブ送信完了. " );
          }
          else
          {
            MessageBox.Show( " ジョブ送信失敗 :" + rc.ToString() );
          }

          // 通信ハンドラを解放します.
          rc = Ms_BscCloseComm( nCid );
        }
        else if( nCid == -8 )
        {
          MessageBox.Show( " ライセンスエラー " );
        }
        else
        {
          MessageBox.Show( " ポートをオープンできません. " );
        }
      }
      else
      {
        MessageBox.Show( " ジョブ名を指定して下さい. " );
      }
    }
```

参考

- プログラム中の２重下線は、MOTOCOM32の伝送関数です。
- プログラム中の１重下線は、プログラムリストが記述されている関数です。
  - Ms_BscOpenComm　：「private short Ms_BscOpenComm()」
  - Ms_BscCloseComm　：「private short Ms_BscCloseComm()」
- ToUpper()　：　Visual Studio C# の関数
  文字列を大文字に変換します。
- MessageBox.Show()：Visual Studio C# の関数
  ダイアログボックスにメッセージを表示し、ボタンが選択されるのを待ちます。
- ToString()　：　Visual Studio C# の関数
  指定した書式、カルチャ固有の書式情報を使用して、文字列形式に変換する文字列処理関数です。

■ private void CmdUpLoad_Click()

```
private void CmdUpLoad_Click( object sender, EventArgs e )
{
   short nCid;
   short rc;
   byte[] jobname;

   // テキストボックスからジョブ名を獲得します.
   jobname = Encoding.Default.GetBytes( TxtJobName.Text.ToUpper() );

   // ジョブ名が指定されているかチェックします.
   if( jobname.Length > 0 )
   {
      // 伝送モードを指定します mode: 0:RS-232C, 1:Ethernet
      nCid = Ms_BscOpenComm( 1 );

      // 通信ハンドラを獲得できたかチェックします.
      if( nCid >= 0 )
      {
         // ジョブを受信します.
         rc = MotoCom32.BscUpLoad( nCid, ref jobname[ 0 ] );

         // 戻り値を確認します.
         if( rc == 0 )
         {
            MessageBox.Show( " ジョブ受信完了. " );
         }
         else
         {
            MessageBox.Show( " ジョブ受信失敗 :" + rc.ToString() );
         }

         // 通信ハンドラを解放します.
         rc = Ms_BscCloseComm( nCid );
      }
      else if( nCid == -8 )
      {
         MessageBox.Show( " ライセンスエラー " );
      }
      else
      {
         MessageBox.Show( " ポートをオープンできません. " );
      }
   }
   else
   {
      MessageBox.Show( " ジョブ名を指定して下さい. " );
   }
```

```
}
```

参考

- プログラム中の２重下線は、MOTOCOM32の伝送関数です。
- プログラム中の１重下線は、プログラムリストが記述されている関数です。
  Ms_BscOpenComm　：「private short Ms_BscOpenComm()」
  Ms_BscCloseComm　：「private short Ms_BscCloseComm()」
- ToUpper() ：　Visual Studio C# の関数
  文字列を大文字に変換します。
- MessageBox.Show()：Visual Studio C# の関数
  ダイアログボックスにメッセージを表示し、ボタンが選択されるのを待ちます。
- ToString() ：　Visual Studio C# の関数
  指定した書式、カルチャ固有の書式情報を使用して、文字列形式に変換する文字列処理関数です。

■　private void CmdExit_Click()

```
private void CmdExit_Click( object sender, EventArgs e )
{
    // フォームを閉じてプログラムを終了します.
    this.Close();
}
```

参考

- this.Close() ：Visual Studio C# の関数
  フォームを閉じます。

7 伝送関数説明

7.1 概要

「Motocom32.DLL」は YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、YASNAC XRC、MRC、ERC、ERC2 のデータ伝送機能をパソコンで制御する伝送ライブラリです。このライブラリは Microsoft Windows の DLL（ダイナミックリンクライブラリ）形式で作成されています。

「Motocom32.DLL」は MOTOCOM32 をインストールしたフォルダに入っています。伝送アプリケーションを作成した場合はこのファイルをアプリケーションと同じフォルダへコピーしてご利用ください。また、MOTOCOM32 インストールディレクトリに「Motocom.H」、「Motocom32.LIB」が入っています。C 言語で伝送アプリケーションを作成する場合はご利用ください。

伝送ライブラリーは以下の機能に分類されます。

- ファイルデータ伝送機能
- ロボット制御機能
- DCI 機能
- I/O 信号のリード・ライト機能
- その他の機能

7.2 ファイルデータ伝送機能

ジョブや条件データ、システム情報などのファイルをロード・セーブします。
以下の関数が使用できます。

BscDownload
BscDownloadEx
BscUpload
BscUploadEx

■ BscDownload

機 能	指定ファイルをロボットコントローラへ送信します。
書 式	_declspec(dllexport) short APIENTRY BscDownLoad(short nCid,char *fname);
引 数	**IN（引き渡し）** nCid　　通信ハンドラの ID 番号 *fname　送信ファイル名
	OUT（戻り） 無し
	リターン値 0　　：正常終了 その他：送信エラー
コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC（シリアルポート、Ethernet） ERC（シリアルポート）
参 照	「BscDownloadEx」「BscUpload」「BscUploadEx」

■ BscDownloadEx

機 能	指定ファイルをロボットコントローラへ送信します。送信ファイルのあるディレクトリを指定できます。
書 式	_declspec(dllexport) short APIENTRY BscDownLoadEx(short nCid,char *fname, char *path, BOOL nFlg);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号 *fname　送信ファイル名 *path　送信元データのディレクトリパス nFlg　TRUE ：一時的に対象ディレクトリを変更し、最後に元に戻す。 FALSE：対象ディレクトリを変更しそのまま処理を完了する。
	OUT（戻り） 無し
	リターン値 0　：正常終了 その他：送信エラー
コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC（シリアルポート、Ethernet） ERC（シリアルポート）
参 照	「BscDownload」「BscUpload」「BscUploadEx」

■ BscUpload

機 能	指定ファイルをロボットコントローラから受信します。
書 式	_declspec(dllexport) short APIENTRY BscUpLoad(short nCid,char *fname);
引 数	**IN（引き渡し）** nCid　　通信ハンドラの ID 番号 *fname　受信ファイル名
	OUT（戻り） 無し
	リターン値 0　　：正常終了 その他：受信エラー
コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC（シリアルポート、Ethernet） ERC（シリアルポート）
参 照	「BscUploadEx」「BscDownload」「BscDownloadEx」

■ BscUploadEx

機 能	指定ファイルをロボットコントローラから受信します。受信先のディレクトリを指定できます。
書 式	_declspec(dllexport) short APIENTRY BscUpLoadEx(short nCid,char *fname, char *path, BOOL nFlg);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号 *fname　受信ファイル名 *path　送信元データのディレクトリパス nFlg　TRUE　：一時的に対象ディレクトリを変更し、最後に元に戻す。 FALSE：対象ディレクトリを変更しそのまま処理を完了する
	OUT（戻り） 無し
	リターン値 0　：正常終了 その他：受信エラー
コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC（シリアルポート、Ethernet） ERC（シリアルポート）
参 照	「BscUpload」「BscDownload」「BscDownloadEx」

7.3 ロボット制御機能

ロボットのステータス（現在位置、アラーム・エラー・サーボなどの状態）のリードや、システムのコントロール（スタート、ホールド、ジョブの呼び出しなど）を行います。
以下の関数が使用できます。

ステータスのリード

BscFindFirst
BscFindFirstMaster
BscFindNext
BscFindNextMaster
BscGetCtrlGroup
BscGetCtrlGroupXrc
BscGetCtrlGroupDX
BscDownload
BscDownloadEx
BscGetError
BscGetError2
BscGetFirstAlarm
BscGetFirstAlarmS
BscGetNextAlarm
BscGetNextAlarmS
BscGetStatus
BscGetUFrame
BscGetVarData
BscGetVarData2
BscHostGetVarData
BscHostGetVarDataM
BscGetVarDataEx
BscIsAlarm
BscIsCtrlGroup
BscIsCtrlGroupXrc
BscIsCtrlGroupDX
BscIsCycle
BscIsError
BscIsErrorCode
BscIsHold
BscIsJobLine
BscIsJobName
BscIsJobStep
BscIsLoc
BscGetPulsePos
BscIsPlayMode
BscIsRemoteMode
BscIsRobotPos
BscGetCartPos
BscIsServo
BscIsTaskInf
BscIsTaskInfXrc
BscIsTeachMode
BscJobWait
BscReadAlarmS
BscGetTorque
BscGetMaxTorque
BscGetEncTmp
BscGetSysTime

システムのコントロール

BscCancel
BscChangeTask
BscContinueJob
BscConvertJobP2R
BscConvertJobR2P
BscDeleteJob
BscHoldOff
BscHoldOn
BscHostPutVarData
BscHostPutVarDataM
BscPutVarDataEx
BscImov
BscImovEx
BscMDSP
BscMov
BscMovEx
BscMovj
BscMovjEx
BscMovl
BscMovlEx
BscOPLock
BscOPUnLock
BscPMov
BscPMovEx
BscPMovj
BscPMovjEx
BscPMovl
BscPMovlEx
BscPutUFrame
BscPutVarData
BscPutVarData2
BscStartJob
BscSelectJob
BscSelectMode
BscSelLoopCycle
BscSelOneCycle
BscSelStepCycle
BscSetLineNumber
BscSetMasterJob
BscReset
BscSetCtrlGroup
BscSetCtrlGroupXrc
BscSetCtrlGroupDX
BscServoOff
BscServoOn
BscUpload
BscUploadEx

■ BscFindFirst

機 能	現在登録されている全ジョブリストの第 1 個目のジョブ名を取得します。
書 式	_declspec(dllexport) short APIENTRY BscFindFirst(short nCid,char *fname,short size);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号 *fname　第 1 個目のジョブ名格納ポインタ size　ジョブ名格納エリアのサイズ
	OUT（戻り） *fname　第 1 個目のジョブ名格納ポインタ
	リターン値 -1：ジョブ無し -2：内部エラー（メモリーアロケーションエラー） -3：内部エラー（メモリーロックエラー） -4：その他のエラー 0：ジョブ有り
コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC（シリアルポート、Ethernet） ERC（シリアルポート）
参 照	「BscFindNext」

■ BscFindFirstMaster

機 能	対象ジョブに付属するジョブリストうち、最初のジョブ名を取得します。
書 式	_declspec(dllexport) short APIENTRY BscFindFirstMaster(short nCid,char *fname,short size);
引 数	IN（引き渡し） nCid　通信ハンドラの ID 番号 *fname　第 1 個目のジョブ名格納ポインタ size　ジョブ名格納エリアのサイズ
	OUT（戻り） *fname　第 1 個目のジョブ名格納ポインタ
	リターン値 -1：ジョブ無し -2：内部エラー（メモリーアロケーションエラー） -3：内部エラー（メモリーロックエラー） -4：その他のエラー 0：ジョブ有り
解 説	**呼び出し条件** 本関数を実行する事前に BscSelectJob 関数をコールして、対象ジョブ名の指定をしておく必要があります。
コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC（シリアルポート、Ethernet） ERC（シリアルポート）
参 照	「BscFindNextMaster」「BscSelectJob」

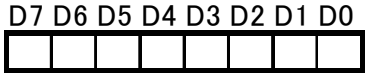
■ BscFindNext

機 能	現在登録されている次のジョブ名を取得します。
書 式	_declspec(dllexport) short APIENTRY BscFindNext(short nCid,char *fname,short size);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号 *fname　第 n 個目のジョブ名格納ポインタ size　ジョブ名格納エリアのサイズ
	OUT（戻り） *fname　第 n 個目のジョブ名格納ポインタ
	リターン値 -1：次のジョブ無し 0：次のジョブ有り
解 説	**呼び出し条件** 本関数を実行する事前に BscFindFirst 関数をコールしておく必要があります。
コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC（シリアルポート、Ethernet） ERC（シリアルポート）
参 照	「BscFindFirst」

■ BscFindNextMaster

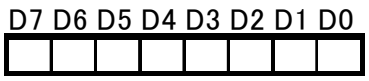
機 能	対象ジョブに付属するジョブリストうち、次のジョブ名を取得します。
書 式	_declspec(dllexport) short APIENTRY BscFindNextMaster(short nCid,char *fname,short size);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号 *fname　第 n 個目のジョブ名格納ポインタ size　ジョブ名格納エリアのサイズ
	OUT（戻り） *fname　第 n 個目のジョブ名格納ポインタ
	リターン値 -1：次のジョブ無し 0：次のジョブ有り
解 説	**呼び出し条件** 本関数を実行する事前に BscFindFirstMaster 関数をコールしておく必要があります。
コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC（シリアルポート、Ethernet） ERC（シリアルポート）
参 照	「BscFindFirstMaster」

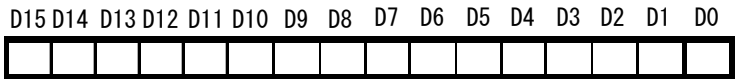
■ BscGetCtrlGroup

機 能	制御グループ、タスク情報を読み込みます。
書 式	_declspec(dllexport) short APIENTRY BscGetCtrlGroup(short nCid,short *groupinf,short *taskinf);
引 数	**IN（引き渡し）** nCid　　通信ハンドラの ID 番号 *groupinf　制御グループ情報格納ポインタ *taskinf　　タスク情報格納ポインタ
	OUT（戻り） *groupinf　制御グループ情報格納ポインタ *taskinf　　タスク情報格納ポインタ
	リターン値 0　　：正常終了 その他：エラーコード
解 説	**制約事項** 本関数は MRC に対して伝送を行う場合に有効です。 YRC1000、YRC1000micro、DX200、DX100　対して伝送を行う場合は、BscGetCtrlGroupDX を参照してください。 FS100、NX100、XRC 対して伝送を行う場合は、BscGetCtrlGroupXrc を参照してください。
	制御グループ情報 制御グループ情報は、以下のビットデータを 10 進数にした値となります。  D0：R1（ロボット1） D1：R2（ロボット2） D2：S1（ステーション1） D3：S2（ステーション2） D4：S3（ステーション3） D5：S4（ステーション4） D6：S5（ステーション5） D7：S6（ステーション6）
	タスク情報 タスク情報は、以下のようになります。 0：マスタータスク 1：サブ１タスク 2：サブ２タスク なお、独立制御が許可されていないシステムでは、「0」を返します。
コントローラ	MRC（シリアルポート、Ethernet）

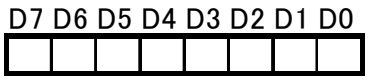
参 照	「BscGetCtrlGroupDX」 「BscSetCtrlGroupDX」 「BscIsCtrlGroupDX」 「BscGetCtrlGroupXrc」 「BscSetCtrlGroupXrc」 「BscIsCtrlGroupXrc」 「BscIsTaskInfXrc」 「BscGetCtrlGroup」 「BscIsCtrlGroup」 「BscIsTaskInf」 「BscChangeTask」

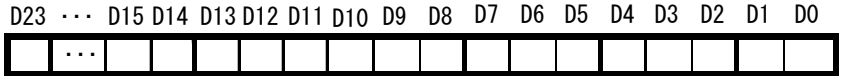
■ BscGetCtrlGroupXrc

機 能	制御グループ、タスク情報を読み込みます。
書 式	_declspec(dllexport) short APIENTRY BscGetCtrlGroupXrc(short nCid,short *groupinf,short *stationinf,short *taskinf);
引 数	**IN（引き渡し）** nCid　　通信ハンドラの ID 番号 *groupinf　制御グループ情報格納ポインタ（ロボット軸） *stationinf　制御グループ情報格納ポインタ（ステーション軸） *taskinf　　タスク情報格納ポインタ
	OUT（戻り） *groupinf　制御グループ情報格納ポインタ（ロボット軸） *stationinf　制御グループ情報格納ポインタ（ステーション軸） *taskinf　　タスク情報格納ポインタ
	リターン値 0　　：正常終了 その他：エラーコード
解 説	**制約事項** 本関数は FS100、NX100、XRC に対して伝送を行う場合に有効です。 YRC1000、YRC1000micro、DX200、DX100 に対しては伝送を行う場合は、BscGetCtrlGroupDX を参照してください。 MRC に対しては伝送を行う場合は、BscGetCtrlGroup を参照してください。
	制御グループ情報（ロボット軸） 制御グループ情報は、以下のビットデータを 10 進数にした値となります。  D0：R1（ロボット1） D1：R2（ロボット2） D2：R3（ロボット3） D3：R4（ロボット4）

解 説	**制御グループ情報（ステーション軸）** 制御グループ情報は、以下のビットデータを 10 進数にした値となります。  D0：S1（ステーション1） D1：S2（ステーション2） D2：S3（ステーション3） D3：S4（ステーション4） D4：S5（ステーション5） D5：S6（ステーション6） D6：S7（ステーション7） D7：S8（ステーション8） D8：S9（ステーション9） D9：S10（ステーション10） D10：S11（ステーション11） D11：S12（ステーション12）
	タスク情報 タスク情報は、以下のようになります。 0：マスタータスク 1：サブ 1 タスク 2：サブ 2 タスク 3：サブ 3 タスク 4：サブ 4 タスク 5：サブ 5 タスク 6：サブ 6 タスク 7：サブ 7 タスク なお、独立制御が許可されていないシステムでは、「0」を返します。
コントローラ	FS100、NX100、XRC（シリアルポート、Ethernet）
参 照	「BscGetCtrlGroupDX」「BscSetCtrlGroupDX」「BscIsCtrlGroupDX」「BscSetCtrlGroupXrc」「BscIsCtrlGroupXrc」「BscIsTaskInfXrc」「BscGetCtrlGroup」「BscSetCtrlGroup」「BscIsCtrlGroup」「BscIsTaskInf」「BscChangeTask」

■ BscGetCtrlGroupDX

機 能	制御グループ、タスク情報を読み込みます。
書 式	_declspec(dllexport) short APIENTRY BscGetCtrlGroupDX(short nCid,long *groupinf,long *stationinf,short *taskinf);
引 数	**IN（引き渡し）** nCid　　通信ハンドラの ID 番号 *groupinf　制御グループ情報格納ポインタ（ロボット軸） *stationinf　制御グループ情報格納ポインタ（ステーション軸） *taskinf　　タスク情報格納ポインタ
	OUT（戻り） *groupinf　制御グループ情報格納ポインタ（ロボット軸） *stationinf　制御グループ情報格納ポインタ（ステーション軸） *taskinf　　タスク情報格納ポインタ
	リターン値 0　　：正常終了 その他：エラーコード
解 説	**制約事項** 本関数は YRC1000、YRC1000micro、DX200、DX100 に対して伝送を行う場合に有効です。 FS100、NX100、XRC に対しては伝送を行う場合は、BscGetCtrlGroupXrc を参照してください。 MRC に対しては伝送を行う場合は、BscGetCtrlGroup を参照してください。
	制御グループ情報（ロボット軸） 制御グループ情報は、以下のビットデータを 10 進数にした値となります。  D0：R1（ロボット1） D1：R2（ロボット2） D2：R3（ロボット3） D3：R4（ロボット4） ： ： D7：R8（ロボット8）

解 説	**制御グループ情報（ステーション軸）** 制御グループ情報は、以下のビットデータを 10 進数にした値となります。  D0：S1（ステーション1） D1：S2（ステーション2） D2：S3（ステーション3） D3：S4（ステーション4） D4：S5（ステーション5） D5：S6（ステーション6） D6：S7（ステーション7） D7：S8（ステーション8） D8：S9（ステーション9） D9：S10（ステーション10） D10：S11（ステーション11） D11：S12（ステーション12） ： ： D23：S24（ステーション 24）
	タスク情報 タスク情報は、以下のようになります。 0：マスタータスク 1：サブ 1 タスク 2：サブ 2 タスク 3：サブ 3 タスク 4：サブ 4 タスク 5：サブ 5 タスク 6：サブ 6 タスク 7：サブ 7 タスク なお、独立制御が許可されていないシステムでは、「0」を返します。
コントローラ	YRC1000、YRC1000micro、DX200、DX100（シリアルポート、Ethernet）
参 照	「BscSetCtrlGroupDX」 「BscIsCtrlGroupDX」 「BscGetCtrlGroupXrc」 「BscSetCtrlGroupXrc」 「BscIsCtrlGroupXrc」 「BscIsTaskInfXrc」 「BscGetCtrlGroup」 「BscSetCtrlGroup」 「BscIsCtrlGroup」 「BscIsTaskInf」 「BscChangeTask」

■ BscGetError

機 能	エラーコード・アラームコードを取得します。
書 式	_declspec(dllexport) short APIENTRY BscGetError(short nCid);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号
	OUT（戻り） 無し
	リターン値 -1　　：エラーコード取得失敗 0　　：エラー無し その他：エラーコード
解 説	**制約事項** 本関数は ERC に対して伝送を行う場合に有効です。 YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC に対しては伝送を行う場合は、BscGetError2 を参照してください。
コントローラ	ERC（シリアルポート）
参 照	「BscGetFirstAlarm」「BscGetNextAlarm」「BscIsAlarm」「BscIsError」「BscIsErrorCode」

■ BscGetError2

機 能	エラーコード・アラームコードを取得します。
書 式	_declspec(dllexport) short APIENTRY BscGetError2(short nCid);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号
	OUT（戻り） 無し
	リターン値 -1　　：エラーコード取得失敗 0　　：エラー無し その他：エラーコード
解 説	**制約事項** 本関数は YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC に対して伝送を行う場合に有効です。 ERC に対しては伝送を行う場合は、BscGetError を参照してください。
コントローラ	YRC1000、YRC1000micro、DX200、DX100、NX100、XRC、MRC（シリアルポート、Ethernet）
参 照	「BscGetFirstAlarm」「BscGetNextAlarm」「BscIsAlarm」「BscIsError」「BscIsErrorCode」

■ BscGetFirstAlarm

機 能	アラームコードを読み取り、アラームコードとアラームデータを返します。
書 式	_declspec(dllexport) short APIENTRY BscGetFirstAlarm(short nCid,short *data);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号 *data　アラームデータ格納ポインタ
	OUT（戻り） *data　アラームデータ格納ポインタ
	リターン値 0　　：アラーム無し その他：アラームコード番号
解 説	本関数を実行する事前に BscGetError2 関数をコールしておく必要があります。
コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC（シリアルポート、Ethernet） ERC（シリアルポート）
参 照	「BscGetError2」「BscGetNextAlarm」「BscIsAlarm」

■ BscGetFirstAlarmS

機 能	アラームコードを読み取り、アラームコードとアラームデータ、及びアラームメッセージを返します。
書 式	_declspec(dllexport) short APIENTRY BscGetFirstAlarmS(short nCid,short *data,char *msg);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号 *data　アラームデータ格納ポインタ *msg　アラームメッセージ格納ポインタ
	OUT（戻り） *data　アラームデータ格納ポインタ *msg　アラームメッセージ格納ポインタ
	リターン値 0　　：アラーム無し その他：アラームコード番号
解 説	**呼び出し条件** 本関数を実行する事前に BscReadAlarmS 関数をコールしておく必要があります。
	制約事項 本関数は YRC1000、YRC1000micro、DX200、DX100、FS100、NX100 に対して伝送を行う場合に有効です。
コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100（シリアルポート、Ethernet）
参 照	「BscReadAlarmS」「BscGetNextAlarmS」「BscIsAlarm」

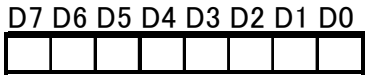
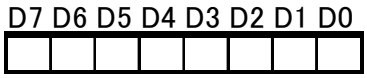
■ BscGetNextAlarm

機 能	次のアラームコードとアラームデータを返します。
書 式	_declspec(dllexport) short APIENTRY BscGetNextAlarm(short nCid,short *data);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号 *data　アラームデータ格納ポインタ
	OUT（戻り） *data　アラームデータ格納ポインタ
	リターン値 0　　：アラーム無し その他：アラームコード番号
解 説	**呼び出し条件** 本関数を実行する事前に BscGetFirstAlarm 関数をコールしておく必要があります。
コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC（シリアルポート、Ethernet） ERC（シリアルポート）
参 照	「BscGetError2」「BscGetFirstAlarm」「BscIsAlarm」

■ BscGetNextAlarmS

機 能	次のアラームコードとアラームデータ、及びアラームメッセージを返します。
書 式	_declspec(dllexport) short APIENTRY BscGetNextAlarmS(short nCid,short *data,char *msg);
引 数	IN（引き渡し） nCid　通信ハンドラの ID 番号 *data　アラームデータ格納ポインタ *char　アラームメッセージ格納ポインタ
	OUT（戻り） *data　アラームデータ格納ポインタ *msg　アラームメッセージ格納ポインタ
	リターン値 0　　：アラーム無し その他：アラームコード番号
解 説	**呼び出し条件** 本関数を実行する事前に BscGetFirstAlarmS 関数をコールしておく必要があります。
	制約事項 本関数は YRC1000、YRC1000micro、DX200、DX100、FS100、NX100 に対して伝送を行う場合に有効です。
コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100（シリアルポート、Ethernet）
参 照	「BscReadAlarmS」「BscGetFirstAlarmS」「BscIsAlarm」

■ BscGetStatus

機 能	ロボットからのステータスの取得を行います。
書 式	_declspec(dllexport) short APIENTRY BscGetStatus(short nCid,short *d1,short *d2);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号 *d1　データ 1 格納ポインタ *d2　データ 2 格納ポインタ
	OUT（戻り） *d1　データ 1 格納ポインタ *d2　データ 2 格納ポインタ
	リターン値 -1　　：取得失敗 その他：正常終了
解 説	**データ 1** データ 1 は、以下のビットデータを 10 進数にした値となります。  D0：ステップ D1：1サイクル D2：連続運転 D3：運転中 D4：柵内運転 D5：ティーチ *注 D6：プレイ *注 D7：コマンドリモート *注 *注）YRC1000、YRC1000micro、DX200、DX100、NX100、XRC MRCのみ有効です。
	データ 2 データ 2 は、以下のビットデータを 10 進数にした値となります。  D0：ホールド中（YRC1000、YRC1000micro、DX200、DX100、NX100、XRC MRC：プレイバックボックス ホールド ERC：パネルホールド） D1：ホールド中（YRC1000、YRC1000micro、DX200、DX100、NX100、XRC MRC：プログラミングペンダント ホールド ERC：T-BOXホールド） D2：ホールド中（外部ホールド） D3：ホールド中（コマンドホールド） D4：アラーム発生中 D5：エラー発生中 D6：サーボオン

コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC（シリアルポート、Ethernet） ERC（シリアルポート）
参 照	「BscStatus」「BscIsAlarm」「BscIsCycle」「BscIsHold」「BscIsPlayMode」「BscIsRemoteMode」「BscIsServo」「BscIsTeachMode」

■ BscGetUFrame

機 能	指定のユーザ座標データを読み取ります。
書 式	_declspec(dllexport) short APIENTRY BscGetUFrame(short nCid,char *ufname,double *p);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号 *ufname　読み取るユーザフレーム名称格納ポインタ *p　ユーザフレームデータ格納ポインタ
	OUT（戻り） *p　ユーザフレームデータ格納ポインタ
	リターン値 -1　：ユーザフレーム名称エラー 0　：正常終了 その他：エラーコード
解 説	**制約事項** 7 軸以上のロボットには対応していません。
	ユーザフレーム名称 ユーザフレーム番号に対応する以下の名称を指定します。

ユーザ座標名称	指定名称
ユーザ座標 1	UF1
ユーザ座標 2	UF2
ユーザ座標 3	UF3
:	:
:	:
ユーザ座標 60	UF60
ユーザ座標 61	UF61
ユーザ座標 62	UF62
ユーザ座標 63	UF63

*ユーザフレーム番号の 9～63 番までは YRC1000、YRC1000micro、DX200 DX100 のみ有効です。
*ユーザフレーム番号の 9～16 番までは FS100 のみ有効です。
*ユーザフレーム番号の 9～24 番までは NX100、XRC、MRC のみ有効です。

解 説

変数タイプ

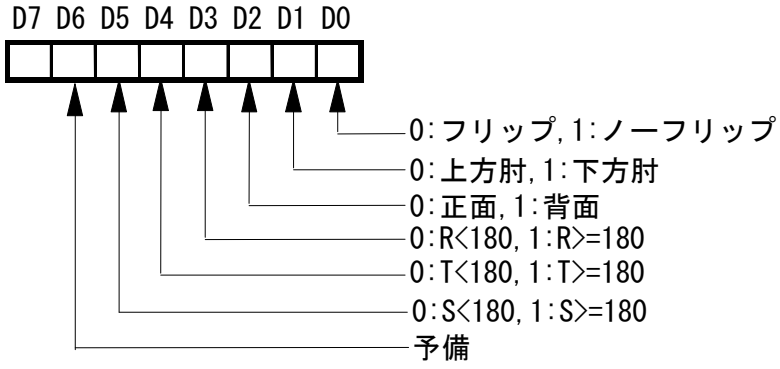
ユーザフレームデータには、ユーザフレーム番号で指定したユーザ座標系の座標値が以下のように割り当てられます。

変数	座標系	意　味
P[0]	ORG	X 座標値（単位 mm、小数点第 3 位有効）
P[1]		Y 座標値（単位 mm、小数点第 3 位有効）
P[2]		Z 座標値（単位 mm、小数点第 3 位有効）
P[3]		手首姿勢 Rx（単位°、小数点第 2 位有効）*1
P[4]		手首姿勢 Ry（単位°、小数点第 2 位有効）*1
P[5]		手首姿勢 Rz（単位°、小数点第 2 位有効）*1
P[6]		形態
P[7]	XX	X 座標値（単位 mm、小数点第 3 位有効）
P[8]		Y 座標値（単位 mm、小数点第 3 位有効）
P[9]		Z 座標値（単位 mm、小数点第 3 位有効）
P[10]		手首姿勢 Rx（単位°、小数点第 2 位有効）*1
P[11]		手首姿勢 Ry（単位°、小数点第 2 位有効）*1
P[12]		手首姿勢 Rz（単位°、小数点第 2 位有効）*1
P[13]		形態
P[14]	XY	X 座標値（単位 mm、小数点第 3 位有効）
P[15]		Y 座標値（単位 mm、小数点第 3 位有効）
P[16]		Z 座標値（単位 mm、小数点第 3 位有効）
P[17]		手首姿勢 Rx（単位°、小数点第 2 位有効）*1
P[18]		手首姿勢 Ry（単位°、小数点第 2 位有効）*1
P[19]		手首姿勢 Rz（単位°、小数点第 2 位有効）*1
P[20]		形態
P[21]	ツール番号（0～23）	
P[22]	第 7 軸パルス番号（走行軸の場合、単位は mm）	
P[23]	第 8 軸パルス番号（走行軸の場合、単位は mm）	
P[24]	第 9 軸パルス番号（走行軸の場合、単位は mm）	
P[25]	第 10 軸パルス番号	
P[26]	第 11 軸パルス番号	
P[27]	第 12 軸パルス番号	

1 YRC1000、YRC1000micro、DX200、DX100 の場合は「小数点第 4 位有効」となります。

形態

形態のデータは、以下のビットデータを 10 進数にした値となります。



コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC（シリアルポート、Ethernet） ERC（シリアルポート）
参 照	「BscPutUFrame」

■ BscGetVarData

機 能	変数情報を受信します。
書 式	_declspec(dllexport) short APIENTRY BscGetVarData(short nCid,short type,short varno,double *p);
引 数	**N（引き渡し）** nCid　通信ハンドラの ID 番号 type　変数タイプ varno　変数番号 *p　数値変数格納エリアの先頭ポインタ
	OUT（戻り） *p　数値変数格納エリアの先頭ポインタ
	リターン値 0　：正常終了 その他：エラーコード
解 説	**制約事項** 本関数は YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC に対して伝送を行う場合に有効です。
	変数タイプ 変数タイプは、以下のようになります。 0：バイト型変数 1：整数型変数 2：倍精度型変数 3：実数型変数 4：ロボット軸位置型変数 5：ベース軸位置型変数 6：ステーション軸位置型変数（パルス型のみ）

解 説

数値変数格納エリアの内容

数値変数格納エリアの内容、有効配列数は変数タイプによって以下のようになります。

変数ﾀｲﾌﾟ番号	ﾃﾞｰﾀ型(ﾊﾟﾙｽ/直交)	有効配列数	内 容				
			p[0]	p[1]	p[2]	p[3]	p[4]
0	―	1	ﾊﾞｲﾄ値	―	―	―	―
1	―	1	整数値	―	―	―	―
2	―	1	倍精度整数値	―	―	―	―
3	―	1	実数値	―	―	―	―
4	ﾊﾟﾙｽ型	8	0	S 軸ﾊﾟﾙｽ数	L 軸ﾊﾟﾙｽ数	U 軸ﾊﾟﾙｽ数	R 軸ﾊﾟﾙｽ数
4	直交型	10	1	座標種別	X 軸座標値(mm)	Y 軸座標値(mm)	Z 軸座標値(mm)
5	ﾊﾟﾙｽ型	8	0	ﾍﾞｰｽ第 1 軸ﾊﾟﾙｽ数	ﾍﾞｰｽ第 2 軸ﾊﾟﾙｽ数	ﾍﾞｰｽ第 3 軸ﾊﾟﾙｽ数	ﾍﾞｰｽ第 4 軸ﾊﾟﾙｽ数
5	直交型	10	1	座標種別	X 軸座標値(mm)	Y 軸座標値(mm)	Z 軸座標値(mm)
6	ﾊﾟﾙｽ型	8	0	ｽﾃｰｼｮﾝ第 1 ﾊﾟﾙｽ数	ｽﾃｰｼｮﾝ第 2 ﾊﾟﾙｽ数	ｽﾃｰｼｮﾝ第 3 ﾊﾟﾙｽ数	ｽﾃｰｼｮﾝ第 4 ﾊﾟﾙｽ数

変数ﾀｲﾌﾟ番号	ﾃﾞｰﾀ型(ﾊﾟﾙｽ/直交)	有効配列数	内 容				
			p[5]	p[6]	p[7]	p[8]	p[9]
0	―	1	―	―	―	―	―
1	―	1	―	―	―	―	―
2	―	1	―	―	―	―	―
3	―	1	―	―	―	―	―
4	ﾊﾟﾙｽ型	8	B 軸ﾊﾟﾙｽ数	T 軸ﾊﾟﾙｽ数	ﾂｰﾙ番号	―	―
4	直交型	10	手首姿勢 Rx 座標値(deg)	手首姿勢 Ry 座標値(deg)	手首姿勢 Rz 座標値(deg)	形態	ﾂｰﾙ番号
5	ﾊﾟﾙｽ型	8	ﾍﾞｰｽ第 5 軸ﾊﾟﾙｽ数	ﾍﾞｰｽ第 6 軸ﾊﾟﾙｽ数	ﾂｰﾙ番号	―	―
5	直交型	10	手首姿勢 Rx 座標値(deg)	手首姿勢 Ry 座標値(deg)	手首姿勢 Rz 座標値(deg)	形態	ﾂｰﾙ番号
6	ﾊﾟﾙｽ型	8	ｽﾃｰｼｮﾝ第 5 ﾊﾟﾙｽ数	ｽﾃｰｼｮﾝ第 6 ﾊﾟﾙｽ数	ﾂｰﾙ番号	―	―

ロボット軸位置、ベース軸位置型変数の場合は、第 1 番目のリターン値によって変数の型がパルス型か直交型かに分かれます。(ステーション軸位置型変数の場合はパルス型のみです。)

座標種別、形態については以下を参照してください。

座標種別

YRC1000、YRC1000micro、DX200、DX100

座標種別のデータは、以下の座標名称に相当します。

座標種別	座標名称	座標種別	座標名称
0	ベース座標	:	:
1	ロボット座標	:	:
2	ユーザ座標 1	63	ユーザ座標 62
3	ユーザ座標 2	64	ユーザ座標 63
:	:	65	ツール座標
:	:	66	マスターツール座標

解 説

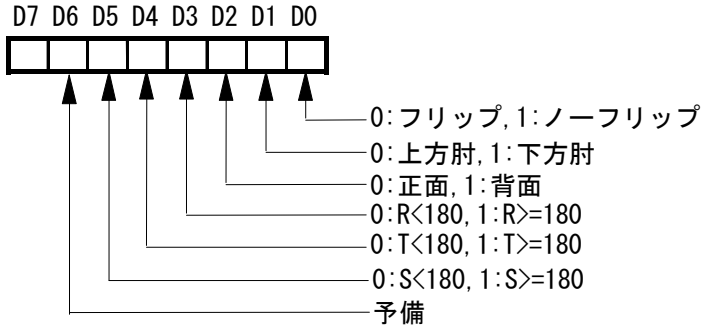
座標種別
FS100
座標種別のデータは、以下の座標名称に相当します。

座標種別	座標名称	座標種別	座標名称
0	ベース座標	10	ユーザ座標 9
1	ロボット座標	11	ユーザ座標 10
2	ユーザ座標 1	12	ユーザ座標 11
3	ユーザ座標 2	13	ユーザ座標 12
4	ユーザ座標 3	14	ユーザ座標 13
5	ユーザ座標 4	15	ユーザ座標 14
6	ユーザ座標 5	16	ユーザ座標 15
7	ユーザ座標 6	17	ユーザ座標 16
8	ユーザ座標 7	18	ツール座標
9	ユーザ座標 8	19	マスターツール座標

座標種別
NX100、XRC、MRC
座標種別のデータは、以下の座標名称に相当します。

座標種別	座標名称	座標種別	座標名称
0	ベース座標	14	ユーザ座標 13
1	ロボット座標	15	ユーザ座標 14
2	ユーザ座標 1	16	ユーザ座標 15
3	ユーザ座標 2	17	ユーザ座標 16
4	ユーザ座標 3	18	ユーザ座標 17
5	ユーザ座標 4	19	ユーザ座標 18
6	ユーザ座標 5	20	ユーザ座標 19
7	ユーザ座標 6	21	ユーザ座標 20
8	ユーザ座標 7	22	ユーザ座標 21
9	ユーザ座標 8	23	ユーザ座標 22
10	ユーザ座標 9	24	ユーザ座標 23
11	ユーザ座標 10	25	ユーザ座標 24
12	ユーザ座標 11	26	ツール座標
13	ユーザ座標 12	27	マスターツール座標

形態
形態のデータは、以下のビットデータを 10 進数にした値となります。



*MRC、MRC2 の場合は D5 － D7 の値は無視されます。

コントローラ

YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC（シリアルポート、Ethernet）

参 照	「BscPutVarData」「BscPutVarData2」

■ BscGetVarData2

機 能	変数情報を受信します。（7 軸以上対応）
書 式	_declspec(dllexport) short APIENTRY BscGetVarData2(short nCid,short type,short varno,double *p);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号 type　変数タイプ varno　変数番号 *p　数値変数格納エリアの先頭ポインタ
	OUT（戻り） *p　数値変数格納エリアの先頭ポインタ
	リターン値 0　：正常終了 その他：エラーコード
解 説	**制約事項** 本関数は YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC に対して伝送を行う場合に有効です。
	変数タイプ 変数タイプは、以下のようになります。 0：バイト型変数 1：整数型変数 2：倍精度型変数 3：実数型変数 4：ロボット軸位置型変数 5：ベース軸位置型変数 6：ステーション軸位置型変数（パルス型のみ）

解 説

数値変数格納エリアの内容

数値変数格納エリアの内容、有効配列数は変数タイプによって以下のようになります。

変数ﾀｲﾌﾟ番号	ﾃﾞｰﾀ型(ﾊﾟﾙｽ/直交)	有効配列数	内 容				
			p[0]	p[1]	p[2]	p[3]	p[4]
0	―	1	ﾊﾞｲﾄ値	―	―	―	―
1	―	1	整数値	―	―	―	―
2	―	1	倍精度整数値	―	―	―	―
3	―	1	実数値	―	―	―	―
4	ﾊﾟﾙｽ型	10	0	S 軸ﾊﾟﾙｽ数	L 軸ﾊﾟﾙｽ数	U 軸ﾊﾟﾙｽ数	R 軸ﾊﾟﾙｽ数
4	直交型	10	1	座標種別	X 軸座標値(mm)	Y 軸座標値(mm)	Z 軸座標値(mm)
5	ﾊﾟﾙｽ型	8	0	ﾍﾞｰｽ第 1 軸ﾊﾟﾙｽ数	ﾍﾞｰｽ第 2 軸ﾊﾟﾙｽ数	ﾍﾞｰｽ第 3 軸ﾊﾟﾙｽ数	ﾍﾞｰｽ第 4 軸ﾊﾟﾙｽ数
5	直交型	10	1	座標種別	X 軸座標値(mm)	Y 軸座標値(mm)	Z 軸座標値(mm)
6	ﾊﾟﾙｽ型	8	0	ｽﾃｰｼｮﾝ第 1 ﾊﾟﾙｽ数	ｽﾃｰｼｮﾝ第 2 ﾊﾟﾙｽ数	ｽﾃｰｼｮﾝ第 3 ﾊﾟﾙｽ数	ｽﾃｰｼｮﾝ第 4 ﾊﾟﾙｽ数

変数ﾀｲﾌﾟ番号	ﾃﾞｰﾀ型(ﾊﾟﾙｽ/直交)	有効配列数	内 容				
			p[5]	p[6]	p[7]	p[8]	p[9]
0	―	1	―	―	―	―	―
1	―	1	―	―	―	―	―
2	―	1	―	―	―	―	―
3	―	1	―	―	―	―	―
4	ﾊﾟﾙｽ型	10	B 軸ﾊﾟﾙｽ数	T 軸ﾊﾟﾙｽ数	7 軸ﾊﾟﾙｽ数	8 軸ﾊﾟﾙｽ数	ﾂｰﾙ番号
4	直交型	10	手首姿勢 Rx 座標値(deg)	手首姿勢 Ry 座標値(deg)	手首姿勢 Rz 座標値(deg)	形態	ﾂｰﾙ番号
5	ﾊﾟﾙｽ型	8	ﾍﾞｰｽ第 5 軸ﾊﾟﾙｽ数	ﾍﾞｰｽ第 6 軸ﾊﾟﾙｽ数	ﾂｰﾙ番号	―	―
5	直交型	10	手首姿勢 Rx 座標値(deg)	手首姿勢 Ry 座標値(deg)	手首姿勢 Rz 座標値(deg)	形態	ﾂｰﾙ番号
6	ﾊﾟﾙｽ型	8	ｽﾃｰｼｮﾝ第 5 ﾊﾟﾙｽ数	ｽﾃｰｼｮﾝ第 6 ﾊﾟﾙｽ数	ﾂｰﾙ番号	―	―

ロボット軸位置、ベース軸位置型変数の場合は、第 1 番目のリターン値によって変数の型がパルス型か直交型かに分かれます。(ステーション軸位置型変数の場合はパルス型のみです。)

座標種別、形態については以下を参照してください。

座標種別

YRC1000、YRC1000micro、DX200、DX100

座標種別のデータは、以下の座標名称に相当します。

座標種別	座標名称	座標種別	座標名称
0	ベース座標	:	:
1	ロボット座標	:	:
2	ユーザ座標 1	63	ユーザ座標 62
3	ユーザ座標 2	64	ユーザ座標 63
:	:	65	ツール座標
:	:	66	マスターツール座標

解 説

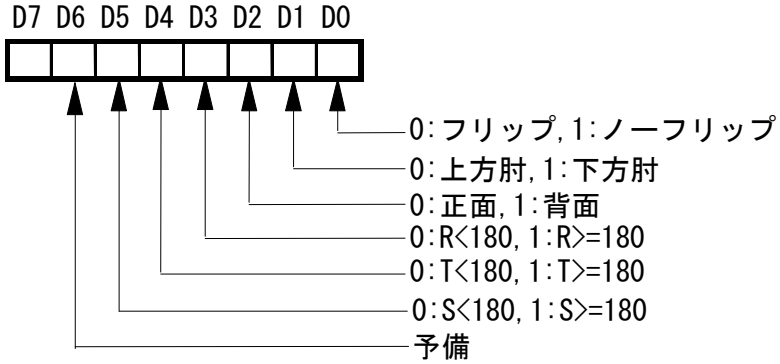
座標種別
FS100
座標種別のデータは、以下の座標名称に相当します。

座標種別	座標名称	座標種別	座標名称
0	ベース座標	10	ユーザ座標 9
1	ロボット座標	11	ユーザ座標 10
2	ユーザ座標 1	12	ユーザ座標 11
3	ユーザ座標 2	13	ユーザ座標 12
4	ユーザ座標 3	14	ユーザ座標 13
5	ユーザ座標 4	15	ユーザ座標 14
6	ユーザ座標 5	16	ユーザ座標 15
7	ユーザ座標 6	17	ユーザ座標 16
8	ユーザ座標 7	18	ツール座標
9	ユーザ座標 8	19	マスターツール座標

座標種別
NX100、XRC
座標種別のデータは、以下の座標名称に相当します。

座標種別	座標名称	座標種別	座標名称
0	ベース座標	14	ユーザ座標 13
1	ロボット座標	15	ユーザ座標 14
2	ユーザ座標 1	16	ユーザ座標 15
3	ユーザ座標 2	17	ユーザ座標 16
4	ユーザ座標 3	18	ユーザ座標 17
5	ユーザ座標 4	19	ユーザ座標 18
6	ユーザ座標 5	20	ユーザ座標 19
7	ユーザ座標 6	21	ユーザ座標 20
8	ユーザ座標 7	22	ユーザ座標 21
9	ユーザ座標 8	23	ユーザ座標 22
10	ユーザ座標 9	24	ユーザ座標 23
11	ユーザ座標 10	25	ユーザ座標 24
12	ユーザ座標 11	26	ツール座標
13	ユーザ座標 12	27	マスターツール座標

形態
形態のデータは、以下のビットデータを 10 進数にした値となります。



コントローラ

YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC（シリアルポート、Ethernet）

参 照	「BscPutVarData」「BscPutVarData2」

■ BscHostGetVarData

機 能	変数情報を受信します。
書 式	_declspec(dllexport) short APIENTRY BscHostGetVarData(short nCid,short type,short varno,double *p,char *str);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号 type　変数タイプ varno　変数番号 *p　数値変数格納エリアの先頭ポインタ *str　文字変数格納エリアの先頭ポインタ
	OUT（戻り） *p　数値変数格納エリアの先頭ポインタ *str　文字変数格納エリアの先頭ポインタ
	リターン値 0　：正常終了 その他：エラーコード
解 説	**制約事項** 本関数は YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC に対して伝送を行う場合に有効です。 また、コントローラが YRC1000、YRC1000micro 及び DX200、DX100、FS100、NX100 Ver3.0 以降の場合に限り文字型変数を取り扱うことが可能です。
	変数タイプ 変数タイプは、以下のようになります。 0：バイト型変数 1：整数型変数 2：倍精度型変数 3：実数型変数 4：ロボット軸位置型変数 5：ベース軸位置型変数 6：ステーション軸位置型変数（パルス型のみ） 7：文字型変数

解 説

変数格納エリアの内容

数値変数格納エリアの内容、有効配列数は変数タイプによって以下のようになります。

変数ﾀｲﾌﾟ番号	ﾃﾞｰﾀ型(ﾊﾟﾙｽ/直交)	有効配列数	内 容				
			p[0]	p[1]	p[2]	p[3]	p[4]
0	―	1	ﾊﾞｲﾄ値	―	―	―	―
1	―	1	整数値	―	―	―	―
2	―	1	倍精度整数値	―	―	―	―
3	―	1	実数値	―	―	―	―
4	ﾊﾟﾙｽ型	8	0	S 軸ﾊﾟﾙｽ数	L 軸ﾊﾟﾙｽ数	U 軸ﾊﾟﾙｽ数	R 軸ﾊﾟﾙｽ数
4	直交型	10	1	座標種別	X 軸座標値(mm)	Y 軸座標値(mm)	Z 軸座標値(mm)
5	ﾊﾟﾙｽ型	8	0	ﾍﾞｰｽ第 1 軸ﾊﾟﾙｽ数	ﾍﾞｰｽ第 2 軸ﾊﾟﾙｽ数	ﾍﾞｰｽ第 3 軸ﾊﾟﾙｽ数	ﾍﾞｰｽ第 4 軸ﾊﾟﾙｽ数
5	直交型	10	1	座標種別	X 軸座標値(mm)	Y 軸座標値(mm)	Z 軸座標値(mm)
6	ﾊﾟﾙｽ型	8	0	ｽﾃｰｼｮﾝ第 1 ﾊﾟﾙｽ数	ｽﾃｰｼｮﾝ第 2 ﾊﾟﾙｽ数	ｽﾃｰｼｮﾝ第 3 ﾊﾟﾙｽ数	ｽﾃｰｼｮﾝ第 4 ﾊﾟﾙｽ数

変数ﾀｲﾌﾟ番号	ﾃﾞｰﾀ型(ﾊﾟﾙｽ/直交)	有効配列数	内 容				
			p[5]	p[6]	p[7]	p[8]	p[9]
0	―	1	―	―	―	―	―
1	―	1	―	―	―	―	―
2	―	1	―	―	―	―	―
3	―	1	―	―	―	―	―
4	ﾊﾟﾙｽ型	8	B 軸ﾊﾟﾙｽ数	T 軸ﾊﾟﾙｽ数	ﾂｰﾙ番号	―	―
4	直交型	10	手首姿勢 Rx 座標値(deg)	手首姿勢 Ry 座標値(deg)	手首姿勢 Rz 座標値(deg)	形態	ﾂｰﾙ番号
5	ﾊﾟﾙｽ型	8	ﾍﾞｰｽ第 5 軸ﾊﾟﾙｽ数	ﾍﾞｰｽ第 6 軸ﾊﾟﾙｽ数	ﾂｰﾙ番号	―	―
5	直交型	10	手首姿勢 Rx 座標値(deg)	手首姿勢 Ry 座標値(deg)	手首姿勢 Rz 座標値(deg)	形態	ﾂｰﾙ番号
6	ﾊﾟﾙｽ型	8	ｽﾃｰｼｮﾝ第 5 ﾊﾟﾙｽ数	ｽﾃｰｼｮﾝ第 6 ﾊﾟﾙｽ数	ﾂｰﾙ番号	―	―

ロボット軸位置、ベース軸位置型変数の場合は、第 1 番目のリターン値によって変数の型がパルス型か直交型かに分かれます。(ステーション軸位置型変数の場合はパルス型のみです。)

変数ﾀｲﾌﾟ番号	ﾃﾞｰﾀ型(ﾊﾟﾙｽ/直交)	有効配列数	内 容
			str
7	―	16	文字列

重要

文字型変数の送受信で本関数をコールする場合、変数格納エリアは以下の例のように 17 文字分の変数エリアを確保してください。

Visual Basic で使用する場合の変数宣言例：

Dim S_Variable As String *17

Visual C++ で使用する場合の変数宣言例：char S_Variable[17];

座標種別、形態については以下を参照してください。

解 説

座標種別
YRC1000、YRC1000micro、DX200、DX100
座標種別のデータは、以下の座標名称に相当します。

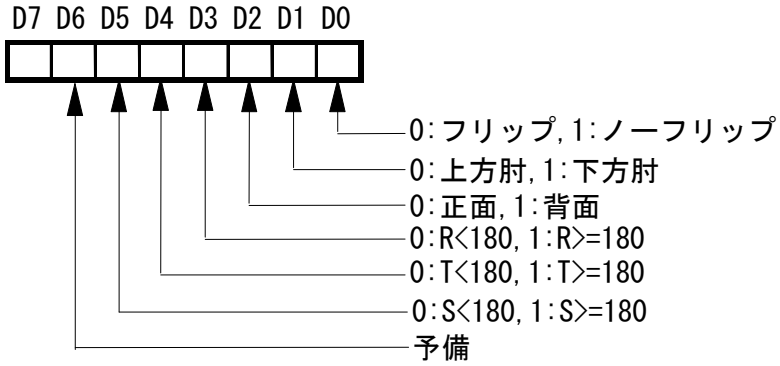
座標種別	座標名称	座標種別	座標名称
0	ベース座標	:	:
1	ロボット座標	:	:
2	ユーザ座標 1	63	ユーザ座標 62
3	ユーザ座標 2	64	ユーザ座標 63
:	:	65	ツール座標
:	:	66	マスターツール座標

座標種別
FS100
座標種別のデータは、以下の座標名称に相当します。

座標種別	座標名称	座標種別	座標名称
0	ベース座標	10	ユーザ座標 9
1	ロボット座標	11	ユーザ座標 10
2	ユーザ座標 1	12	ユーザ座標 11
3	ユーザ座標 2	13	ユーザ座標 12
4	ユーザ座標 3	14	ユーザ座標 13
5	ユーザ座標 4	15	ユーザ座標 14
6	ユーザ座標 5	16	ユーザ座標 15
7	ユーザ座標 6	17	ユーザ座標 16
8	ユーザ座標 7	18	ツール座標
9	ユーザ座標 8	19	マスターツール座標

座標種別
NX100、XRC、MRC
座標種別のデータは、以下の座標名称に相当します。

座標種別	座標名称	座標種別	座標名称
0	ベース座標	14	ユーザ座標 13
1	ロボット座標	15	ユーザ座標 14
2	ユーザ座標 1	16	ユーザ座標 15
3	ユーザ座標 2	17	ユーザ座標 16
4	ユーザ座標 3	18	ユーザ座標 17
5	ユーザ座標 4	19	ユーザ座標 18
6	ユーザ座標 5	20	ユーザ座標 19
7	ユーザ座標 6	21	ユーザ座標 20
8	ユーザ座標 7	22	ユーザ座標 21
9	ユーザ座標 8	23	ユーザ座標 22
10	ユーザ座標 9	24	ユーザ座標 23
11	ユーザ座標 10	25	ユーザ座標 24
12	ユーザ座標 11	26	ツール座標
13	ユーザ座標 12	27	マスターツール座標

解 説	**形態** 形態のデータは、以下のビットデータを１０進数にした値となります。  *MRC, MRC2 の場合は D5-D7 の値は無視されます。
コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC（シリアルポート、Ethernet）
参 照	「BscHostPutVarData」

■ BscHostGetVarDataM

機 能	複数の変数情報を同時に受信します。
書 式	_declspec(dllexport) short APIENTRY BscHostGetVarDataM(short nCid,short type,short varno,short num,double *p);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号 type　変数タイプ varno　変数番号 num　変数個数 *p　数値変数格納エリアの先頭ポインタ
	OUT（戻り） *p　数値変数格納エリアの先頭ポインタ
	リターン値 0　：正常終了 その他：エラーコード
解 説	**制約事項** 本関数は YRC1000、YRC1000micro、DX200、DX100、FS100、NX100 に対して伝送を行う場合に有効です。
	変数タイプ 変数タイプは、以下のようになります。 0：バイト型変数 1：整数型変数 2：倍精度型変数 3：実数型変数
	変数指定方法 変数情報は、指定した変数タイプの指定した変数番号を先頭に指定した変数個数分が受信されます。 変数個数は最大 15 個までです。
コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100（シリアルポート、Ethernet）
参 照	「BscHostPutVarDataM」

■ BscGetVarDataEx

機 能	変数情報を受信します。(7 軸以上対応)
書 式	_declspec(dllexport) short APIENTRY BscGetVarDataEx(short nCid,short type,short varno,double *p,char *str,short *axisNum);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号 type　変数タイプ varno　変数番号 *p　数値変数格納エリアの先頭ポインタ *str　文字変数格納エリアの先頭ポインタ *axisNum　軸数格納ポインタ
	OUT（戻り） *p　数値変数格納エリアの先頭ポインタ *str　文字変数格納エリアの先頭ポインタ *axisNum　軸数格納ポインタ
	リターン値 0　：正常終了 その他：エラーコード
解 説	**制約事項** 本関数は YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC に対して伝送を行う場合に有効です。 また、コントローラが YRC1000 及び YRC1000micro、DX200、DX100、FS100、NX100 Ver3.0 以降の場合に限り文字型変数を取り扱うことが可能です。
	変数タイプ 変数タイプは、以下のようになります。 0：バイト型変数 1：整数型変数 2：倍精度型変数 3：実数型変数 4：ロボット軸位置型変数 5：ベース軸位置型変数 6：ステーション軸位置型変数（パルス型のみ） 7：文字型変数

解 説

変数格納エリアの内容

数値変数格納エリアの内容、有効配列数は変数タイプによって以下のようになります。

変数ﾀｲﾌﾟ番号	ﾃﾞｰﾀ型（ﾊﾟﾙｽ/直交）	有効配列数	内 容				
			p[0]	p[1]	p[2]	p[3]	p[4]
0	―	1	ﾊﾞｲﾄ値	―	―	―	―
1	―	1	整数値	―	―	―	―
2	―	1	倍精度整数値	―	―	―	―
3	―	1	実数値	―	―	―	―
4	ﾊﾟﾙｽ型	9	0	S 軸ﾊﾟﾙｽ数	L 軸ﾊﾟﾙｽ数	U 軸ﾊﾟﾙｽ数	R 軸ﾊﾟﾙｽ数
4	直交型	11	1	座標種別	X 軸座標値 (mm)	Y 軸座標値 (mm)	Z 軸座標値 (mm)
5	ﾊﾟﾙｽ型	8	0	ﾍﾞｰｽ第 1 軸ﾊﾟﾙｽ数	ﾍﾞｰｽ第 2 軸ﾊﾟﾙｽ数	ﾍﾞｰｽ第 3 軸ﾊﾟﾙｽ数	ﾍﾞｰｽ第 4 軸ﾊﾟﾙｽ数
5	直交型	10	1	座標種別	X 軸座標値 (mm)	Y 軸座標値 (mm)	Z 軸座標値 (mm)
6	ﾊﾟﾙｽ型	8	0	ｽﾃｰｼｮﾝ第 1 ﾊﾟﾙｽ数	ｽﾃｰｼｮﾝ第 2 ﾊﾟﾙｽ数	ｽﾃｰｼｮﾝ第 3 ﾊﾟﾙｽ数	ｽﾃｰｼｮﾝ第 4 ﾊﾟﾙｽ数

変数ﾀｲﾌﾟ番号	ﾃﾞｰﾀ型（ﾊﾟﾙｽ/直交）	有効配列数	内 容					
			p[5]	p[6]	p[7]	p[8]	p[9]	p[10]
0	―	1	―	―	―	―	―	―
1	―	1	―	―	―	―	―	―
2	―	1	―	―	―	―	―	―
3	―	1	―	―	―	―	―	―
4	ﾊﾟﾙｽ型	9	B 軸ﾊﾟﾙｽ数	T 軸ﾊﾟﾙｽ数	E 軸ﾊﾟﾙｽ数	ﾂｰﾙ番号	―	―
4	直交型	11	手首姿勢 Rx 座標値 (deg)	手首姿勢 Ry 座標値 (deg)	手首姿勢 Rz 座標値 (deg)	Re (deg)	形態	ﾂｰﾙ番号
5	ﾊﾟﾙｽ型	8	ﾍﾞｰｽ第 5 軸 ﾊﾟﾙｽ数	ﾍﾞｰｽ第 6 軸 ﾊﾟﾙｽ数	ﾂｰﾙ番号	―	―	―
5	直交型	10	手首姿勢 Rx 座標値 (deg)	手首姿勢 Ry 座標値 (deg)	手首姿勢 Rz 座標値 (deg)	形態	ﾂｰﾙ番号	―
6	ﾊﾟﾙｽ型	8	ｽﾃｰｼｮﾝ第 5 ﾊﾟﾙｽ数	ｽﾃｰｼｮﾝ第 6 ﾊﾟﾙｽ数	ﾂｰﾙ番号	―	―	―

ロボット軸位置、ベース軸位置型変数の場合は、第１番目のリターン値によって変数の型がパルス型か直交型かに分かれます。(ステーション軸位置型変数の場合はパルス型のみです。)

変数ﾀｲﾌﾟ番号	ﾃﾞｰﾀ型（ﾊﾟﾙｽ/直交）	有効配列数	内 容
			str
7	―	16	文字列

重要

文字型変数の送受信で本関数をコールする場合、変数格納エリアは以下の例のように 17 文字分の変数エリアを確保してください。

Visual Basic で使用する場合の変数宣言例：
Dim S_Variable As String *17

Visual C++ で使用する場合の変数宣言例：char S_Variable[17];

座標種別、形態については以下を参照してください。

解 説

座標種別
YRC1000、YRC1000micro、DX200、DX100
座標種別のデータは、以下の座標名称に相当します。

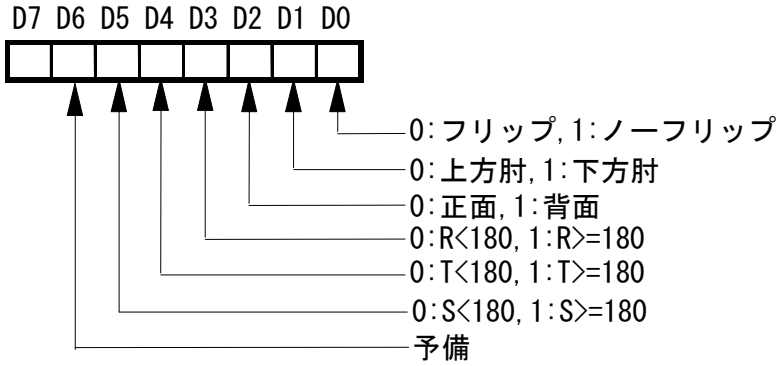
座標種別	座標名称	座標種別	座標名称
0	ベース座標	:	:
1	ロボット座標	:	:
2	ユーザ座標 1	63	ユーザ座標 62
3	ユーザ座標 2	64	ユーザ座標 63
:	:	65	ツール座標
:	:	66	マスターツール座標

座標種別
FS100
座標種別のデータは、以下の座標名称に相当します。

座標種別	座標名称	座標種別	座標名称
0	ベース座標	10	ユーザ座標 9
1	ロボット座標	11	ユーザ座標 10
2	ユーザ座標 1	12	ユーザ座標 11
3	ユーザ座標 2	13	ユーザ座標 12
4	ユーザ座標 3	14	ユーザ座標 13
5	ユーザ座標 4	15	ユーザ座標 14
6	ユーザ座標 5	16	ユーザ座標 15
7	ユーザ座標 6	17	ユーザ座標 16
8	ユーザ座標 7	18	ツール座標
9	ユーザ座標 8	19	マスターツール座標

座標種別
NX100、XRC、MRC
座標種別のデータは、以下の座標名称に相当します。

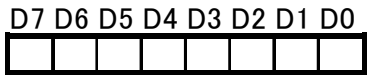
座標種別	座標名称	座標種別	座標名称
0	ベース座標	14	ユーザ座標 13
1	ロボット座標	15	ユーザ座標 14
2	ユーザ座標 1	16	ユーザ座標 15
3	ユーザ座標 2	17	ユーザ座標 16
4	ユーザ座標 3	18	ユーザ座標 17
5	ユーザ座標 4	19	ユーザ座標 18
6	ユーザ座標 5	20	ユーザ座標 19
7	ユーザ座標 6	21	ユーザ座標 20
8	ユーザ座標 7	22	ユーザ座標 21
9	ユーザ座標 8	23	ユーザ座標 22
10	ユーザ座標 9	24	ユーザ座標 23
11	ユーザ座標 10	25	ユーザ座標 24
12	ユーザ座標 11	26	ツール座標
13	ユーザ座標 12	27	マスターツール座標

解 説	**形態** 形態のデータは、以下のビットデータを１０進数にした値となります。  *MRC, MRC2 の場合は D5-D7 の値は無視されます。
コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC（シリアルポート、Ethernet）
参 照	「BscPutVarDataEx」

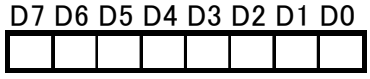
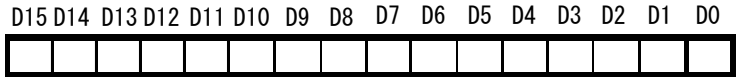
■ BscIsAlarm

機 能	アラーム状態か否かを返します。
書 式	_declspec(dllexport) short APIENTRY BscIsAlarm(short nCid);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号
	OUT（戻り） 無し
	リターン値 -1：状態取得失敗 0：アラーム無し 1：アラーム有り
コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC（シリアルポート、Ethernet） ERC（シリアルポート）
参 照	「BscGetError2」「BscGetFirstAlarm」「BscGetNextAlarm」「BscGetStatus」「BscIsCycle」「BscIsError」「BscIsHold」「BscIsPlayMode」「BscIsRemoteMode」 「BscIsServo」「BscIsTeachMode」

■ BscIsCtrlGroup

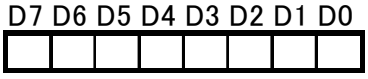
機 能	制御グループ情報を読み込みます。
書 式	_declspec(dllexport) short APIENTRY BscIsCtrlGroup(short nCid);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号
	OUT（戻り） 無し
	リターン値 -1　　：情報取得失敗 その他：制御グループ情報
解 説	**制約事項** 本関数は MRC に対して伝送を行う場合に有効です。 YRC1000、YRC1000micro、DX200、DX100 に対して伝送を行う場合は、BscIsCtrlGroupDX を参照してください。 FS100、NX100、XRC に対して伝送を行う場合は、BscIsCtrlGroupXrc を参照してください。
	制御グループ情報 制御グループ情報は、以下のビットデータを 10 進数にした値となります。  D0：R1（ロボット1） D1：R2（ロボット2） D2：S1（ステーション1） D3：S2（ステーション2） D4：S3（ステーション3） D5：S4（ステーション4） D6：S5（ステーション5） D7：S6（ステーション6）
コントローラ	MRC（シリアルポート、Ethernet）
参 照	「BscGetCtrlGroupDX」「BscSetCtrlGroupDX」「BscIsCtrlGroupDX」「BscIsCtrlGroupXrc」「BscGetCtrlGroupXrc」「BscSetCtrlGroupXrc」「BscIsTaskInfXrc」「BscGetCtrlGroup」「BscSetCtrlGroup」「BscIsTaskInf」「BscChangeTask」

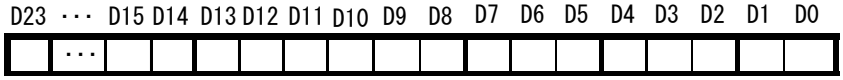
■ BscIsCtrlGroupXrc

機 能	制御グループ情報を読み込みます。
書 式	_declspec(dllexport) short APIENTRY BscIsCtrlGroupXrc(short nCid,short *robtask,short *stattask);
引 数	**IN（引き渡し）** nCid　　通信ハンドラの ID 番号 *robtask　制御グループ情報格納ポインタ（ロボット軸） *stattask　制御グループ情報格納ポインタ（ステーション軸）
	OUT（戻り） *robtask　制御グループ情報格納ポインタ（ロボット軸） *stattask　制御グループ情報格納ポインタ（ステーション軸）
	リターン値 -1 ：情報取得失敗 0 ：正常終了
解 説	**制約事項** 本関数は FS100、NX100、XRC に対して伝送を行う場合に有効です。 YRC1000、YRC1000micro、DX200、DX100 に対して伝送を行う場合は、BscIsCtrlGroupDX を参照してください。 MRC に対して伝送を行う場合は、BscIsCtrlGroup を参照してください。
	制御グループ情報（ロボット軸） 制御グループ情報は、以下のビットデータを 10 進数にした値となります。  D0：R1（ロボット1） D1：R2（ロボット2） D2：R3（ロボット3） D3：R4（ロボット4）
解 説	**制御グループ情報（ステーション軸）** 制御グループ情報は、以下のビットデータを 10 進数にした値となります。  D0：S1（ステーション1） D1：S2（ステーション2） D2：S3（ステーション3） D3：S4（ステーション4） D4：S5（ステーション5） D5：S6（ステーション6） D6：S7（ステーション7） D7：S8（ステーション8） D8：S9（ステーション9） D9：S10（ステーション10） D10：S11（ステーション11） D11：S12（ステーション12）

コントローラ	FS100、NX100、XRC（シリアルポート、Ethernet）
参 照	「BscGetCtrlGroupDX」 「BscSetCtrlGroupDX」 「BscIsCtrlGroupDX」 「BscGetCtrlGroupXrc」 「BscSetCtrlGroupXrc」 「BscIsTaskInfXrc」 「BscIsCtrlGroup」 「BscGetCtrlGroup」 「BscSetCtrlGroup」 「BscIsTaskInf」 「BscChangeTask」

■ BscIsCtrlGroupDX

機 能	制御グループ情報を読み込みます。
書 式	_declspec(dllexport) short APIENTRY BscIsCtrlGroupDX(short nCid,long *robtask,long *stattask);
引 数	**IN（引き渡し）** nCid　　通信ハンドラの ID 番号 *robtask　制御グループ情報格納ポインタ（ロボット軸） *stattask　制御グループ情報格納ポインタ（ステーション軸）
	OUT（戻り） *robtask　制御グループ情報格納ポインタ（ロボット軸） *stattask　制御グループ情報格納ポインタ（ステーション軸）
	リターン値 -1 ：情報取得失敗 0 ：正常終了
解 説	**制約事項** 本関数は YRC1000、YRC1000micro、DX200、DX100 に対して伝送を行う場合に有効です。 FS100、NX100、XRC に対して伝送を行う場合は、BscIsCtrlGroupXrc を参照してください。 MRC に対して伝送を行う場合は、BscIsCtrlGroup を参照してください。
	制御グループ情報（ロボット軸） 制御グループ情報は、以下のビットデータを 10 進数にした値となります。  D0：R1（ロボット1） D1：R2（ロボット2） D2：R3（ロボット3） D3：R4（ロボット4） ： ： D7：R8（ロボット8）

解 説	**制御グループ情報（ステーション軸）** 制御グループ情報は、以下のビットデータを 10 進数にした値となります。  D0：S1（ステーション1） D1：S2（ステーション2） D2：S3（ステーション3） D3：S4（ステーション4） D4：S5（ステーション5） D5：S6（ステーション6） D6：S7（ステーション7） D7：S8（ステーション8） D8：S9（ステーション9） D9：S10（ステーション10） D10：S11（ステーション11） D11：S12（ステーション12） ： ： D23：S24（ステーション 24）
コントローラ	YRC1000、YRC1000micro、DX200、DX100（シリアルポート、Ethernet）
参 照	「BscGetCtrlGroupDX」「BscSetCtrlGroupXrc」「BscIsCtrlGroupXrc」 「BscGetCtrlGroupXrc」「BscSetCtrlGroupXrc」「BscIsTaskInfXrc」 「BscIsCtrlGroup」「BscGetCtrlGroup」「BscSetCtrlGroup」「BscIsTaskInf」 「BscChangeTask」

■ BscIsCycle

機 能	サイクルの状態を読み取ります。
書 式	_declspec(dllexport) short APIENTRY BscIsCycle(short nCid);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号
	OUT（戻り） 無し
	リターン値 -1：状態取得失敗 0：ステップモード 1：1 サイクルモード 2：連続自動モード
コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC（シリアルポート、Ethernet） ERC（シリアルポート）
参 照	「BscGetStatus」「BscIsAlarm」「BscIsError」「BscIsHold」「BscIsPlayMode」「BscIsRemoteMode」「BscIsServo」「BscIsTeachMode」

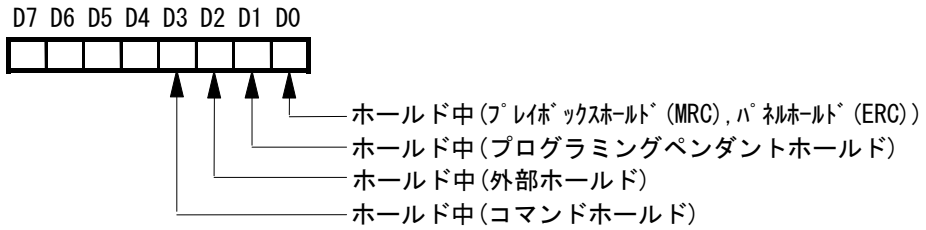
■ BscIsError

機 能	エラーの発生の有無を返します。
書 式	_declspec(dllexport) short APIENTRY BscIsError(short nCid);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号
	OUT（戻り） 無し
	リターン値 -1：状態取得失敗 0：エラー無し 1：エラー有り
コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC（シリアルポート、Ethernet） ERC（シリアルポート）
参 照	「BscGetError2」「BscGetStatus」「BscIsAlarm」「BscIsCycle」「BscIsErrorCode」「BscIsHold」「BscIsPlayMode」「BscIsRemoteMode」「BscIsServo」「BscIsTeachMode」

■ BscIsErrorCode

機 能	エラーコードの取得を行います。
書 式	_declspec(dllexport) short APIENTRY BscIsErrorCode(short nCid);
引 数	IN（引き渡し） nCid　通信ハンドラの ID 番号
	OUT（戻り） 無し
	リターン値 0　　：エラー無し その他：エラーコード
解 説	**呼び出し条件** 本関数を実行する事前に BscIsError 関数をコールして、エラー発生の確認をしておく必要があります。
コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC（シリアルポート、Ethernet） ERC（シリアルポート）
参 照	「BscGetError2」「BscIsError」

■ BscIsHold

機 能	ホールド状態か否かを読み取ります。
書 式	_declspec(dllexport) short APIENTRY BscIsHold(short nCid);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号
	OUT（戻り） 無し
	リターン値 -1　　：状態取得失敗 0　　：非ホールド中 その他：下記参照
解 説	**ホールド状態** ホールド状態のデータは、以下のビットデータを 10 進数にした値となります。 
コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC（シリアルポート、Ethernet） ERC（シリアルポート）
参 照	「BscGetStatus」「BscIsAlarm」「BscIsCycle」「BscIsError」「BscIsPlayMode」「BscIsRemoteMode」「BscIsServo」「BscIsTeachMode」

■ BscIsJobLine

機 能	現在のジョブのライン番号を読み取ります。
書 式	_declspec(dllexport) short APIENTRY BscIsJobLine(short nCid);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号
	OUT（戻り） 無し
	リターン値 -1　　：番号取得失敗 その他：ライン番号
コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC（シリアルポート、Ethernet） ERC（シリアルポート）
参 照	「BscIsJobName」「BscIsJobStep」

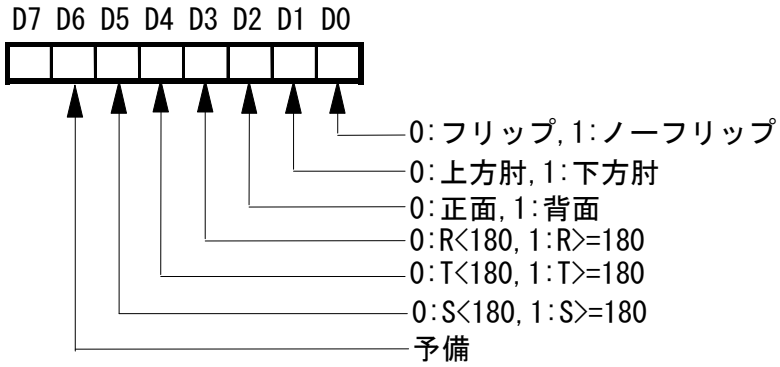
■ BscIsJobName

機 能	現在のジョブの名称を読み取ります。
書 式	_declspec(dllexport) short APIENTRY BscIsJobName(short nCid,char *jobname,short size);
引 数	**IN（引き渡し）** nCid　　通信ハンドラの ID 番号 *jobname　ジョブ名格納ポインタ size　　ジョブ名格納エリアのサイズ
	OUT（戻り） *jobname　ジョブ名格納ポインタ
	リターン値 -1：ジョブ名取得失敗 0：正常終了
コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC（シリアルポート、Ethernet） ERC（シリアルポート）
参 照	「BscIsJobLine」「BscIsJobStep」

■ BscIsJobStep

機 能	現在のジョブのステップ番号を読み取ります。
書 式	_declspec(dllexport) short APIENTRY BscIsJobStep(short nCid);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号
	OUT（戻り） 無し
	リターン値 -1　　：番号取得失敗 その他：ステップ番号
コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC（シリアルポート、Ethernet） ERC（シリアルポート）
参 照	「BscIsJobName」「BscIsJobLine」

■ BscIsLoc

機 能	関節座標系、あるいは直交座標系における現在位置を読み取ります。
書 式	_declspec(dllexport) short APIENTRY BscIsLoc(short nCid,short ispulse,short *rconf,double *p);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号 ispulse　0：直交座標系（MRC、ERC のみ有効）、1：関節座標系 *rconf　形態格納ポインタ *p　現在位置格納エリアの先頭ポインタ
	OUT（戻り） *rconf　形態格納ポインタ *p　現在位置格納エリアの先頭ポインタ
	リターン値 -1：現在位置取得失敗 0：正常終了
解 説	形態 形態のデータは、以下のビットデータを 10 進数にした値となります。  * ERC, ERC2 の場合は D3 － D7 の値は無視されます。 * MRC, MRC2 の場合は D5 － D7 の値は無視されます。

解 説	**現在位置** 関節座標系及び直交座標系を指定した場合の現在位置データは以下のようになります。

	関節座標系の場合	直交座標系の場合
P[0]	S 軸パルス数	X 座標値（単位 mm）
P[1]	L 軸パルス数	Y 座標値（単位 mm）
P[2]	U 軸パルス数	Z 座標値（単位 mm）
P[3]	R 軸パルス数	手首姿勢 Rx（単位°）
P[4]	B 軸パルス数	手首姿勢 Ry（単位°）
P[5]	T 軸パルス数	手首姿勢 Rz（単位°）
P[6]	第 7 軸パルス数	第 7 軸パルス数（走行軸の場合 mm）
P[7]	第 8 軸パルス数	第 8 軸パルス数（走行軸の場合 mm）
P[8]	第 9 軸パルス数	第 9 軸パルス数（走行軸の場合 mm）
P[9]	第 10 軸パルス数	第 10 軸パルス数
P[10]	第 11 軸パルス数	第 11 軸パルス数
P[11]	第 12 軸パルス数	第 12 軸パルス数

コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC（シリアルポート、Ethernet） ERC（シリアルポート）
参 照	「BscIsRobotPos」

■ BscGetPulsePos

機 能	関節座標系における現在位置を読み取ります。
書 式	_declspec(dllexport) short APIENTRY BscGetPulsePos(short nCid,double *p);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号 *p　現在位置格納エリアの先頭ポインタ
	OUT（戻り） *p　現在位置格納エリアの先頭ポインタ
	リターン値 -1：現在位置取得失敗 0：正常終了
解 説	**現在位置** 関節座標系及び直交座標系を指定した場合の現在位置データは以下のようになります。
コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC（シリアルポート、Ethernet）
参 照	「BscGetCartPos」

	関節座標系の場合	7 軸の場合
P[0]	S 軸パルス数	S 軸パルス数
P[1]	L 軸パルス数	L 軸パルス数
P[2]	U 軸パルス数	U 軸パルス数
P[3]	R 軸パルス数	R 軸パルス数
P[4]	B 軸パルス数	B 軸パルス数
P[5]	T 軸パルス数	T 軸パルス数
P[6]	第 7 軸パルス数	E 軸パルス数
P[7]	第 8 軸パルス数	第 8 軸パルス数
P[8]	第 9 軸パルス数	第 9 軸パルス数
P[9]	第 10 軸パルス数	第 10 軸パルス数
P[10]	第 11 軸パルス数	第 11 軸パルス数
P[11]	第 12 軸パルス数	第 12 軸パルス数
P[12]	-	第 13 軸パルス数

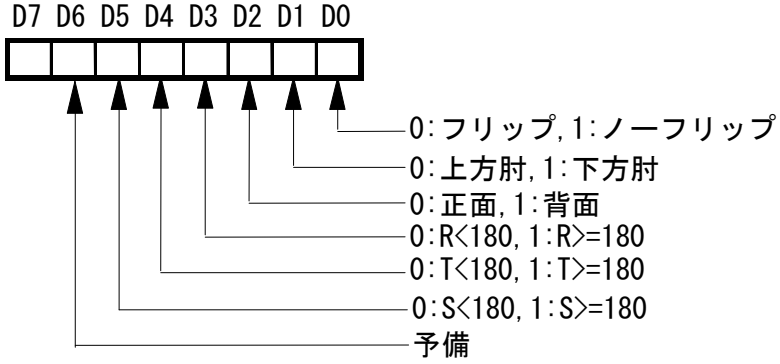
■ BscIsPlayMode

機 能	動作状態を読み取ります。
書 式	_declspec(dllexport) short APIENTRY BscIsPlayMode(short nCid);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号
	OUT（戻り） 無し
	リターン値 -1：状態取得失敗 0：非運転中 1：運転中 2：柵内運転中
コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC（シリアルポート、Ethernet）
参 照	「BscGetStatus」「BscIsAlarm」「BscIsCycle」「BscIsError」 「BscIsRemoteMode」「BscIsServo」「BscIsTeachMode」

■ BscIsRemoteMode

機 能	コマンドリモートの状態か否かを読み取ります。
書 式	_declspec(dllexport) short APIENTRY BscIsRemoteMode(short nCid);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号
	OUT（戻り） 無し
	リターン値 -1：状態取得失敗 0：非コマンドリモートモード 1：コマンドリモートモード
コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC（シリアルポート、Ethernet）
参 照	「BscGetStatus」「BscIsAlarm」「BscIsCycle」「BscIsError」「BscIsHold」「BscIsPlayMode」「BscIsServo」「BscIsTeachMode」

■ BscIsRobotPos

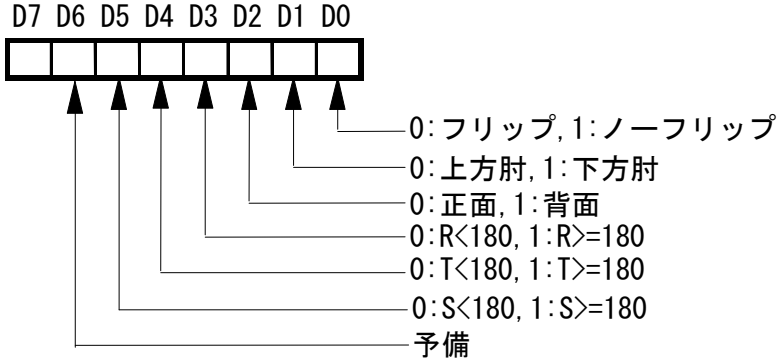
機 能	指定座標系における現在位置を読み取ります。また、外部軸の有・無の指定ができます。
書 式	declspec(dllexport) short APIENTRY BscIsRobotPos(short nCid,char *framename,int isex,short *rconf,short *toolno,double *p);
引 数	**IN（引き渡し）** nCid 通信ハンドラの ID 番号 *framename 座標名称 BASE：ベース座標、ROBOT：ロボット座標、UF1：ユーザー座標 1 ... isex 0：外部軸無し、1：外部軸有り *rconf 形態格納ポインタ *toolno ツール番号格納ポインタ *p 現在位置格納エリアの先頭ポインタ
	OUT（戻り） *rconf 形態格納ポインタ *toolno ツール番号格納ポインタ *p 現在位置格納エリアの先頭ポインタ
	リターン値 -1：現在位置取得失敗 0：正常終了
解 説	**形態** 形態のデータは、以下のビットデータを 10 進数にした値となります。  *ERC、ERC2 の場合は D3 － D7 データは無視されます。 *MRC、MRC2 の場合は D5 － D7 データは無視されます。

解 説	**現在位置** 関節座標系及び直交座標系を指定した場合の現在位置データは以下のようになります。

	指定座標系での現在位置
P[0]	X 座標値（単位 mm）
P[1]	Y 座標値（単位 mm）
P[2]	Z 座標値（単位 mm）
P[3]	手首姿勢 Rx（単位°）
P[4]	手首姿勢 Ry（単位°）
P[5]	手首姿勢 Rz（単位°）
P[6]	第 7 軸パルス数（走行軸の場合 mm）
P[7]	第 8 軸パルス数（走行軸の場合 mm）
P[8]	第 9 軸パルス数（走行軸の場合 mm）
P[9]	第 10 軸パルス数
P[10]	第 11 軸パルス数
P[11]	第 12 軸パルス数

コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC（シリアルポート、Ethernet） ERC（シリアルポート）
参 照	「BscIsLoc」

■ BscGetCartPos

機 能	指定座標系における現在位置を読み取ります。また、外部軸の有・無の指定ができます。
書 式	declspec(dllexport) short APIENTRY BscGetCartPos(short nCid,char *framename,int isex,long *rconf,short *toolno,double *p);
引 数	**IN（引き渡し）** nCid 通信ハンドラの ID 番号 *framename 座標名称 BASE：ベース座標、ROBOT：ロボット座標、UF1：ユーザー座標 1 ... isex 0：外部軸無し、1：外部軸有り *rconf 形態格納ポインタ *toolno ツール番号格納ポインタ *p 現在位置格納エリアの先頭ポインタ
	OUT（戻り） *rconf 形態格納ポインタ *toolno ツール番号格納ポインタ *p 現在位置格納エリアの先頭ポインタ
	リターン値 -1：現在位置取得失敗 0：正常終了
解 説	**形態** 形態のデータは、以下のビットデータを 10 進数にした値となります。  *ERC、ERC2 の場合は D3 － D7 データは無視されます。 *MRC、MRC2 の場合は D5 － D7 データは無視されます。

解 説	**現在位置** 関節座標系及び直交座標系を指定した場合の現在位置データは以下のようになります。

	指定座標系での現在位置	7 軸の場合
P[0]	X 座標値（単位 mm）	X 座標値（単位 mm）
P[1]	Y 座標値（単位 mm）	Y 座標値（単位 mm）
P[2]	Z 座標値（単位 mm）	Z 座標値（単位 mm）
P[3]	手首姿勢 Rx（単位°）	手首姿勢 Rx（単位°）
P[4]	手首姿勢 Ry（単位°）	手首姿勢 Ry（単位°）
P[5]	手首姿勢 Rz（単位°）	手首姿勢 Rz（単位°）
P[6]	第 7 軸パルス数(走行軸の場合 mm)	Re（単位°）
P[7]	第 8 軸パルス数(走行軸の場合 mm)	第 8 軸パルス数（走行軸の場合 mm）
P[8]	第 9 軸パルス数(走行軸の場合 mm)	第 9 軸パルス数（走行軸の場合 mm）
P[9]	第 10 軸パルス数	第 10 軸パルス数（走行軸の場合 mm）
P[10]	第 11 軸パルス数	第 11 軸パルス数
P[11]	第 12 軸パルス数	第 12 軸パルス数
P[12]	ー	第 13 軸パルス数

コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC（シリアルポート、Ethernet） ERC（シリアルポート）
参 照	「BscGetPulsePos」

■ BscIsServo

機 能	サーボオン状態か否かを読み取ります。
書 式	_declspec(dllexport) short APIENTRY BscIsServo(short nCid);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号
	OUT（戻り） 無し
	リターン値 -1：状態取得失敗 0：サーボオフ中 1：サーボオン中
コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC（シリアルポート、Ethernet） ERC（シリアルポート）
参 照	「BscGetStatus」「BscIsAlarm」「BscIsCycle」「BscIsError」「BscIsHold」「BscIsPlayMode」「BscIsRemoteMode」「BscIsTeachMode」「BscServoOn」「BscServoOff」

■ BscIsTaskInf

機 能	タスク情報を読み込みます。
書 式	_declspec(dllexport) short APIENTRY BscIsTaskInf(short nCid);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号
	OUT（戻り） 無し
	リターン値 -1　　：タスク情報取得失敗 その他：タスク情報
解 説	**制約事項** 本関数は MRC に対して伝送を行う場合に有効です。 YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC に対して伝送を行う場合は、BscIsTaskInfXrc を参照してください。
	タスク情報 タスク情報は、以下のようになります。 0：マスタータスク 1：サブ 1 タスク 2：サブ 2 タスク なお、独立制御が許可されていないシステムでは、「0」を返します。
コントローラ	MRC（シリアルポート、Ethernet）
参 照	「BscIsTaskInfXrc」「BscGetCtrlGroupXrc」「BscSetCtrlGroupXrc」「BscIsCtrlGroupXrc」「BscGetCtrlGroup」「BscSetCtrlGroup」「BscIsCtrlGroup」「BscChangeTask」

■ BscIsTaskInfXrc

機 能	タスク情報を読み込みます。
書 式	_declspec(dllexport) short APIENTRY BscIsTaskInfXrc(short nCid);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号
	OUT（戻り） 無し
	リターン値 -1　　：タスク情報取得失敗 その他：タスク情報
解 説	**制約事項** 本関数は YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC に対して伝送を行う場合に有効です。 MRC に対しては伝送を行う場合は、BscIsTaskInf を参照してください。
	タスク情報 タスク情報は、以下のようになります。 0：マスタータスク 1：サブ 1 タスク 2：サブ 2 タスク 3：サブ 3 タスク 4：サブ 4 タスク 5：サブ 5 タスク 6：サブ 6 タスク 7：サブ 7 タスク なお、独立制御が許可されていないシステムでは、「0」を返します。
コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC（シリアルポート、Ethernet）
参 照	「BscGetCtrlGroupXrc」「BscSetCtrlGroupXrc」「BscIsCtrlGroupXrc」「BscIsTaskInf」「BscGetCtrlGroup」「BscSetCtrlGroup」「BscIsCtrlGroup」「BscChangeTask」

■ BscIsTeachMode

機 能	ティーチモード、プレイモードのどちらの状態かを読み取ります。
書 式	_declspec(dllexport) short APIENTRY BscIsTeachMode(short nCid);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号
	OUT（戻り） 無し
	リターン値 -1：状態取得失敗 0：ティーチモード 1：プレイモード
コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC（シリアルポート、Ethernet）
参 照	「BscGetStatus」「BscIsAlarm」「BscIsCycle」「BscIsError」「BscIsHold」「BscIsPlayMode」「BscIsRemoteMode」「BscIsServo」

■ BscJobWait

機 能	マニピュレータの動作が完了するか、指定時間が経過するまでジョブの完了を待ちます。
書 式	_declspec(dllexport) short APIENTRY BscJobWait(short nCid,short time);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号 time　待ち時間（-1：無制限、0 ～ 32767 秒）
	OUT（戻り） 無し
	リターン値 -2　　：異常終了 -1　　：動作未完 0　　：動作完了 その他：エラーコード
解 説	**動作未完の要因** 動作未完の要因は、以下のものが考えられます。 ＊ティーチペンダント、外部ホールドによるロボットの停止 ＊アラーム発生によるロボットの停止 ＊非常停止によるロボットの停止 ＊タイムアップ時
コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC（シリアルポート、Ethernet） ERC（シリアルポート）

■ BscReadAlarmS

機 能	エラーコード、エラーデータ、及びエラーメッセージを取得します。
書 式	_declspec(dllexport) short APIENTRY BscReadAlarmS(short nCid,short *data,char *msg);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号 *data　エラーデータ格納ポインタ *msg　エラーメッセージ格納ポインタ
	OUT（戻り） *data　エラーデータ格納ポインタ *msg　エラーメッセージ格納ポインタ
	リターン値 -1　　：エラーコード取得失敗 0　　：エラー無し その他：エラーコード
解 説	制約事項 本関数は YRC1000、YRC1000micro、DX200、DX100、FS100、NX100 に対して伝送を行う場合に有効です。
コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100（シリアルポート、Ethernet）
参 照	「BscGetFirstAlarmS」「BscGetNextAlarmS」「BscGetError2」

■ BscGetTorque

機 能	各軸モータのトルク値を読み取ります。
書 式	_declspec(dllexport) BscGetTorque(short nCid,double *p);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号 *p　　トルク値格納エリアの先頭ポインタ
	OUT（戻り） *p　　トルク値格納エリアの先頭ポインタ
	リターン値 -1：トルク値取得失敗 0：正常終了
解 説	**制約事項** 本関数はコントローラが YRC1000、YRC1000micro、DX200 Ver1.60-00 以降でのみサポートされています。
	トルク値 関節座標系を指定した場合のトルク値データは以下のようになります。

	関節座標系の場合	7 軸の場合
P[0]	S 軸モータトルク	S 軸モータトルク
P[1]	L 軸モータトルク	L 軸モータトルク
P[2]	U 軸モータトルク	U 軸モータトルク
P[3]	R 軸モータトルク	R 軸モータトルク
P[4]	B 軸モータトルク	B 軸モータトルク
P[5]	T 軸モータトルク	T 軸モータトルク
P[6]	第 7 軸モータトルク	E 軸モータトルク
P[7]	第 8 軸モータトルク	第 8 軸モータトルク
P[8]	第 9 軸モータトルク	第 9 軸モータトルク
P[9]	第 10 軸モータトルク	第 10 軸モータトルク
P[10]	第 11 軸モータトルク	第 11 軸モータトルク
P[11]	第 12 軸モータトルク	第 12 軸モータトルク
P[12]	―	第 13 軸モータトルク

コントローラ	YRC1000、YRC1000micro、DX200（Ver1.60-00 以降）
参 照	「BscGetMaxTorque」

■ BscGetMaxTorque

機 能	各軸モータの最大トルク値を読み取ります。
書 式	_declspec(dllexport) BscGetMaxTorque(short nCid,double *p);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号 *p　最大トルク値格納エリアの先頭ポインタ
	OUT（戻り） *p　最大トルク値格納エリアの先頭ポインタ
	リターン値 -1：最大トルク値取得失敗 0：正常終了
解 説	**制約事項** 本関数はコントローラが YRC1000、YRC1000micro、DX200 Ver1.60-00 以降でのみサポートされています。
	最大トルク値 関節座標系を指定した場合の最大トルク値データは以下のようになります。

	関節座標系の場合	7 軸の場合
P[0]	S 軸モータ最大トルク	S 軸モータ最大トルク
P[1]	L 軸モータ最大トルク	L 軸モータ最大トルク
P[2]	U 軸モータ最大トルク	U 軸モータ最大トルク
P[3]	R 軸モータ最大トルク	R 軸モータ最大トルク
P[4]	B 軸モータ最大トルク	B 軸モータ最大トルク
P[5]	T 軸モータ最大トルク	T 軸モータ最大トルク
P[6]	第 7 軸モータ最大トルク	E 軸モータ最大トルク
P[7]	第 8 軸モータ最大トルク	第 8 軸モータ最大トルク
P[8]	第 9 軸モータ最大トルク	第 9 軸モータ最大トルク
P[9]	第 10 軸モータ最大トルク	第 10 軸モータ最大トルク
P[10]	第 11 軸モータ最大トルク	第 11 軸モータ最大トルク
P[11]	第 12 軸モータ最大トルク	第 12 軸モータ最大トルク
P[12]	―	第 13 軸モータ最大トルク

コントローラ	YRC1000、YRC1000micro、DX200（Ver1.60-00 以降）
参 照	「BscGetTorque」

■ BscGetEncTmp

機 能	各軸モータのエンコーダ温度を読み取ります。
書 式	_declspec(dllexport) BscGetEncTmp(short nCid,double *p);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号 *p　　エンコーダ温度格納エリアの先頭ポインタ
	OUT（戻り） *p　　エンコーダ温度格納エリアの先頭ポインタ
	リターン値 -1：エンコーダ温度取得失敗 0：正常終了
解 説	制約事項 本関数はコントローラが YRC1000、YRC1000micro、DX200 Ver1.60-00 以降でのみサポートされています。
	エンコーダ温度 関節座標系を指定した場合のエンコーダ温度データは以下のようになります。
コントローラ	YRC1000、YRC1000micro、DX200（Ver1.60-00 以降）

	関節座標系の場合	7 軸の場合
P[0]	S 軸モータエンコーダ温度	S 軸モータエンコーダ温度
P[1]	L 軸モータエンコーダ温度	L 軸モータエンコーダ温度
P[2]	U 軸モータエンコーダ温度	U 軸モータエンコーダ温度
P[3]	R 軸モータエンコーダ温度	R 軸モータエンコーダ温度
P[4]	B 軸モータエンコーダ温度	B 軸モータエンコーダ温度
P[5]	T 軸モータエンコーダ温度	T 軸モータエンコーダ温度
P[6]	第 7 軸モータエンコーダ温度	E 軸モータエンコーダ温度
P[7]	第 8 軸モータエンコーダ温度	第 8 軸モータエンコーダ温度
P[8]	第 9 軸モータエンコーダ温度	第 9 軸モータエンコーダ温度
P[9]	第 10 軸モータエンコーダ温度	第 10 軸モータエンコーダ温度
P[10]	第 11 軸モータエンコーダ温度	第 11 軸モータエンコーダ温度
P[11]	第 12 軸モータエンコーダ温度	第 12 軸モータエンコーダ温度
P[12]	－	第 13 軸モータエンコーダ温度

■ BscGetSysTime

機 能	ロボットコントローラのシステム管理時間を読み取ります。
書 式	_declspec(dllexport) BscGetSysTime(short nCid,char *contpwr, char *svpwr, char *playback, char *mov, char *ope);
引 数	**IN（引き渡し）** nCid 通信ハンドラの ID 番号 *contpwr 制御電源投入時間格納エリアの先頭ポインタ * svpwr サーボ電源投入時間格納エリアの先頭ポインタ * playback プレイバック時間格納エリアの先頭ポインタ * mov 移動時間格納エリアの先頭ポインタ * ope 作業時間格納エリアの先頭ポインタ
	OUT（戻り） *contpwr 制御電源投入時間格納エリアの先頭ポインタ * svpwr サーボ電源投入時間格納エリアの先頭ポインタ * playback プレイバック時間格納エリアの先頭ポインタ * mov 移動時間格納エリアの先頭ポインタ * ope 作業時間格納エリアの先頭ポインタ
	リターン値 -1：システム管理時間取得失敗 0：正常終了
解 説	**制約事項** 本関数はコントローラが YRC1000、YRC1000micro、DX200 Ver1.60-00 以降でのみサポートされています。
	システム管理時間 各時間の出力書式は以下のようになります。 HHHHH:MM'SS HHHHH= 時間、MM= 分、SS= 秒
コントローラ	YRC1000、YRC1000micro、DX200（Ver1.60-00 以降）

■ BscCancel

機 能	エラーのキャンセルを行います。
書 式	_declspec(dllexport) short APIENTRY BscCancel(short nCid);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号
	OUT（戻り） 無し
	リターン値 0　　：正常終了 その他：エラーコード
コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC（シリアルポート、Ethernet） ERC（シリアルポート）
参 照	「BscReset」

■ BscChangeTask

機 能	タスクの切り替えを行います。
書 式	_declspec(dllexport) short APIENTRY BscChangeTask(short nCid,short task);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号 task　指定タスク番号
	OUT（戻り） 無し
	リターン値 0　　：正常終了 その他：エラーコード
解 説	**制約事項** 本関数は YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC に対して伝送を行う場合に有効です。 コントローラの電源立ち上げ時には、対象タスクとしてマスタータスクが選択された状態になります。また独立制御が許可されていないシステムでは、この関数は使用することができません。
	指定タスク番号 指定タスク番号の意味は以下のようになります。 0：タスク 1：サブ 1 タスク 2：サブ 2 タスク 3：サブ 3 タスク 4：サブ 4 タスク 5：サブ 5 タスク 6：サブ 6 タスク 7：サブ 7 タスク * 指定タスク番号の 3 ～ 7 番までは YRC1000、YRC1000micro、DX200、DX100、NX100、XRC のみ有効です。 * 指定タスク番号の 3 ～ 5 番までは FS100 まで有効です。
コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC（シリアルポート、Ethernet） ERC（シリアルポート）
参 照	「BscGetCtrlGroupXrc」「BscSetCtrlGroupXrc」「BscIsCtrlGroupXrc」「BscIsTaskInfXrc」「BscGetCtrlGroup」「BscSetCtrlGroup」「BscIsCtrlGroup」「BscIsTaskInf」

■ BscContinueJob

機 能	始動をかけます。(始動をかけるジョブは現在ロボットで選択されているジョブ名で、現ラインから実行を開始します。)
書 式	_declspec(dllexport) short APIENTRY BscContinueJob(short nCid);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号
	OUT（戻り） 無し
	リターン値 0　　：正常終了 その他：エラーコード
コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC（シリアルポート、Ethernet） ERC（シリアルポート）
参 照	「BscStartJob」

■ BscConvertJobP2R

機 能	ジョブを指定の座標の相対ジョブに変換します。
書 式	_declspec(dllexport) short APIENTRY BscConvertJobP2R(short nCid,char *name,char *framename);
引 数	**IN（引き渡し）** nCid　　通信ハンドラの id 番号 *name　　ジョブ名称格納ポインタ *framename　座標名称 BASE：ベース座標、ROBOT：ロボット座標、 UF1：ユーザー座標 1 ...
	OUT（戻り） 無し
	リターン値 0　　：正常終了 その他：エラーコード
コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC（シリアルポート、Ethernet） ERC（シリアルポート）
参 照	「BscConvertJobR2P」

■ BscConvertJobR2P

機 能	指定の座標の相対ジョブを指定の座標のパルスジョブに変換します。
書 式	_declspec(dllexport) short APIENTRY BscConvertJobR2P(short nCid,char *name,short cv_type,char *var_no);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号 *name　ジョブ名称格納ポインタ cv_type　変換方法 var_no　参照位置変数番号
	OUT（戻り） 無し
	リターン値 0　：正常終了 その他：エラーコード
解 説	**制約事項** 本関数は YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC に対して伝送を行う場合に有効です。
	変換方法 変換方法で指定する番号の意味は以下のようになります。 0：前ステップ重視（B 軸符号同一） 1：形態重視 2：前ステップ重視（R 軸移動量最小）
コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC（シリアルポート、Ethernet）
参 照	「BscConvertJobP2R」

■ BscDeleteJob

機 能	ジョブを消去します。
書 式	_declspec(dllexport) short APIENTRY BscDeleteJob(short nCid);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号
	OUT（戻り） 無し
	リターン値 0　　：正常終了 1　　：消去不可 その他：エラーコード
解 説	**呼び出し条件** 本関数を実行する事前に BscSelectJob 関数をコールして、消去ジョブ名の指定をしておく必要があります。
コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC（シリアルポート、Ethernet） ERC（シリアルポート）
参 照	「BscSelectJob」「BscSetMasterJob」

■ BscHoldOff

機 能	ホールドオフを実行します。
書 式	_declspec(dllexport) short APIENTRY BscHoldOff(short nCid);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号
	OUT（戻り） 無し
	リターン値 0　　：正常終了 その他：エラーコード
コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC（シリアルポート、Ethernet） ERC（シリアルポート）
参 照	「BscHoldOn」

■ BscHoldOn

機 能	ホールドオンを実行します。
書 式	_declspec(dllexport) short APIENTRY BscHoldOn(short nCid);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号
	OUT（戻り） 無し
	リターン値 0　　：正常終了 その他：エラーコード
コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC（シリアルポート、Ethernet） ERC（シリアルポート）
参 照	「BscHoldOff」

■ BscHostPutVarData

機 能	変数情報を送信します。
書 式	_declspec(dllexport) short APIENTRY BscHostPutVarData(short nCid,short type,short varno,double *p, char *str);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号 type　変数タイプ varno　変数番号 *p　数値変数格納エリアの先頭ポインタ *str　文字変数格納エリアの先頭ポインタ
	OUT（戻り） 無し
	リターン値 0　：正常終了 その他：エラーコード
解 説	**制約事項** 本関数は YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC に対して伝送を行う場合に有効です。 また、コントローラが YRC1000 及び YRC1000micro、DX200、DX100、FS100、NX100 Ver3.0 以降の場合に限り文字型変数を取り扱うことが可能です。
	変数タイプ 変数タイプは、以下のようになります。 0：バイト型変数 1：整数型変数 2：倍精度型変数 3：実数型変数 4：ロボット軸位置型変数 5：ベース軸位置型変数 6：ステーション軸位置型変数（パルス型のみ） 7：文字型変数

解 説

変数格納エリアの内容
数値変数格納エリアの内容、有効配列数は変数タイプによって以下のようになります。

変数ﾀｲﾌﾟ番号	ﾃﾞｰﾀ型(ﾊﾟﾙｽ/直交)	有効配列数	内 容				
			p[0]	p[1]	p[2]	p[3]	p[4]
0	―	1	ﾊﾞｲﾄ値	―	―	―	―
1	―	1	整数値	―	―	―	―
2	―	1	倍精度整数値	―	―	―	―
3	―	1	実数値	―	―	―	―
4	ﾊﾟﾙｽ型	8	0	S 軸ﾊﾟﾙｽ数	L 軸ﾊﾟﾙｽ数	U 軸ﾊﾟﾙｽ数	R 軸ﾊﾟﾙｽ数
4	直交型	10	1	座標種別	X 軸座標値(mm)	Y 軸座標値(mm)	Z 軸座標値(mm)
5	ﾊﾟﾙｽ型	8	0	ﾍﾞｰｽ第 1 軸ﾊﾟﾙｽ数	ﾍﾞｰｽ第 2 軸ﾊﾟﾙｽ数	ﾍﾞｰｽ第 3 軸ﾊﾟﾙｽ数	ﾍﾞｰｽ第 4 軸ﾊﾟﾙｽ数
5	直交型	10	1	座標種別	X 軸座標値(mm)	Y 軸座標値(mm)	Z 軸座標値(mm)
6	ﾊﾟﾙｽ型	8	0	ｽﾃｰｼｮﾝ第 1 ﾊﾟﾙｽ数	ｽﾃｰｼｮﾝ第 2 ﾊﾟﾙｽ数	ｽﾃｰｼｮﾝ第 3 ﾊﾟﾙｽ数	ｽﾃｰｼｮﾝ第 4 ﾊﾟﾙｽ数

変数ﾀｲﾌﾟ番号	ﾃﾞｰﾀ型(ﾊﾟﾙｽ/直交)	有効配列数	内 容				
			p[5]	p[6]	p[7]	p[8]	p[9]
0	―	1	―	―	―	―	―
1	―	1	―	―	―	―	―
2	―	1	―	―	―	―	―
3	―	1	―	―	―	―	―
4	ﾊﾟﾙｽ型	8	B 軸ﾊﾟﾙｽ数	T 軸ﾊﾟﾙｽ数	ﾂｰﾙ番号	―	―
4	直交型	10	手首姿勢 Rx 座標値(deg)	手首姿勢 Ry 座標値(deg)	手首姿勢 Rz 座標値(deg)	形態	ﾂｰﾙ番号
5	ﾊﾟﾙｽ型	8	ﾍﾞｰｽ第 5 軸ﾊﾟﾙｽ数	ﾍﾞｰｽ第 6 軸ﾊﾟﾙｽ数	ﾂｰﾙ番号	―	―
5	直交型	10	手首姿勢 Rx 座標値(deg)	手首姿勢 Ry 座標値(deg)	手首姿勢 Rz 座標値(deg)	形態	ﾂｰﾙ番号
6	ﾊﾟﾙｽ型	8	ｽﾃｰｼｮﾝ第 5 ﾊﾟﾙｽ数	ｽﾃｰｼｮﾝ第 6 ﾊﾟﾙｽ数	ﾂｰﾙ番号	―	―

ロボット軸位置、ベース軸位置型変数の場合は、第 1 番目のリターン値によって変数の型がパルス型か直交型かに分かれます。(ステーション軸位置型変数の場合はパルス型のみです。)

変数ﾀｲﾌﾟ番号	ﾃﾞｰﾀ型(ﾊﾟﾙｽ/直交)	有効配列数	内 容
			str
7	―	16	文字列

重要

文字型変数の送受信で本関数をコールする場合、変数格納エリアは以下の例のように 17 文字分の変数エリアを確保してください。
Visual Basic で使用する場合の変数宣言例：
Dim S_Variable As String *17
Visual C++ で使用する場合の変数宣言例：char S_Variable[17];

座標種別、形態については以下を参照してください。

解 説

座標種別
YRC1000、YRC1000micro、DX200、DX100、
座標種別のデータは、以下の座標名称に相当します。

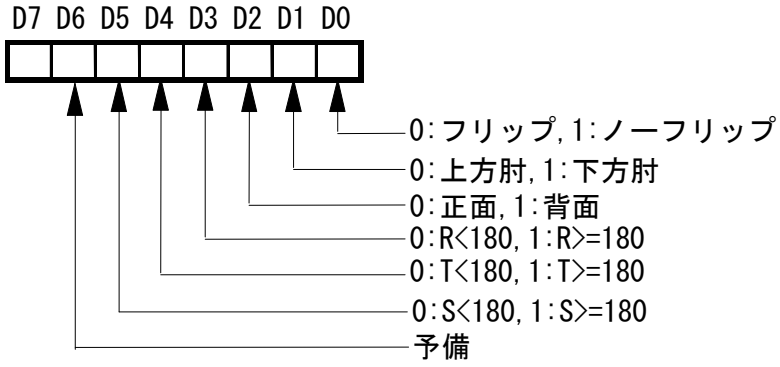
座標種別	座標名称	座標種別	座標名称
0	ベース座標	:	:
1	ロボット座標	:	:
2	ユーザ座標 1	63	ユーザ座標 62
3	ユーザ座標 2	64	ユーザ座標 63
:	:	65	ツール座標
:	:	66	マスターツール座標

座標種別
FS100
座標種別のデータは、以下の座標名称に相当します。

座標種別	座標名称	座標種別	座標名称
0	ベース座標	10	ユーザ座標 9
1	ロボット座標	11	ユーザ座標 10
2	ユーザ座標 1	12	ユーザ座標 11
3	ユーザ座標 2	13	ユーザ座標 12
4	ユーザ座標 3	14	ユーザ座標 13
5	ユーザ座標 4	15	ユーザ座標 14
6	ユーザ座標 5	16	ユーザ座標 15
7	ユーザ座標 6	17	ユーザ座標 16
8	ユーザ座標 7	18	ツール座標
9	ユーザ座標 8	19	マスターツール座標

座標種別
NX100、XRC、MRC
座標種別のデータは、以下の座標名称に相当します。

座標種別	座標名称	座標種別	座標名称
0	ベース座標	14	ユーザ座標 13
1	ロボット座標	15	ユーザ座標 14
2	ユーザ座標 1	16	ユーザ座標 15
3	ユーザ座標 2	17	ユーザ座標 16
4	ユーザ座標 3	18	ユーザ座標 17
5	ユーザ座標 4	19	ユーザ座標 18
6	ユーザ座標 5	20	ユーザ座標 19
7	ユーザ座標 6	21	ユーザ座標 20
8	ユーザ座標 7	22	ユーザ座標 21
9	ユーザ座標 8	23	ユーザ座標 22
10	ユーザ座標 9	24	ユーザ座標 23
11	ユーザ座標 10	25	ユーザ座標 24
12	ユーザ座標 11	26	ツール座標
13	ユーザ座標 12	27	マスターツール座標

解 説	形態 形態のデータは、以下のビットデータを 10 進数にした値となります。  *MRC, MRC2 の場合は D5 － D7 の値は無視されます。
コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC（シリアルポート、Ethernet）
参 照	「BscHostGetVarData」

■ BscHostPutVarDataM

機 能	複数の変数情報を同時に送信します。
書 式	_declspec(dllexport) short APIENTRY BscHostPutVarDataM(short nCid,short type,short varno, short num, double *p);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号 type　変数タイプ varno　変数番号 num　変数個数 *p　数値変数格納エリアの先頭ポインタ
	OUT（戻り） 無し
	リターン値 0　：正常終了 その他：エラーコード
解 説	**制約事項** 本関数は YRC1000、YRC1000micro、DX200、DX100、FS100、NX100 に対して伝送を行う場合に有効です。
	変数タイプ 変数タイプは、以下のようになります。 0：バイト型変数 1：整数型変数 2：倍精度型変数 3：実数型変数
	変数指定方法 変数情報は、指定した変数タイプの指定した変数番号を先頭に指定した変数個数分が送信されます。 変数個数は最大 15 個までです。
コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100（シリアルポート、Ethernet）
参 照	「BscHostGetVarDataM」

■ BscPutVarDataEx

機 能	変数情報を送信します。(7 軸以上対応)
書 式	_declspec(dllexport) short APIENTRY BscPutVarDataEx(short nCid,short type,short varno,double *p, char *str,short axisNum);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号 type　変数タイプ varno　変数番号 *p　数値変数格納エリアの先頭ポインタ *str　文字変数格納エリアの先頭ポインタ axisNum　軸数
	OUT（戻り） 無し
	リターン値 0　：正常終了 その他：エラーコード
解 説	**制約事項** 本関数は YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC に対して伝送を行う場合に有効です。 また、コントローラが YRC1000 及び YRC1000micro、DX200、DX100、FS100、NX100 Ver3.0 以降の場合に限り文字型変数を取り扱うことが可能です。
	変数タイプ 変数タイプは、以下のようになります。 0：バイト型変数 1：整数型変数 2：倍精度型変数 3：実数型変数 4：ロボット軸位置型変数 5：ベース軸位置型変数 6：ステーション軸位置型変数（パルス型のみ） 7：文字型変数

解 説

変数格納エリアの内容
数値変数格納エリアの内容、有効配列数は変数タイプによって以下のようになります。

変数ﾀｲﾌﾟ番号	ﾃﾞｰﾀ型（ﾊﾟﾙｽ/直交）	有効配列数	内 容				
			p[0]	p[1]	p[2]	p[3]	p[4]
0	―	1	ﾊﾞｲﾄ値	―	―	―	―
1	―	1	整数値	―	―	―	―
2	―	1	倍精度整数値	―	―	―	―
3	―	1	実数値	―	―	―	―
4	ﾊﾟﾙｽ型	9	0	S 軸ﾊﾟﾙｽ数	L 軸ﾊﾟﾙｽ数	U 軸ﾊﾟﾙｽ数	R 軸ﾊﾟﾙｽ数
4	直交型	11	1	座標種別	X 軸座標値（mm）	Y 軸座標値（mm）	Z 軸座標値（mm）
5	ﾊﾟﾙｽ型	8	0	ﾍﾞｰｽ第 1 軸ﾊﾟﾙｽ数	ﾍﾞｰｽ第 2 軸ﾊﾟﾙｽ数	ﾍﾞｰｽ第 3 軸ﾊﾟﾙｽ数	ﾍﾞｰｽ第 4 軸ﾊﾟﾙｽ数
5	直交型	10	1	座標種別	X 軸座標値（mm）	Y 軸座標値（mm）	Z 軸座標値（mm）
6	ﾊﾟﾙｽ型	8	0	ｽﾃｰｼｮﾝ第 1 ﾊﾟﾙｽ数	ｽﾃｰｼｮﾝ第 2 ﾊﾟﾙｽ数	ｽﾃｰｼｮﾝ第 3 ﾊﾟﾙｽ数	ｽﾃｰｼｮﾝ第 4 ﾊﾟﾙｽ数

変数ﾀｲﾌﾟ番号	ﾃﾞｰﾀ型（ﾊﾟﾙｽ/直交）	有効配列数	内 容					
			p[5]	p[6]	p[7]	p[8]	p[9]	p[10]
0	―	1	―	―	―	―	―	―
1	―	1	―	―	―	―	―	―
2	―	1	―	―	―	―	―	―
3	―	1	―	―	―	―	―	―
4	ﾊﾟﾙｽ型	9	B 軸ﾊﾟﾙｽ数	T 軸ﾊﾟﾙｽ数	E 軸ﾊﾟﾙｽ数	ﾂｰﾙ番号	―	―
4	直交型	11	手首姿勢 Rx 座標値（deg）	手首姿勢 Ry 座標値（deg）	手首姿勢 Rz 座標値（deg）	Re（deg）	形態	ﾂｰﾙ番号
5	ﾊﾟﾙｽ型	8	ﾍﾞｰｽ第 5 軸ﾊﾟﾙｽ数	ﾍﾞｰｽ第 6 軸ﾊﾟﾙｽ数	ﾂｰﾙ番号	―	―	―
5	直交型	10	手首姿勢 Rx 座標値（deg）	手首姿勢 Ry 座標値（deg）	手首姿勢 Rz 座標値（deg）	形態	ﾂｰﾙ番号	―
6	ﾊﾟﾙｽ型	8	ｽﾃｰｼｮﾝ第 5 ﾊﾟﾙｽ数	ｽﾃｰｼｮﾝ第 6 ﾊﾟﾙｽ数	ﾂｰﾙ番号	―	―	―

ロボット軸位置、ベース軸位置型変数の場合は、第 1 番目のリターン値によって変数の型がパルス型か直交型かに分かれます。(ステーション軸位置型変数の場合はパルス型のみです。)

変数ﾀｲﾌﾟ番号	ﾃﾞｰﾀ型（ﾊﾟﾙｽ/直交）	有効配列数	内 容
			str
7	―	16	文字列

重要

文字型変数の送受信で本関数をコールする場合、変数格納エリアは以下の例のように 17 文字分の変数エリアを確保してください。
Visual Basic で使用する場合の変数宣言例：
Dim S_Variable As String *17
Visual C++ で使用する場合の変数宣言例：char S_Variable[17];

座標種別、形態については以下を参照してください。

解 説

座標種別

YRC1000、YRC1000micro、DX200、DX100

座標種別のデータは、以下の座標名称に相当します。

座標種別	座標名称	座標種別	座標名称
0	ベース座標	:	:
1	ロボット座標	:	:
2	ユーザ座標 1	63	ユーザ座標 62
3	ユーザ座標 2	64	ユーザ座標 63
:	:	65	ツール座標
:	:	66	マスターツール座標

座標種別

FS100

座標種別のデータは、以下の座標名称に相当します。

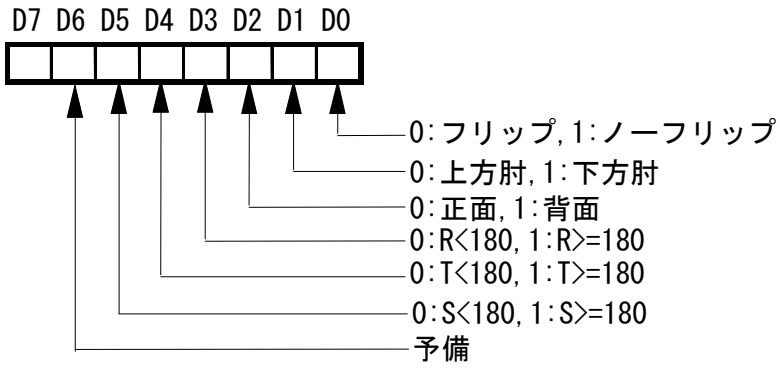
座標種別	座標名称	座標種別	座標名称
0	ベース座標	10	ユーザ座標 9
1	ロボット座標	11	ユーザ座標 10
2	ユーザ座標 1	12	ユーザ座標 11
3	ユーザ座標 2	13	ユーザ座標 12
4	ユーザ座標 3	14	ユーザ座標 13
5	ユーザ座標 4	15	ユーザ座標 14
6	ユーザ座標 5	16	ユーザ座標 15
7	ユーザ座標 6	17	ユーザ座標 16
8	ユーザ座標 7	18	ツール座標
9	ユーザ座標 8	19	マスターツール座標

座標種別

NX100、XRC、MRC

座標種別のデータは、以下の座標名称に相当します。

座標種別	座標名称	座標種別	座標名称
0	ベース座標	14	ユーザ座標 13
1	ロボット座標	15	ユーザ座標 14
2	ユーザ座標 1	16	ユーザ座標 15
3	ユーザ座標 2	17	ユーザ座標 16
4	ユーザ座標 3	18	ユーザ座標 17
5	ユーザ座標 4	19	ユーザ座標 18
6	ユーザ座標 5	20	ユーザ座標 19
7	ユーザ座標 6	21	ユーザ座標 20
8	ユーザ座標 7	22	ユーザ座標 21
9	ユーザ座標 8	23	ユーザ座標 22
10	ユーザ座標 9	24	ユーザ座標 23
11	ユーザ座標 10	25	ユーザ座標 24
12	ユーザ座標 11	26	ツール座標
13	ユーザ座標 12	27	マスターツール座標

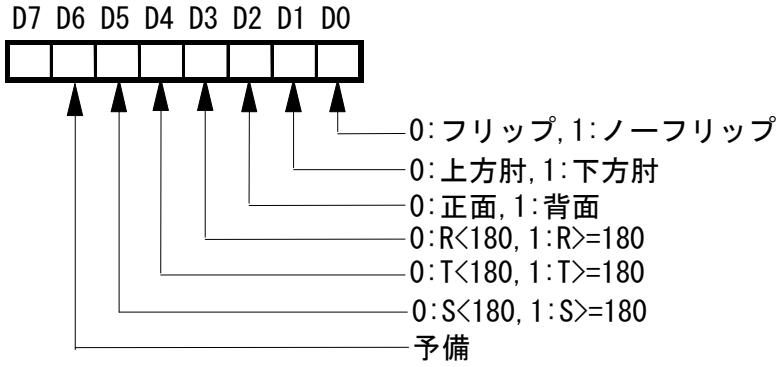
解 説	形態 形態のデータは、以下のビットデータを 10 進数にした値となります。  *MRC, MRC2 の場合は D5 － D7 の値は無視されます。
コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC（シリアルポート、Ethernet）
参 照	「BscGetVarDataEx」

■ BscImov

機 能	現在位置から指定の座標系の増分値で直線動作します。
書 式	_declspec(dllexport) short APIENTRY BscImov(short nCid,char *vtype,double spd,char *framename,short toolno,double *p);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号 *vtype　動作速度の選択 V：制御点速度、VR：姿勢角速度 spd　動作速度（0.1 ～□□□□．□ mm/s、0.1 ～□□□．□ °/s） *framename　座標名称 BASE：ベース座標、ROBOT：ロボット座標、UF1：ユーザー座標 1 ...、TOOL：ツール座標（NX100、XRC、MRC のみ対応） toolno　ツール番号 *p　目標位置格納エリアの先頭ポインタ
	OUT（戻り） 無し
	リターン値 0　：正常終了 その他：エラーコード
解 説	**目標位置** 目標位置データは以下のようになります。

	指定座標系での目標位置
P[0]	X 座標値（単位 mm）
P[1]	Y 座標値（単位 mm）
P[2]	Z 座標値（単位 mm）
P[3]	手首姿勢 Rx（単位°）
P[4]	手首姿勢 Ry（単位°）
P[5]	手首姿勢 Rz（単位°）
P[6]	第 7 軸パルス数（走行軸の場合 mm）
P[7]	第 8 軸パルス数（走行軸の場合 mm）
P[8]	第 9 軸パルス数（走行軸の場合 mm）
P[9]	第 10 軸パルス数
P[10]	第 11 軸パルス数
P[11]	第 12 軸パルス数

* 外部軸無しのシステムでは P[7] － P[11] のデータは「0」を設定してください。

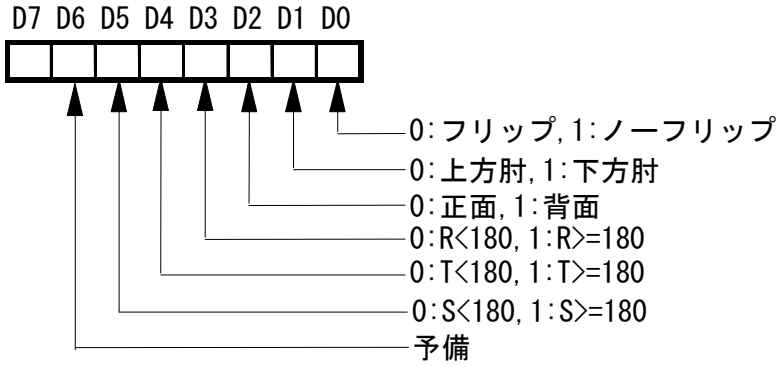
解 説	**形態** 形態のデータは、以下のビットデータを 10 進数にした値となります。  *ERC、ERC2 の場合は D3 － D7 データは無視されます。 *MRC、MRC2 の場合は D5 － D7 データは無視されます。
コントローラ	NX100、XRC、MRC（シリアルポート、Ethernet）、ERC（シリアルポート） *YRC1000、YRC1000micro、DX200、DX100、FS100 の場合は BscImovEx2 を参照してください。
参 照	「BscMov」「BscMovj」「BscMovl」「BscImovEx2」

■ BscImovEx

機 能	現在位置から指定の座標系の増分値で直線動作します。(7 軸以上対応)
書 式	_declspec(dllexport) short APIENTRY BscImovEx(short nCid,char *vtype,double spd,char *framename,short toolno,double *p,short axisNum);
引 数	**IN（引き渡し）** nCid 通信ハンドラの ID 番号 *vtype 動作速度の選択 V：制御点速度、VR：姿勢角速度 spd 動作速度（0.1 ～□□□□．□ mm/s、0.1 ～□□□．□ °/s） *framename 座標名称 BASE：ベース座標、ROBOT：ロボット座標、 UF1：ユーザー座標 1 ...、 TOOL：ツール座標 toolno ツール番号 *p 目標位置格納エリアの先頭ポインタ axisNum 軸数
	OUT（戻り） 無し
	リターン値 0 ：正常終了 その他：エラーコード
解 説	**制約事項** 本関数は互換性維持のためのものです。BscImovEx2 を使用ください。
	目標位置 目標位置データは以下のようになります。

	指定座標系での目標位置	7 軸の場合
P[0]	X 座標値（単位 mm）	X 座標値（単位 mm）
P[1]	Y 座標値（単位 mm）	Y 座標値（単位 mm）
P[2]	Z 座標値（単位 mm）	Z 座標値（単位 mm）
P[3]	手首姿勢 Rx（単位°）	手首姿勢 Rx（単位°）
P[4]	手首姿勢 Ry（単位°）	手首姿勢 Ry（単位°）
P[5]	手首姿勢 Rz（単位°）	手首姿勢 Rz（単位°）
P[6]	第 7 軸パルス数(走行軸の場合 mm)	Re（単位°）
P[7]	第 8 軸パルス数(走行軸の場合 mm)	第 8 軸パルス数（走行軸の場合 mm）
P[8]	第 9 軸パルス数(走行軸の場合 mm)	第 9 軸パルス数（走行軸の場合 mm）
P[9]	第 10 軸パルス数	第 10 軸パルス数（走行軸の場合 mm）
P[10]	第 11 軸パルス数	第 11 軸パルス数
P[11]	第 12 軸パルス数	第 12 軸パルス数
P[12]	－	第 13 軸パルス数

* 外部軸無しのシステムでは P[6] － P[11] のデータは「0」を設定してください。7 軸の場合は、P[7] － P[12] のデータは「0」を設定してください。

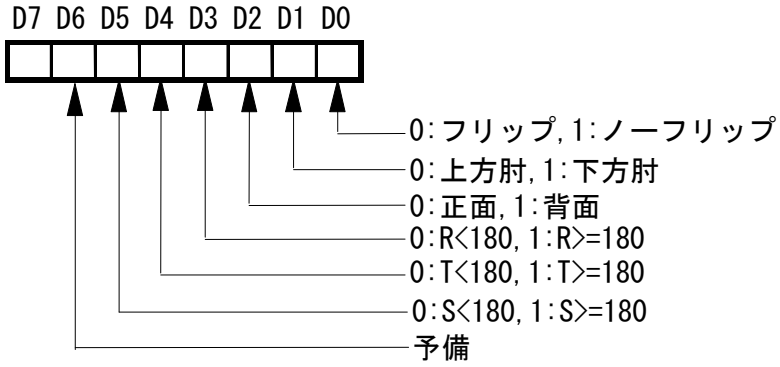
解 説	**形態** 形態のデータは、以下のビットデータを 10 進数にした値となります。 
コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100（シリアルポート、Ethernet）
参 照	「BscMovEx2」「BscMovjEx」「BscMovlEx」

■ BscImovEx2

機 能	現在位置から指定の座標系の増分値で直線動作します。(7 軸以上対応)
書 式	_declspec(dllexport) short APIENTRY BscImovEx2(short nCid,short ctype,char *vtype,double spd,char *framename,short toolno,double *p,short axisNum);
引 数	**IN (引き渡し)** nCid 通信ハンドラの ID 番号 ctype コントローラの選択 0 : ERC、1 : MRC、2 : XRC、3 : NX100、4 : YRC1000、YRC1000micro、DX200、DX100、5 : FS100 *vtype 動作速度の選択 V : 制御点速度、VR : 姿勢角速度 spd 動作速度 (0.1 ～□□□□. □ mm/s、0.1 ～□□□. □ °/s) *framename 座標名称 BASE : ベース座標、ROBOT : ロボット座標、UF1 : ユーザー座標 1 ...、TOOL : ツール座標 (YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC のみ対応) toolno ツール番号 *p 目標位置格納エリアの先頭ポインタ axisNum ロボット軸数
	OUT (戻り) 無し
	リターン値 0 : 正常終了 その他 : エラーコード
解 説	**制約事項** 7 軸以上には、YRC1000、YRC1000micro、DX200、DX100、FS100 のみ対応します。
	目標位置 目標位置データは以下のようになります。

	指定座標系での目標位置	7 軸の場合
P[0]	X 座標値 (単位 mm)	X 座標値 (単位 mm)
P[1]	Y 座標値 (単位 mm)	Y 座標値 (単位 mm)
P[2]	Z 座標値 (単位 mm)	Z 座標値 (単位 mm)
P[3]	手首姿勢 Rx (単位°)	手首姿勢 Rx (単位°)
P[4]	手首姿勢 Ry (単位°)	手首姿勢 Ry (単位°)
P[5]	手首姿勢 Rz (単位°)	手首姿勢 Rz (単位°)
P[6]	第 7 軸パルス数(走行軸の場合 mm)	Re (単位°)
P[7]	第 8 軸パルス数(走行軸の場合 mm)	第 8 軸パルス数 (走行軸の場合 mm)
P[8]	第 9 軸パルス数(走行軸の場合 mm)	第 9 軸パルス数 (走行軸の場合 mm)
P[9]	第 10 軸パルス数	第 10 軸パルス数 (走行軸の場合 mm)
P[10]	第 11 軸パルス数	第 11 軸パルス数
P[11]	第 12 軸パルス数	第 12 軸パルス数
P[12]	―	第 13 軸パルス数

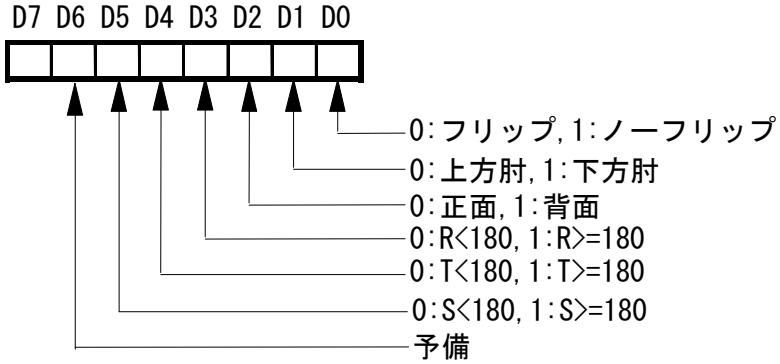
* 外部軸無しのシステムでは P[6] ― P[11] のデータは「0」を設定してください。7 軸の場合は、P[7] ― P[12] のデータは「0」を設定してください。

解 説	**形態** 形態のデータは、以下のビットデータを 10 進数にした値となります。  *ERC、ERC2 の場合は D3 － D7 データは無視されます。 *MRC、MRC2 の場合は D5 － D7 データは無視されます。
コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC（シリアルポート、Ethernet） ERC（シリアルポート）
参 照	「BscMovEx2」「BscMovjEx」「BscMovlEx」

■ BscMDSP

機 能	メッセージデータを送信します。
書 式	_declspec(dllexport) short APIENTRY BscMDSP(short nCid,char *ptr);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号 *ptr　メッセージ格納ポインタ
	OUT（戻り） 無し
	リターン値 0　　：正常終了 その他：エラーコード
解 説	**メッセージ文字数** メッセージ文字数は以下の制限があります。 YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC の場合 ：MAX.30 のキャラクタの文字列。 ERC の場合　：MAX.28 のキャラクタの文字列。
コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC（シリアルポート、Ethernet） ERC（シリアルポート）

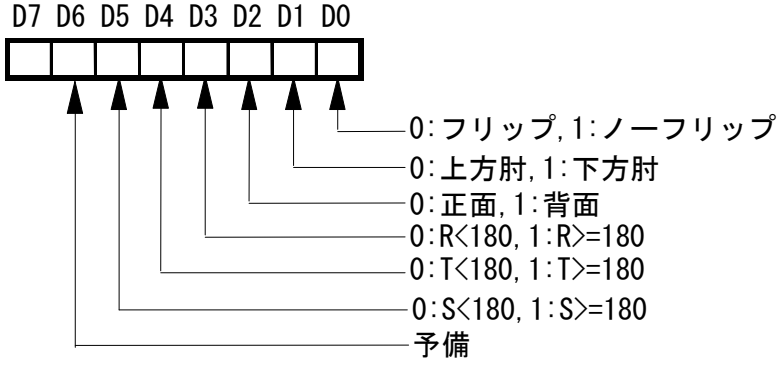
■ BscMov

機 能	現在位置から指定の座標系の増分値で直線動作します。
書 式	_declspec(dllexport) short APIENTRY BscMov(short nCid,char *movtype,char *vtype,double spd,char *framename,short rconf,short toolno,double *p);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号 *movtype　動作種別 MOVJ：リンク動作、MOVL：直線動作、IMOV：直線動作（増分値） *vtype　動作速度の選択 V：制御点速度、VR：姿勢角速度、VJ：リンク速度 spd　動作速度（0.1 ～□□□□．□ mm/s、0.1 ～□□□．□ °/s、0.01 ～ 100.00%） *framename　座標名称 BASE：ベース座標、ROBOT：ロボット座標、UF1：ユーザー座標 1...、TOOL：ツール座標（NX100、XRC、MRC で動作種別が「IMOV」の時のみ対応） rconf　形態 toolno　ツール番号 *p　目標位置格納エリアの先頭ポインタ
	OUT（戻り） 無し
	リターン値 0　：正常終了 その他：エラーコード
解 説	**形態** 形態のデータは、以下のビットデータを 10 進数にした値となります。  *ERC、ERC2 の場合は D3 － D7 データは無視されます。 *MRC、MRC2 の場合は D5 － D7 データは無視されます。

解 説	**目標位置** 目標位置データは以下のようになります。 ＊外部軸無しのシステムでは P[7] － P[11] のデータは「0」を設定してください。
コントローラ	NX100、XRC、MRC（シリアルポート、Ethernet） ERC（シリアルポート） *YRC1000、YRC1000micro、DX200、DX100、FS100 の場合は BscMovEx2 を参照してください。
参 照	「BscMovj」「BscMovl」「BscImov」「BscMovEx2」

	指定座標系での目標位置
P[0]	X 座標値（単位 mm）
P[1]	Y 座標値（単位 mm）
P[2]	Z 座標値（単位 mm）
P[3]	手首姿勢 Rx（単位°）
P[4]	手首姿勢 Ry（単位°）
P[5]	手首姿勢 Rz（単位°）
P[6]	第 7 軸パルス数（走行軸の場合 mm）
P[7]	第 8 軸パルス数（走行軸の場合 mm）
P[8]	第 9 軸パルス数（走行軸の場合 mm）
P[9]	第 10 軸パルス数
P[10]	第 11 軸パルス数
P[11]	第 12 軸パルス数

■ BscMovEx

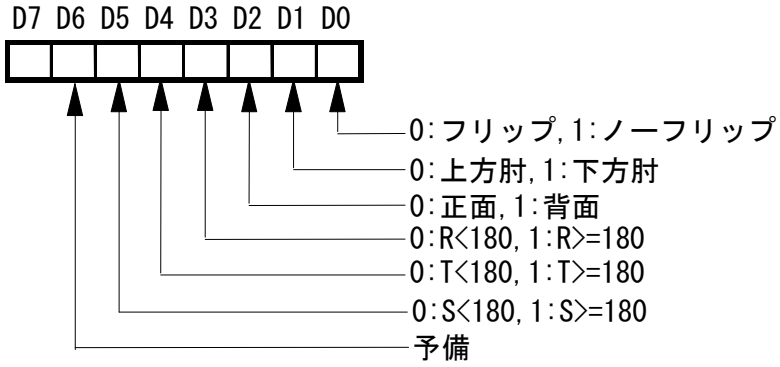
機 能	現在位置から指定の座標系の増分値で直線動作します。(7 軸以上対応)
書 式	_declspec(dllexport) short APIENTRY BscMovEx(short nCid,char *movtype,char *vtype,double spd,char *framename,long rconf,short toolno,double *p,short axisNum);
引 数	**IN（引き渡し）** nCid 通信ハンドラの ID 番号 *movtype 動作種別 MOVJ：リンク動作、MOVL：直線動作、IMOV：直線動作（増分値） *vtype 動作速度の選択 V：制御点速度、VR：姿勢角速度、VJ：リンク速度 spd 動作速度（0.1 ～□□□□．□ mm/s、0.1 ～□□□．□ °/s、0.01 ～ 100.00%） *framename 座標名称 BASE：ベース座標、ROBOT：ロボット座標、UF1：ユーザー座標 1...、TOOL：ツール座標（動作種別が「IMOV」の時のみ対応） rconf 形態 toolno ツール番号 *p 目標位置格納エリアの先頭ポインタ axisNum 軸数
	OUT（戻り） 無し
	リターン値 0 ：正常終了 その他：エラーコード
解 説	**制約事項** 本関数は互換性維持のためのものです。BscMovEx2 を使用ください。
	形態 形態のデータは、以下のビットデータを 10 進数にした値となります。 

解 説	**目標位置** 目標位置データは以下のようになります。 ＊外部軸無しのシステムでは P[6] － P[11] のデータは「0」を設定してください。7 軸の場合は、P[7] － P[12] のデータは「0」を設定してください。
コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100（シリアルポート、Ethernet）
参 照	「BscMovjEx」 「BscMovlEx」 「BscImovEx」

	指定座標系での目標位置	7 軸の場合
P[0]	X 座標値（単位 mm）	X 座標値（単位 mm）
P[1]	Y 座標値（単位 mm）	Y 座標値（単位 mm）
P[2]	Z 座標値（単位 mm）	Z 座標値（単位 mm）
P[3]	手首姿勢 Rx（単位°）	手首姿勢 Rx（単位°）
P[4]	手首姿勢 Ry（単位°）	手首姿勢 Ry（単位°）
P[5]	手首姿勢 Rz（単位°）	手首姿勢 Rz（単位°）
P[6]	第 7 軸パルス数（走行軸の場合 mm）	Re（単位°）
P[7]	第 8 軸パルス数（走行軸の場合 mm）	第 8 軸パルス数（走行軸の場合 mm）
P[8]	第 9 軸パルス数（走行軸の場合 mm）	第 9 軸パルス数（走行軸の場合 mm）
P[9]	第 10 軸パルス数	第 10 軸パルス数（走行軸の場合 mm）
P[10]	第 11 軸パルス数	第 11 軸パルス数
P[11]	第 12 軸パルス数	第 12 軸パルス数
P[12]	－	第 13 軸パルス数

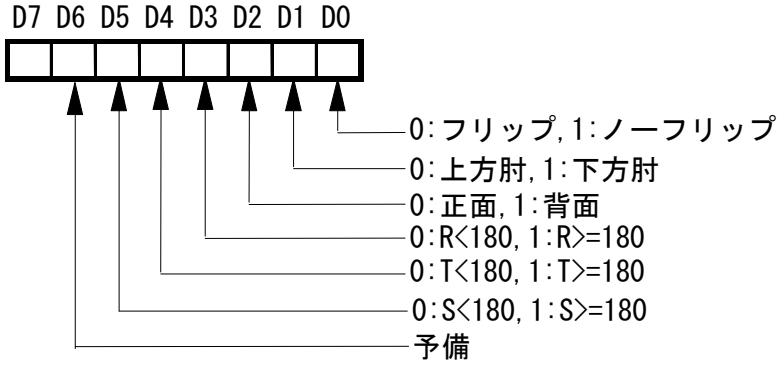
■ BscMovEx2

機 能	現在位置から指定の座標系の増分値で直線動作します。(7 軸以上対応)
書 式	_declspec(dllexport) short APIENTRY BscMovEx(short nCid,short ctype,char *movtype,char *vtype,double spd,char *framename,long rconf,short toolno,double *p,short axisNum);
引 数	**IN (引き渡し)** nCid 通信ハンドラの ID 番号 ctype コントローラの選択 0 : ERC、1 : MRC、2 : XRC、3 : NX100、4 : YRC1000、YRC1000micro、DX200、DX100、5 : FS100 *movtype 動作種別 MOVJ : リンク動作、MOVL : 直線動作、IMOV : 直線動作 (増分値) *vtype 動作速度の選択 V : 制御点速度、VR : 姿勢角速度、VJ : リンク速度 spd 動作速度 (0.1 ～□□□□. □ mm/s、0.1 ～□□□. □ °/s、0.01 ～ 100.00%) *framename 座標名称 BASE : ベース座標、ROBOT : ロボット座標、UF1 : ユーザー座標 1...、TOOL : ツール座標 (YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC で動作種別が「IMOV」の時のみ対応) rconf 形態 toolno ツール番号 *p 目標位置格納エリアの先頭ポインタ axisNum ロボット軸数
	OUT (戻り) 無し
	リターン値 0 : 正常終了 その他 : エラーコード
解 説	**制約事項** 7 軸以上には、YRC1000、YRC1000micro、DX200、DX100、FS100 のみ対応します。

解 説	**形態** 形態のデータは、以下のビットデータを 10 進数にした値となります。  *ERC、ERC2 の場合は D3 － D7 データは無視されます。 *MRC、MRC2 の場合は D5 － D7 データは無視されます。
	目標位置 目標位置データは以下のようになります。 ＊外部軸無しのシステムでは P[6] － P[11] のデータは「0」を設定してください。7 軸の場合は、P[7] － P[12] のデータは「0」を設定してください。
コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC（シリアルポート、Ethernet） ERC（シリアルポート）
参 照	「BscMovjEx」「BscMovlEx」「BscImovEx2」

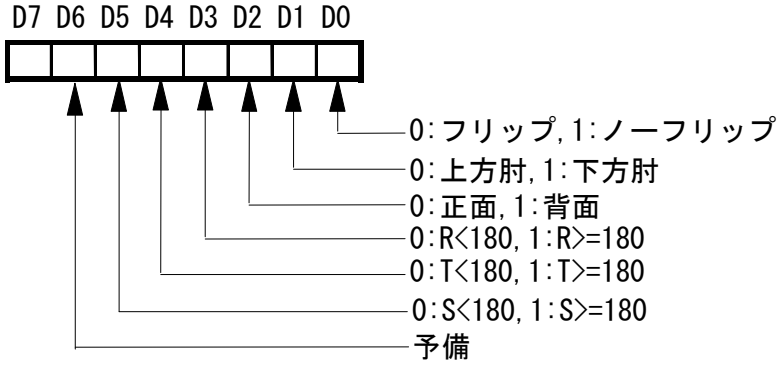
	指定座標系での目標位置	7 軸の場合
P[0]	X 座標値（単位 mm）	X 座標値（単位 mm）
P[1]	Y 座標値（単位 mm）	Y 座標値（単位 mm）
P[2]	Z 座標値（単位 mm）	Z 座標値（単位 mm）
P[3]	手首姿勢 Rx（単位°）	手首姿勢 Rx（単位°）
P[4]	手首姿勢 Ry（単位°）	手首姿勢 Ry（単位°）
P[5]	手首姿勢 Rz（単位°）	手首姿勢 Rz（単位°）
P[6]	第 7 軸パルス数(走行軸の場合 mm)	Re（単位°）
P[7]	第 8 軸パルス数(走行軸の場合 mm)	第 8 軸パルス数（走行軸の場合 mm）
P[8]	第 9 軸パルス数(走行軸の場合 mm)	第 9 軸パルス数（走行軸の場合 mm）
P[9]	第 10 軸パルス数	第 10 軸パルス数（走行軸の場合 mm）
P[10]	第 11 軸パルス数	第 11 軸パルス数
P[11]	第 12 軸パルス数	第 12 軸パルス数
P[12]	－	第 13 軸パルス数

■ BscMovj

機 能	指定の座標系の位置へリンク動作します。
書 式	_declspec(dllexport) short APIENTRY BscMovj(short nCid,double spd,char *framename,short rconf,short toolno,double *p);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号 spd　動作速度（0.01 ～ 100.0%） *framename　座標名称 BASE：ベース座標、ROBOT：ロボット座標、UF1：ユーザー座標 1 ... rconf　形態 toolno　ツール番号 *p　目標位置格納エリアの先頭ポインタ
	OUT（戻り） 無し
	リターン値 0　：正常終了 その他：エラーコード
解 説	**形態** 形態のデータは、以下のビットデータを 10 進数にした値となります。  *ERC、ERC2 の場合は D3 － D7 データは無視されます。 *MRC、MRC2 の場合は D5 － D7 データは無視されます。

解 説	**目標位置** 目標位置データは以下のようになります。 <table><tr><th></th><th>指定座標系での目標位置</th></tr><tr><td>P[0]</td><td>X 座標値（単位 mm）</td></tr><tr><td>P[1]</td><td>Y 座標値（単位 mm）</td></tr><tr><td>P[2]</td><td>Z 座標値（単位 mm）</td></tr><tr><td>P[3]</td><td>手首姿勢 Rx（単位°）</td></tr><tr><td>P[4]</td><td>手首姿勢 Ry（単位°）</td></tr><tr><td>P[5]</td><td>手首姿勢 Rz（単位°）</td></tr><tr><td>P[6]</td><td>第 7 軸パルス数(走行軸の場合 mm)</td></tr><tr><td>P[7]</td><td>第 8 軸パルス数(走行軸の場合 mm)</td></tr><tr><td>P[8]</td><td>第 9 軸パルス数(走行軸の場合 mm)</td></tr><tr><td>P[9]</td><td>第 10 軸パルス数</td></tr><tr><td>P[10]</td><td>第 11 軸パルス数</td></tr><tr><td>P[11]</td><td>第 12 軸パルス数</td></tr></table> * 外部軸無しのシステムでは P[7] － P[11] のデータは「0」を設定してください。
コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC（シリアルポート、Ethernet） ERC（シリアルポート） *YRC1000、YRC1000micro、DX200、DX100、FS100　7 軸以上の場合は BscMovjEx を参照してください。
参 照	「BscMov」「BscMovl」「BscImov」

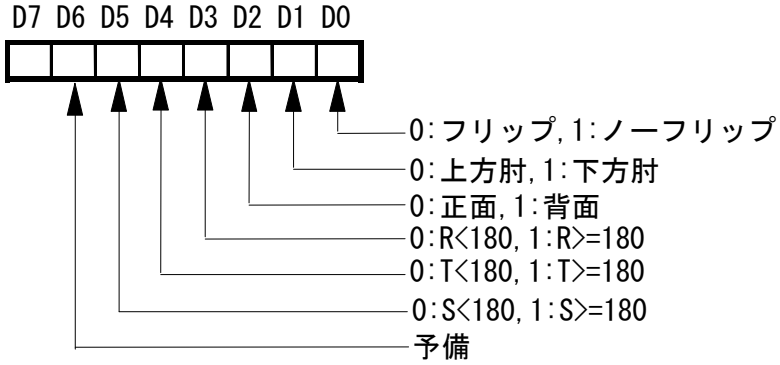
■ BscMovjEx

機 能	指定の座標系の位置へリンク動作します。(7 軸以上対応)
書 式	_declspec(dllexport) short APIENTRY BscMovjEx(short nCid,double spd,char *framename,long rconf,short toolno,double *p,short axisNum);
引 数	**IN（引き渡し）** nCid 通信ハンドラの ID 番号 spd 動作速度（0.01 ～ 100.0%） *framename 座標名称 BASE：ベース座標、ROBOT：ロボット座標、UF1：ユーザー座標 1 ... rconf 形態 toolno ツール番号 *p 目標位置格納エリアの先頭ポインタ axisNum 軸数
	OUT（戻り） 無し
	リターン値 0 ：正常終了 その他：エラーコード
解 説	**形態** 形態のデータは、以下のビットデータを 10 進数にした値となります。 

解 説	**目標位置** 目標位置データは以下のようになります。 * 外部軸無しのシステムでは P[6] － P[11] のデータは「0」を設定してください。7 軸の場合は、P[7] － P[12] のデータは「0」を設定してください。
コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100（シリアルポート、Ethernet）
参 照	「BscMovEx2」「BscMovlEx」「BscImovEx2」

	指定座標系での目標位置	7 軸の場合
P[0]	X 座標値（単位 mm）	X 座標値（単位 mm）
P[1]	Y 座標値（単位 mm）	Y 座標値（単位 mm）
P[2]	Z 座標値（単位 mm）	Z 座標値（単位 mm）
P[3]	手首姿勢 Rx（単位°）	手首姿勢 Rx（単位°）
P[4]	手首姿勢 Ry（単位°）	手首姿勢 Ry（単位°）
P[5]	手首姿勢 Rz（単位°）	手首姿勢 Rz（単位°）
P[6]	第 7 軸パルス数（走行軸の場合 mm）	Re（単位°）
P[7]	第 8 軸パルス数（走行軸の場合 mm）	第 8 軸パルス数（走行軸の場合 mm）
P[8]	第 9 軸パルス数（走行軸の場合 mm）	第 9 軸パルス数（走行軸の場合 mm）
P[9]	第 10 軸パルス数	第 10 軸パルス数（走行軸の場合 mm）
P[10]	第 11 軸パルス数	第 11 軸パルス数
P[11]	第 12 軸パルス数	第 12 軸パルス数
P[12]	－	第 13 軸パルス数

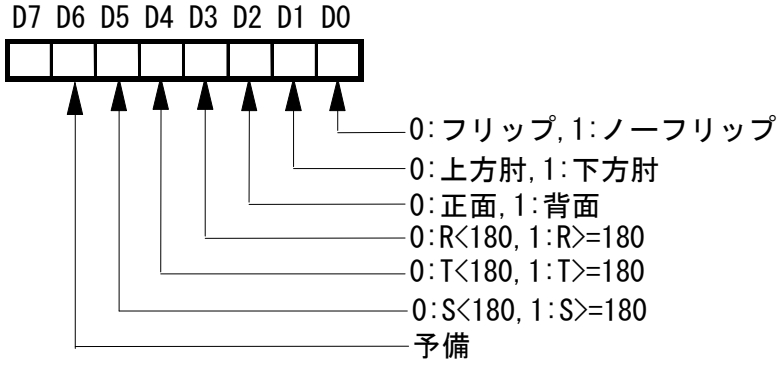
■ BscMovl

機 能	指定の座標系の位置へ直線動作します。
書 式	_declspec(dllexport) short APIENTRY BscMovl(short nCid,char *vtype,double spd,char *framename,short rconf,short toolno,double *p);
引 数	**IN（引き渡し）** nCid 通信ハンドラの ID 番号 *vtype 動作速度の選択 V：制御点速度、VR：姿勢角速度 spd 動作速度（0.1 ～□□□□．□ mm/s,0.1 ～□□□．□ °/s） *framename 座標名称 BASE：ベース座標、ROBOT：ロボット座標、UF1：ユーザー座標１ ... rconf 形態 toolno ツール番号 *p 目標位置格納エリアの先頭ポインタ
	OUT（戻り） 無し
	リターン値 0 ：正常終了 その他：エラーコード
解 説	**形態** 形態のデータは、以下のビットデータを 10 進数にした値となります。  *ERC、ERC2 の場合は D3 － D7 データは無視されます。 *MRC、MRC ２の場合は D5 － D7 データは無視されます。

解 説	**目標位置** 目標位置データは以下のようになります。 * 外部軸無しのシステムでは P[7] ー P[11] のデータは「0」を設定してください。
コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC（シリアルポート、Ethernet） ERC（シリアルポート） *YRC1000、YRC1000micro、DX200、DX100、FS100　7 軸以上の場合は BscMovlEx を参照してください。
参 照	「BscMov」「BscMovj」「BscImov」

	指定座標系での目標位置
P[0]	X 座標値（単位 mm）
P[1]	Y 座標値（単位 mm）
P[2]	Z 座標値（単位 mm）
P[3]	手首姿勢 Rx（単位°）
P[4]	手首姿勢 Ry（単位°）
P[5]	手首姿勢 Rz（単位°）
P[6]	第 7 軸パルス数（走行軸の場合 mm）
P[7]	第 8 軸パルス数（走行軸の場合 mm）
P[8]	第 9 軸パルス数（走行軸の場合 mm）
P[9]	第 10 軸パルス数
P[10]	第 11 軸パルス数
P[11]	第 12 軸パルス数

■ BscMovlEx

機 能	指定の座標系の位置へ直線動作します。(7 軸以上対応)
書 式	_declspec(dllexport) short APIENTRY BscMovlEx(short nCid,char *vtype,double spd,char *framename,long rconf,short toolno,double *p,short axisNum);
引 数	**IN（引き渡し）** nCid 通信ハンドラの ID 番号 *vtype 動作速度の選択 V：制御点速度、VR：姿勢角速度 spd 動作速度（0.1 ～□□□□．□ mm/s,0.1 ～□□□．□ °/s） *framename 座標名称 BASE：ベース座標、ROBOT：ロボット座標、UF1：ユーザー座標 1 ... rconf 形態 toolno ツール番号 *p 目標位置格納エリアの先頭ポインタ axisNum 軸数
	OUT（戻り） 無し
	リターン値 0 ：正常終了 その他：エラーコード
解 説	**形態** 形態のデータは、以下のビットデータを 10 進数にした値となります。 

解 説	**目標位置** 目標位置データは以下のようになります。 * 外部軸無しのシステムでは P[6] － P[11] のデータは「0」を設定してください。7 軸の場合は、P[7] － P[12] のデータは「0」を設定してください。
コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100（シリアルポート、Ethernet）
参 照	「BscMovEx2」「BscMovjEx」「BscImovEx2」

	指定座標系での目標位置	7 軸の場合
P[0]	X 座標値（単位 mm）	X 座標値（単位 mm）
P[1]	Y 座標値（単位 mm）	Y 座標値（単位 mm）
P[2]	Z 座標値（単位 mm）	Z 座標値（単位 mm）
P[3]	手首姿勢 Rx（単位°）	手首姿勢 Rx（単位°）
P[4]	手首姿勢 Ry（単位°）	手首姿勢 Ry（単位°）
P[5]	手首姿勢 Rz（単位°）	手首姿勢 Rz（単位°）
P[6]	第 7 軸パルス数（走行軸の場合 mm）	Re（単位°）
P[7]	第 8 軸パルス数（走行軸の場合 mm）	第 8 軸パルス数（走行軸の場合 mm）
P[8]	第 9 軸パルス数（走行軸の場合 mm）	第 9 軸パルス数（走行軸の場合 mm）
P[9]	第 10 軸パルス数	第 10 軸パルス数（走行軸の場合 mm）
P[10]	第 11 軸パルス数	第 11 軸パルス数
P[11]	第 12 軸パルス数	第 12 軸パルス数
P[12]	－	第 13 軸パルス数

■ BscOPLock

機 能	ロボットをインタロック状態にします。
書 式	_declspec(dllexport) short APIENTRY BscOPLock(short nCid);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号
	OUT（戻り） 無し
	リターン値 0　　：正常終了 その他：エラーコード
解 説	**インタロック状態** インタロックが設定されると下記以外はインタロックされます。
	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100 の場合 * プログラミングペンダントからの非常停止、ホールド *I/O のモード切り替え、外部スタート、外部サーボオン、サイクル切り替え、I/O 禁止、PP/PANEL 禁止、マスタ呼び出しを除く入力信号 プログラミングペンダントが編集モード中、または他の機能でファイルアクセス 中の場合は、インタロック状態にできません。
	XRC の場合 * プログラミングペンダントからのホールド * プレイバックボックスからのホールド、非常停止 *I/O の 404x,405x,409x を除く入力信号（外部ホールド、外部サーボオフを含む） プログラミングペンダントが編集モード中、または他の機能でファイルアクセス 中の場合は、インタロック状態にできません。
	MRC の場合 * プログラミングペンダントからのホールド * プレイバックボックスからのホールド、非常停止 *I/O の 404x,405x を除く入力信号（外部ホールド、外部サーボオフを含む） プログラミングペンダントが編集モード中、または他の機能でファイルアクセス中の場合は、インタロック状態にできません。
	ERC の場合 * パネル操作でのスタート及びホールドボタン * パネル操作での非常停止ボタン * パネル操作でのサーボ電源投入ボタン
コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC（シリアルポート、Ethernet） ERC（シリアルポート）

参 照	「BscOPUnLock」

■ BscOPUnLock

機 能	ロボットのインタロック状態を解除します。
書 式	_declspec(dllexport) short APIENTRY BscOPUnLock(short nCid);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号
	OUT（戻り） 無し
	リターン値 0　　：正常終了 その他：エラーコード
コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC（シリアルポート、Ethernet） ERC（シリアルポート）
参 照	「BscOPLock」

■ BscPMov

機 能	指定パルス位置へ移動します。
書 式	_declspec(dllexport) short APIENTRY BscPMov(short nCid,char *movtype,char *vtype,double spd,short toolno,double *p);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号 *movtype　動作種別 MOVJ：リンク動作、MOVL：直線動作、 *vtype　動作速度の選択 V：制御点速度、VR：姿勢角速度、 VJ：リンク速度 spd　動作速度（0.1 ～□□□□．□ mm/s、0.1 ～□□□．□ °/s、 0.01 ～ 100.00%） toolno　ツール番号 *p　目標位置格納エリアの先頭ポインタ
	OUT（戻り） 無し
	リターン値 0　：正常終了 その他：エラーコード
解 説	**目標位置** 目標位置データは以下のようになります。 | | パルスでの目標位置 | |---|---| | P[0] | S 軸パルス数 | | P[1] | L 軸パルス数 | | P[2] | U 軸パルス数 | | P[3] | R 軸パルス数 | | P[4] | B 軸パルス数 | | P[5] | T 軸パルス数 | | P[6] | 第 7 軸パルス数 | | P[7] | 第 8 軸パルス数 | | P[8] | 第 9 軸パルス数 | | P[9] | 第 10 軸パルス数 | | P[10] | 第 11 軸パルス数 | | P[11] | 第 12 軸パルス数 | * 外部軸無しのシステムでは P[7] － P[11] のデータは「0」を設定してください。
コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC（シリアルポート、Ethernet） ERC（シリアルポート） *YRC1000、YRC1000micro、DX200、DX100、FS100　7 軸以上の場合は BscPMovEx を参照してください。
参 照	「BscPMovj」「BscPMovl」

■ BscPMovEx

機 能	指定パルス位置へ移動します。(7 軸以上対応)
書 式	_declspec(dllexport) short APIENTRY BscPMovEx(short nCid,char *movtype,char *vtype,double spd,short toolno,double *p,short axisNum);
引 数	**IN (引き渡し)** nCid 通信ハンドラの ID 番号 *movtype 動作種別 MOVJ：リンク動作、MOVL：直線動作 *vtype 動作速度の選択 V：制御点速度、VR：姿勢角速度、VJ：リンク速度 spd 動作速度 (0.1 ～□□□□. □ mm/s、0.1 ～□□□. □ °/s、0.01 ～ 100.00%) toolno ツール番号 *p 目標位置格納エリアの先頭ポインタ axisNum 軸数
	OUT (戻り) 無し
	リターン値 0 ：正常終了 その他：エラーコード
解 説	**目標位置** 目標位置データは以下のようになります。

	パルスでの目標位置	7 軸の場合
P[0]	S 軸パルス数	S 軸パルス数
P[1]	L 軸パルス数	L 軸パルス数
P[2]	U 軸パルス数	U 軸パルス数
P[3]	R 軸パルス数	R 軸パルス数
P[4]	B 軸パルス数	B 軸パルス数
P[5]	T 軸パルス数	T 軸パルス数
P[6]	第 7 軸パルス数	E 軸パルス数
P[7]	第 8 軸パルス数	第 8 軸パルス数
P[8]	第 9 軸パルス数	第 9 軸パルス数
P[9]	第 10 軸パルス数	第 10 軸パルス数
P[10]	第 11 軸パルス数	第 11 軸パルス数
P[11]	第 12 軸パルス数	第 12 軸パルス数
P[12]	―	第 13 軸パルス数

* 外部軸無しのシステムでは P[6] ― P[11] のデータは「0」を設定してください。7 軸の場合は、P[7] ― P[12] のデータは「0」を設定してください。

コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100 (シリアルポート、Ethernet)
参 照	「BscPMovjEx」「BscPMovlEx」

■ BscPMovj

機 能	指定パルス位置にリンク動作します。
書 式	_declspec(dllexport) short APIENTRY BscPMovj(short nCid,double spd,short toolno,double *p);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号 spd　動作速度（0.01 ～ 100.00%） toolno　ツール番号 *p　目標位置格納エリアの先頭ポインタ
	OUT（戻り） 無し
	リターン値 0　：正常終了 その他：エラーコード
解 説	**目標位置** 目標位置データは以下のようになります。 * 外部軸無しのシステムでは P[7]-P[11] のデータは「0」を設定してください。
コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC（シリアルポート、Ethernet） ERC（シリアルポート） *YRC1000、YRC1000micro、DX200、DX100、FS100　7 軸以上の場合は BscPMovjEx を参照してください。
参 照	「BscPMovl」「BscPMov」

	パルスでの目標位置
P[0]	S 軸パルス数
P[1]	L 軸パルス数
P[2]	U 軸パルス数
P[3]	R 軸パルス数
P[4]	B 軸パルス数
P[5]	T 軸パルス数
P[6]	第 7 軸パルス数
P[7]	第 8 軸パルス数
P[8]	第 9 軸パルス数
P[9]	第 10 軸パルス数
P[10]	第 11 軸パルス数
P[11]	第 12 軸パルス数

■ BscPMovjEx

機 能	指定パルス位置にリンク動作します。（7 軸以上対応）
書 式	_declspec(dllexport) short APIENTRY BscPMovjEx(short nCid,double spd,short toolno,double *p,short axisNum);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号 spd　動作速度（0.01 ～ 100.00%） toolno　ツール番号 *p　目標位置格納エリアの先頭ポインタ axisNum　軸数
	OUT（戻り） 無し
	リターン値 0　：正常終了 その他：エラーコード
解 説	**目標位置** 目標位置データは以下のようになります。

	パルスでの目標位置	7 軸の場合
P[0]	S 軸パルス数	S 軸パルス数
P[1]	L 軸パルス数	L 軸パルス数
P[2]	U 軸パルス数	U 軸パルス数
P[3]	R 軸パルス数	R 軸パルス数
P[4]	B 軸パルス数	B 軸パルス数
P[5]	T 軸パルス数	T 軸パルス数
P[6]	第 7 軸パルス数	E 軸パルス数
P[7]	第 8 軸パルス数	第 8 軸パルス数
P[8]	第 9 軸パルス数	第 9 軸パルス数
P[9]	第 10 軸パルス数	第 10 軸パルス数
P[10]	第 11 軸パルス数	第 11 軸パルス数
P[11]	第 12 軸パルス数	第 12 軸パルス数
P[12]	―	第 13 軸パルス数

* 外部軸無しのシステムでは P[6] ― P[11] のデータは「0」を設定してください。7 軸の場合は、P[7] ― P[12] のデータは「0」を設定してください。

コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100（シリアルポート、Ethernet）
参 照	「BscPMovlEx」「BscPMovEx」

■ BscPMovl

機 能	指定パルス位置に直線動作します。
書 式	_declspec(dllexport) short APIENTRY BscPMovl(short nCid,char *vtype,double spd,short toolno,double *p);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号 *vtype　動作速度の選択 V：制御点速度、VR：姿勢角速度 spd　動作速度（0.1 ～□□□□．□ mm/s、0.1 ～□□□．□ °/s） toolno　ツール番号 *p　目標位置格納エリアの先頭ポインタ
	OUT（戻り） 無し
	リターン値 0　：正常終了 その他：エラーコード
解 説	**目標位置** 目標位置データは以下のようになります。 パルスでの目標位置 P[0] S 軸パルス数 P[1] L 軸パルス数 P[2] U 軸パルス数 P[3] R 軸パルス数 P[4] B 軸パルス数 P[5] T 軸パルス数 P[6] 第 7 軸パルス数 P[7] 第 8 軸パルス数 P[8] 第 9 軸パルス数 P[9] 第 10 軸パルス数 P[10] 第 11 軸パルス数 P[11] 第 12 軸パルス数 * 外部軸無しのシステムでは P[7]-P[11] のデータは「0」を設定してください。
コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC（シリアルポート、Ethernet） ERC（シリアルポート） *YRC1000、YRC1000micro、DX200、DX100、FS100　7 軸以上の場合は BscPMovlEx を参照してください。
参 照	「BscPMovj」「BscPMov」

■ BscPMovlEx

機 能	指定パルス位置に直線動作します。
書 式	_declspec(dllexport) short APIENTRY BscPMovlEx(short nCid,char *vtype,double spd,short toolno,double *p,short axisNum);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号 *vtype　動作速度の選択 V：制御点速度、VR：姿勢角速度 spd　動作速度（0.1 ～□□□□．□ mm/s、0.1 ～□□□．□ °/s） toolno　ツール番号 *p　目標位置格納エリアの先頭ポインタ axisNum　軸数
	OUT（戻り） 無し
	リターン値 0　：正常終了 その他：エラーコード
解 説	**目標位置** 目標位置データは以下のようになります。

	パルスでの目標位置	7 軸の場合
P[0]	S 軸パルス数	S 軸パルス数
P[1]	L 軸パルス数	L 軸パルス数
P[2]	U 軸パルス数	U 軸パルス数
P[3]	R 軸パルス数	R 軸パルス数
P[4]	B 軸パルス数	B 軸パルス数
P[5]	T 軸パルス数	T 軸パルス数
P[6]	第 7 軸パルス数	E 軸パルス数
P[7]	第 8 軸パルス数	第 8 軸パルス数
P[8]	第 9 軸パルス数	第 9 軸パルス数
P[9]	第 10 軸パルス数	第 10 軸パルス数
P[10]	第 11 軸パルス数	第 11 軸パルス数
P[11]	第 12 軸パルス数	第 12 軸パルス数
P[12]	―	第 13 軸パルス数

* 外部軸無しのシステムでは P[6] ― P[11] のデータは「0」を設定してください。7 軸の場合は、P[7] ― P[12] のデータは「0」を設定してください。

コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100（シリアルポート、Ethernet）
参 照	「BscPMovjEx」「BscPMovEx」

■ BscPutUFrame

機 能	指定のユーザ座標データを書き込みます。
書 式	_declspec(dllexport) short APIENTRY BscPutUFrame(short nCid,char *ufname,double *p);
引 数	**IN（引き渡し）** nCid　　通信ハンドラの ID 番号 *ufname　書き込むユーザフレーム名称格納ポインタ *p　　ユーザフレームデータ格納エリアの先頭ポインタ
	OUT（戻り） 無し
	リターン値 0　　：正常終了 その他：エラーコード
解 説	**制約事項** 7 軸以上のロボットには対応していません。
	ユーザフレーム名称 ユーザフレーム番号に対応する以下の名称を指定します。

ユーザ座標名称	指定名称
ユーザ座標 1	UF1
ユーザ座標 2	UF2
ユーザ座標 3	UF3
：	：
：	：
ユーザ座標 21	UF21
ユーザ座標 22	UF22
ユーザ座標 23	UF23
ユーザ座標 24	UF24

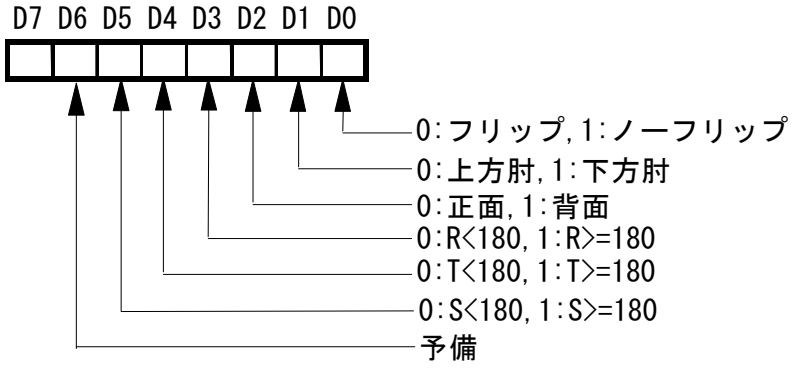
*ユーザフレーム番号の 9～24 番までは NX100、XRC、MRC のみ有効です。

解 説	**変数タイプ** ユーザフレームデータには、ユーザフレーム番号で指定したユーザ座標系の座標値が以下のように割り当てられます。

変数	座標系	意　味
P[0]	ORG	X 座標値（単位 mm、小数点第 3 位有効）
P[1]		Y 座標値（単位 mm、小数点第 3 位有効）
P[2]		Z 座標値（単位 mm、小数点第 3 位有効）
P[3]		手首姿勢 Rx（単位°、小数点第 2 位有効）
P[4]		手首姿勢 Ry（単位°、小数点第 2 位有効）
P[5]		手首姿勢 Rz（単位°、小数点第 2 位有効）
P[6]		形態
P[7]	XX	X 座標値（単位 mm、小数点第 3 位有効）
P[8]		Y 座標値（単位 mm、小数点第 3 位有効）
P[9]		Z 座標値（単位 mm、小数点第 3 位有効）
P[10]		手首姿勢 Rx（単位°、小数点第 2 位有効）
P[11]		手首姿勢 Ry（単位°、小数点第 2 位有効）
P[12]		手首姿勢 Rz（単位°、小数点第 2 位有効）
P[13]		形態
P[14]	XY	X 座標値（単位 mm、小数点第 3 位有効）
P[15]		Y 座標値（単位 mm、小数点第 3 位有効）
P[16]		Z 座標値（単位 mm、小数点第 3 位有効）
P[17]		手首姿勢 Rx（単位°、小数点第 2 位有効）
P[18]		手首姿勢 Ry（単位°、小数点第 2 位有効）
P[19]		手首姿勢 Rz（単位°、小数点第 2 位有効）
P[20]		形態
P[21]	ツール番号	
P[22]	第 7 軸パルス番号（走行軸の場合、単位は mm）	
P[23]	第 8 軸パルス番号（走行軸の場合、単位は mm）	
P[24]	第 9 軸パルス番号（走行軸の場合、単位は mm）	
P[25]	第 10 軸パルス番号	
P[26]	第 11 軸パルス番号	
P[27]	第 12 軸パルス番号	

形態

形態のデータは、以下のビットデータを 10 進数にした値となります。



*ERC、ERC2 の場合は D3 － D7 データは無視されます。
*MRC、MRC2 の場合は D5 － D7 データは無視されます。

コントローラ	NX100、XRC、MRC（シリアルポート、Ethernet）、ERC（シリアルポート）

参 照	「BscGetUFrame」「BscPutUFrameEx2」

■ BscPutUFrameEx2

機 能	指定のユーザ座標データを書き込みます。
書 式	_declspec(dllexport) short APIENTRY BscPutUFrame(short nCid,short ctype,char *ufname,double *p);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号 ctype　コントローラの選択 0：ERC、1：MRC、2：XRC、3；NX100、4:YRC1000、YRC1000micro、DX200、DX100、5：FS100 *ufname　書き込むユーザフレーム名称格納ポインタ *p　ユーザフレームデータ格納エリアの先頭ポインタ
	OUT（戻り） 無し
	リターン値 0　：正常終了 その他：エラーコード
解 説	**制約事項** 7 軸以上のロボットには対応していません。
	ユーザフレーム名称 ユーザフレーム番号に対応する以下の名称を指定します。

ユーザ座標名称	指定名称
ユーザ座標 1	UF1
ユーザ座標 2	UF2
ユーザ座標 3	UF3
:	:
:	:
ユーザ座標 60	UF60
ユーザ座標 61	UF61
ユーザ座標 62	UF62
ユーザ座標 63	UF63

*ユーザフレーム番号の 9～63 番までは YRC1000、YRC1000micro、DX200 DX100 のみ有効です。
*ユーザフレーム番号の 9～16 番までは FS100 のみ有効です。
*ユーザフレーム番号の 9～24 番までは NX100、XRC、MRC のみ有効です。

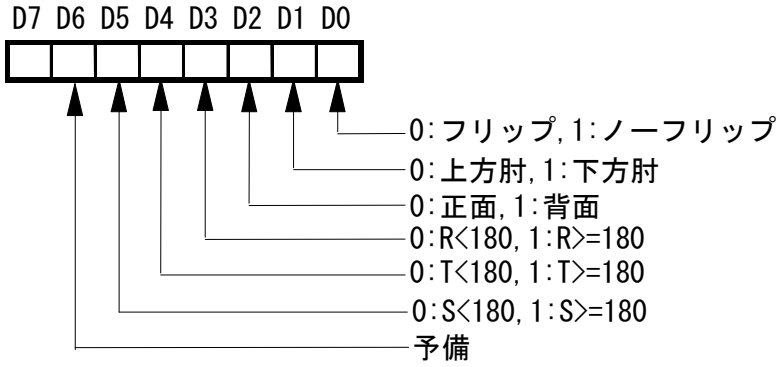
解 説

変数タイプ

ユーザフレームデータには、ユーザフレーム番号で指定したユーザ座標系の座標値が以下のように割り当てられます。

変数	座標系	意　味
P[0]	ORG	X 座標値（単位 mm、小数点第 3 位有効）
P[1]		Y 座標値（単位 mm、小数点第 3 位有効）
P[2]		Z 座標値（単位 mm、小数点第 3 位有効）
P[3]		手首姿勢 Rx（単位°、小数点第 2 位有効）*1
P[4]		手首姿勢 Ry（単位°、小数点第 2 位有効）*1
P[5]		手首姿勢 Rz（単位°、小数点第 2 位有効）*1
P[6]		形態
P[7]	XX	X 座標値（単位 mm、小数点第 3 位有効）
P[8]		Y 座標値（単位 mm、小数点第 3 位有効）
P[9]		Z 座標値（単位 mm、小数点第 3 位有効）
P[10]		手首姿勢 Rx（単位°、小数点第 2 位有効）*1
P[11]		手首姿勢 Ry（単位°、小数点第 2 位有効）*1
P[12]		手首姿勢 Rz（単位°、小数点第 2 位有効）*1
P[13]		形態
P[14]	XY	X 座標値（単位 mm、小数点第 3 位有効）
P[15]		Y 座標値（単位 mm、小数点第 3 位有効）
P[16]		Z 座標値（単位 mm、小数点第 3 位有効）
P[17]		手首姿勢 Rx（単位°、小数点第 2 位有効）*1
P[18]		手首姿勢 Ry（単位°、小数点第 2 位有効）*1
P[19]		手首姿勢 Rz（単位°、小数点第 2 位有効）*1
P[20]		形態
P[21]	ツール番号	
P[22]	第 7 軸パルス番号（走行軸の場合、単位は mm）	
P[23]	第 8 軸パルス番号（走行軸の場合、単位は mm）	
P[24]	第 9 軸パルス番号（走行軸の場合、単位は mm）	
P[25]	第 10 軸パルス番号	
P[26]	第 11 軸パルス番号	
P[27]	第 12 軸パルス番号	

1 YRC1000、YRC1000micro、DX200、DX100、FS100 の場合は「小数点第 4 位有効」となります。

	形態 形態のデータは、以下のビットデータを 10 進数にした値となります。  *ERC、ERC2 の場合は D3 － D7 データは無視されます。 *MRC、MRC2 の場合は D5 － D7 データは無視されます。
コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC（シリアルポート、Ethernet） ERC（シリアルポート）
参 照	「BscGetUFrame」「BscPutUFrame」

■ BscPutVarData

機 能	変数情報を送信します。
書 式	_declspec(dllexport) short APIENTRY BscPutVarData(short nCid,short type,short varno,double *p);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号 type　変数タイプ varno　変数番号 *p　数値変数格納エリアの先頭ポインタ
	OUT（戻り） 無し
	リターン値 0　：正常終了 その他：エラーコード
解 説	**制約事項** 本関数は YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC に対して伝送を行う場合に有効です。
	変数タイプ 変数タイプは、以下のようになります。 0：バイト型変数 1：整数型変数 2：倍精度型変数 3：実数型変数 4：ロボット軸位置型変数 5：ベース軸位置型変数 6：ステーション軸位置型変数（パルス型のみ）

解 説

数値変数格納エリアの内容

数値変数格納エリアの内容、有効配列数は変数タイプによって以下のようになります。

変数ﾀｲﾌﾟ番号	ﾃﾞｰﾀ型(ﾊﾟﾙｽ/直交)	有効配列数	内 容				
			p[0]	p[1]	p[2]	p[3]	p[4]
0	―	1	ﾊﾞｲﾄ値	―	―	―	―
1	―	1	整数値	―	―	―	―
2	―	1	倍精度整数値	―	―	―	―
3	―	1	実数値	―	―	―	―
4	ﾊﾟﾙｽ型	8	0	S 軸ﾊﾟﾙｽ数	L 軸ﾊﾟﾙｽ数	U 軸ﾊﾟﾙｽ数	R 軸ﾊﾟﾙｽ数
4	直交型	10	1	座標種別	X 軸座標値(mm)	Y 軸座標値(mm)	Z 軸座標値(mm)
5	ﾊﾟﾙｽ型	8	0	ﾍﾞｰｽ第 1 軸ﾊﾟﾙｽ数	ﾍﾞｰｽ第 2 軸ﾊﾟﾙｽ数	ﾍﾞｰｽ第 3 軸ﾊﾟﾙｽ数	ﾍﾞｰｽ第 4 軸ﾊﾟﾙｽ数
5	直交型	10	1	座標種別	X 軸座標値(mm)	Y 軸座標値(mm)	Z 軸座標値(mm)
6	ﾊﾟﾙｽ型	8	0	ｽﾃｰｼｮﾝ第 1 ﾊﾟﾙｽ数	ｽﾃｰｼｮﾝ第 2 ﾊﾟﾙｽ数	ｽﾃｰｼｮﾝ第 3 ﾊﾟﾙｽ数	ｽﾃｰｼｮﾝ第 4 ﾊﾟﾙｽ数

変数ﾀｲﾌﾟ番号	ﾃﾞｰﾀ型(ﾊﾟﾙｽ/直交)	有効配列数	内 容				
			p[5]	p[6]	p[7]	p[8]	p[9]
0	―	1	―	―	―	―	―
1	―	1	―	―	―	―	―
2	―	1	―	―	―	―	―
3	―	1	―	―	―	―	―
4	ﾊﾟﾙｽ型	8	B 軸ﾊﾟﾙｽ数	T 軸ﾊﾟﾙｽ数	ﾂｰﾙ番号	―	―
4	直交型	10	手首姿勢 Rx 座標値(deg)	手首姿勢 Ry 座標値(deg)	手首姿勢 Rz 座標値(deg)	形態	ﾂｰﾙ番号
5	ﾊﾟﾙｽ型	8	ﾍﾞｰｽ第 5 軸ﾊﾟﾙｽ数	ﾍﾞｰｽ第 6 軸ﾊﾟﾙｽ数	ﾂｰﾙ番号	―	―
5	直交型	10	手首姿勢 Rx 座標値(deg)	手首姿勢 Ry 座標値(deg)	手首姿勢 Rz 座標値(deg)	形態	ﾂｰﾙ番号
6	ﾊﾟﾙｽ型	8	ｽﾃｰｼｮﾝ第 5 ﾊﾟﾙｽ数	ｽﾃｰｼｮﾝ第 6 ﾊﾟﾙｽ数	ﾂｰﾙ番号	―	―

ロボット軸位置、ベース軸位置型変数の場合は、第 1 番目のリターン値によって変数の型がパルス型か直交型かに分かれます。(ステーション軸位置型変数の場合はパルス型のみです。)

座標種別、形態については以下を参照してください。

座標種別

YRC1000、YRC1000micro、DX200、DX100

座標種別のデータは、以下の座標名称に相当します。

座標種別	座標名称	座標種別	座標名称
0	ベース座標	:	:
1	ロボット座標	:	:
2	ユーザ座標 1	63	ユーザ座標 62
3	ユーザ座標 2	64	ユーザ座標 63
:	:	65	ツール座標
:	:	66	マスターツール座標

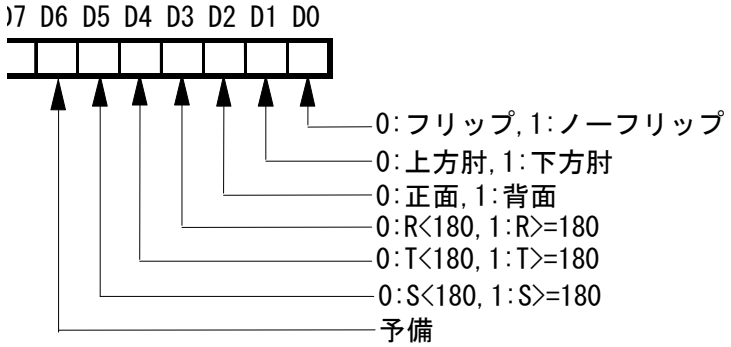
解 説	**座標種別** **FS100** 座標種別のデータは、以下の座標名称に相当します。

座標種別	座標名称	座標種別	座標名称
0	ベース座標	10	ユーザ座標 9
1	ロボット座標	11	ユーザ座標 10
2	ユーザ座標 1	12	ユーザ座標 11
3	ユーザ座標 2	13	ユーザ座標 12
4	ユーザ座標 3	14	ユーザ座標 13
5	ユーザ座標 4	15	ユーザ座標 14
6	ユーザ座標 5	16	ユーザ座標 15
7	ユーザ座標 6	17	ユーザ座標 16
8	ユーザ座標 7	18	ツール座標
9	ユーザ座標 8	19	マスターツール座標

座標種別
NX100、XRC、MRC
座標種別のデータは、以下の座標名称に相当します。

座標種別	座標名称	座標種別	座標名称
0	ベース座標	14	ユーザ座標 13
1	ロボット座標	15	ユーザ座標 14
2	ユーザ座標 1	16	ユーザ座標 15
3	ユーザ座標 2	17	ユーザ座標 16
4	ユーザ座標 3	18	ユーザ座標 17
5	ユーザ座標 4	19	ユーザ座標 18
6	ユーザ座標 5	20	ユーザ座標 19
7	ユーザ座標 6	21	ユーザ座標 20
8	ユーザ座標 7	22	ユーザ座標 21
9	ユーザ座標 8	23	ユーザ座標 22
10	ユーザ座標 9	24	ユーザ座標 23
11	ユーザ座標 10	25	ユーザ座標 24
12	ユーザ座標 11	26	ツール座標
13	ユーザ座標 12	27	マスターツール座標

形態
形態のデータは、以下のビットデータを１０進数にした値となります。



*MRC、MRC2 の場合は D5-D7 データは無視されます。

コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC（シリアルポート、Ethernet）

参 照	「BscGetVarData」「BscGetVarData2」

■ BscPutVarData2

機 能	変数情報を送信します。（７軸以上対応）
書 式	_declspec(dllexport) short APIENTRY BscPutVarData2(short nCid,short type,short varno,double *p,short axisNum);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号 type　変数タイプ varno　変数番号 *p　数値変数格納エリアの先頭ポインタ axisNum　軸数
	OUT（戻り） 無し
	リターン値 0　：正常終了 その他：エラーコード
解 説	**制約事項** 本関数は YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC に対して伝送を行う場合に有効です。
	変数タイプ 変数タイプは、以下のようになります。 0：バイト型変数 1：整数型変数 2：倍精度型変数 3：実数型変数 4：ロボット軸位置型変数 5：ベース軸位置型変数 6：ステーション軸位置型変数（パルス型のみ）

解 説

数値変数格納エリアの内容

数値変数格納エリアの内容、有効配列数は変数タイプによって以下のようになります。

変数ﾀｲﾌﾟ番号	ﾃﾞｰﾀ型(ﾊﾟﾙｽ/直交)	有効配列数	内 容				
			p[0]	p[1]	p[2]	p[3]	p[4]
0	―	1	ﾊﾞｲﾄ値	―	―	―	―
1	―	1	整数値	―	―	―	―
2	―	1	倍精度整数値	―	―	―	―
3	―	1	実数値	―	―	―	―
4	ﾊﾟﾙｽ型	10	0	S 軸ﾊﾟﾙｽ数	L 軸ﾊﾟﾙｽ数	U 軸ﾊﾟﾙｽ数	R 軸ﾊﾟﾙｽ数
4	直交型	10	1	座標種別	X 軸座標値(mm)	Y 軸座標値(mm)	Z 軸座標値(mm)
5	ﾊﾟﾙｽ型	8	0	ﾍﾞｰｽ第 1 軸ﾊﾟﾙｽ数	ﾍﾞｰｽ第 2 軸ﾊﾟﾙｽ数	ﾍﾞｰｽ第 3 軸ﾊﾟﾙｽ数	ﾍﾞｰｽ第 4 軸ﾊﾟﾙｽ数
5	直交型	10	1	座標種別	X 軸座標値(mm)	Y 軸座標値(mm)	Z 軸座標値(mm)
6	ﾊﾟﾙｽ型	8	0	ｽﾃｰｼｮﾝ第 1 ﾊﾟﾙｽ数	ｽﾃｰｼｮﾝ第 2 ﾊﾟﾙｽ数	ｽﾃｰｼｮﾝ第 3 ﾊﾟﾙｽ数	ｽﾃｰｼｮﾝ第 4 ﾊﾟﾙｽ数

変数ﾀｲﾌﾟ番号	ﾃﾞｰﾀ型(ﾊﾟﾙｽ/直交)	有効配列数	内 容				
			p[5]	p[6]	p[7]	p[8]	p[9]
0	―	1	―	―	―	―	―
1	―	1	―	―	―	―	―
2	―	1	―	―	―	―	―
3	―	1	―	―	―	―	―
4	ﾊﾟﾙｽ型	10	B 軸ﾊﾟﾙｽ数	T 軸ﾊﾟﾙｽ数	7 軸ﾊﾟﾙｽ数	8 軸ﾊﾟﾙｽ数	ﾂｰﾙ番号
4	直交型	10	手首姿勢 Rx 座標値(deg)	手首姿勢 Ry 座標値(deg)	手首姿勢 Rz 座標値(deg)	形態	ﾂｰﾙ番号
5	ﾊﾟﾙｽ型	8	ﾍﾞｰｽ第 5 軸ﾊﾟﾙｽ数	ﾍﾞｰｽ第 6 軸ﾊﾟﾙｽ数	ﾂｰﾙ番号	―	―
5	直交型	10	手首姿勢 Rx 座標値(deg)	手首姿勢 Ry 座標値(deg)	手首姿勢 Rz 座標値(deg)	形態	ﾂｰﾙ番号
6	ﾊﾟﾙｽ型	8	ｽﾃｰｼｮﾝ第 5 ﾊﾟﾙｽ数	ｽﾃｰｼｮﾝ第 6 ﾊﾟﾙｽ数	ﾂｰﾙ番号	―	―

ロボット軸位置、ベース軸位置型変数の場合は、第 1 番目のリターン値によって変数の型がパルス型か直交型かに分かれます。(ステーション軸位置型変数の場合はパルス型のみです。)

座標種別、形態については以下を参照してください。

座標種別

YRC1000、YRC1000micro、DX200、DX100

座標種別のデータは、以下の座標名称に相当します。

座標種別	座標名称	座標種別	座標名称
0	ベース座標	:	:
1	ロボット座標	:	:
2	ユーザ座標 1	63	ユーザ座標 62
3	ユーザ座標 2	64	ユーザ座標 63
:	:	65	ツール座標
:	:	66	マスターツール座標

解 説

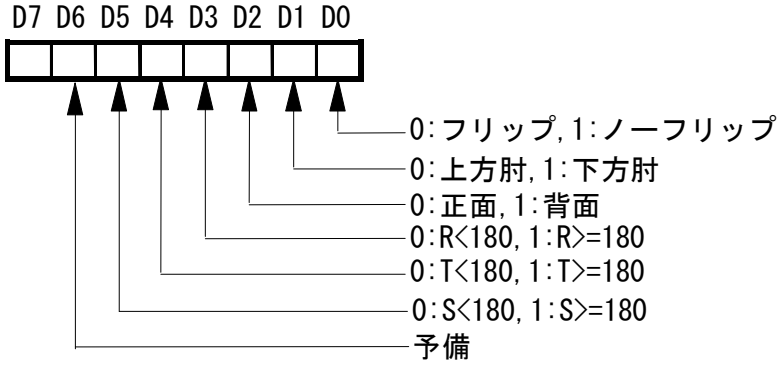
座標種別
FS100
座標種別のデータは、以下の座標名称に相当します。

座標種別	座標名称	座標種別	座標名称
0	ベース座標	10	ユーザ座標 9
1	ロボット座標	11	ユーザ座標 10
2	ユーザ座標 1	12	ユーザ座標 11
3	ユーザ座標 2	13	ユーザ座標 12
4	ユーザ座標 3	14	ユーザ座標 13
5	ユーザ座標 4	15	ユーザ座標 14
6	ユーザ座標 5	16	ユーザ座標 15
7	ユーザ座標 6	17	ユーザ座標 16
8	ユーザ座標 7	18	ツール座標
9	ユーザ座標 8	19	マスターツール座標

座標種別
NX100、XRC
座標種別のデータは、以下の座標名称に相当します。

座標種別	座標名称	座標種別	座標名称
0	ベース座標	14	ユーザ座標 13
1	ロボット座標	15	ユーザ座標 14
2	ユーザ座標 1	16	ユーザ座標 15
3	ユーザ座標 2	17	ユーザ座標 16
4	ユーザ座標 3	18	ユーザ座標 17
5	ユーザ座標 4	19	ユーザ座標 18
6	ユーザ座標 5	20	ユーザ座標 19
7	ユーザ座標 6	21	ユーザ座標 20
8	ユーザ座標 7	22	ユーザ座標 21
9	ユーザ座標 8	23	ユーザ座標 22
10	ユーザ座標 9	24	ユーザ座標 23
11	ユーザ座標 10	25	ユーザ座標 24
12	ユーザ座標 11	26	ツール座標
13	ユーザ座標 12	27	マスターツール座標

形態
形態のデータは、以下のビットデータを 10 進数にした値となります。



コントローラ

YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC（シリアルポート、Ethernet）

参 照	「BscGetVarData」「BscGetVarData2」

■ BscStartJob

機 能	始動をかけます。(始動をかけるジョブは最後に選択されたジョブ名)
書 式	_declspec(dllexport) short APIENTRY BscStartJob(short nCid);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号
	OUT（戻り） 無し
	リターン値 0　　：正常終了 1　　：カレントジョブ名称指定無し その他：エラーコード
解 説	**呼び出し条件** 本関数を実行する事前に BscSelectJob 関数をコールして、カレントジョブ名の指定をしておく必要があります。 BscHoldOn 関数で始動中のジョブがホールドされ、これを再スタートさせたい場合は、BscHoldOff 関数でホールドを解除し、BscContinueJob 関数をコールして、再スタートをかけてください。
コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC（シリアルポート、Ethernet） ERC（シリアルポート）
参 照	「BscContinueJob」「BscSelectJob」

■ BscSelectJob

機 能	ジョブを選択します。
書 式	_declspec(dllexport) short APIENTRY BscSelectJob(short nCid,char *name);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号 *name　ジョブ名称格納ポインタ
	OUT（戻り） 無し
	リターン値 0　：正常終了 その他：エラーコード
解 説	**ジョブ名称** ジョブ名称文字数は以下の制限があります。 Max.30 のキャラクタの文字列。（MS-DOS で使用できる文字 8 文字） また、全ジョブを選択する場合は '*" をジョブ名称としてください。
コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC（シリアルポート、Ethernet） ERC（シリアルポート）
参 照	「BscFindFirstMaster」「BscDeleteJob」「BscSetMasterJob」 「BscSetLineNumber」「BscStartJob」

■ BscSelectMode

機 能	モードを選択します。
書 式	_declspec(dllexport) short APIENTRY BscSelectMode(short nCid,short mode);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号 mode　選択モード
	OUT（戻り） 無し
	リターン値 0　　：正常終了 その他：エラーコード
解 説	**選択モード** 選択モードは、以下のようになります。 1：ティーチ 2：プレイ
コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC（シリアルポート、Ethernet） ERC（シリアルポート）

■ BscSelLoopCycle

機 能	サイクルモードを連続自動モードに変更します。
書 式	_declspec(dllexport) short APIENTRY BscSelLoopCycle(short nCid);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号
	OUT（戻り） 無し
	リターン値 0　　：正常終了 その他：エラーコード
コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC（シリアルポート、Ethernet） ERC（シリアルポート）
参 照	「BscSelStepCycle」「BscSelOneCycle」

■ BscSelOneCycle

機 能	サイクルモードを 1 サイクルモードに変更します。
書 式	_declspec(dllexport) short APIENTRY BscSelOneCycle(short nCid);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号
	OUT（戻り） 無し
	リターン値 0　　：正常終了 その他：エラーコード
コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC（シリアルポート、Ethernet） ERC（シリアルポート）
参 照	「BscSelStepCycle」「BscSelLoopCycle」

■ BscSelStepCycle

機 能	サイクルモードをステップモードに変更します。
書 式	_declspec(dllexport) short APIENTRY BscSelStepCycle(short nCid);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号
	OUT（戻り） 無し
	リターン値 0　　：正常終了 その他：エラーコード
コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC（シリアルポート、Ethernet） ERC（シリアルポート）
参 照	「BscSelOneCycle」「BscSelLoopCycle」

■ BscSetLineNumber

機 能	ライン番号の設定を行います。
書 式	_declspec(dllexport) short APIENTRY BscSetLineNumber(short nCid,short line);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号 line　ライン番号
	OUT（戻り） 無し
	リターン値 0　　：正常終了 その他：エラーコード
コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC（シリアルポート、Ethernet） ERC（シリアルポート）
参 照	「BscSelectJob」

■ BscSetMasterJob

機 能	ジョブをマスタージョブ登録します。
書 式	_declspec(dllexport) short APIENTRY BscSetMasterJob(short nCid);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号
	OUT（戻り） 無し
	リターン値 0　　：正常終了 その他：エラーコード
解 説	**呼び出し条件** 本関数を実行する事前に BscSelectJob 関数をコールして、登録ジョブ名の指定をしておく必要があります。
コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC（シリアルポート、Ethernet） ERC（シリアルポート）
参 照	「BscDeleteJob」「BscSelectJob」

■ BscReset

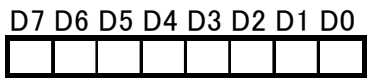
機 能	ロボットのアラームのリセットを行います。
書 式	_declspec(dllexport) short APIENTRY BscReset(short nCid);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号
	OUT（戻り） 無し
	リターン値 0　　：正常終了 その他：エラーコード
コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC（シリアルポート、Ethernet） ERC（シリアルポート）
参 照	BscCancel

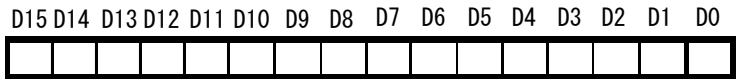
■ BscSetCtrlGroup

機 能	制御グループの設定を行います。
書 式	_declspec(dllexport) short APIENTRY BscSetCtrlGroup(short nCid,short groupno);
引 数	**IN（引き渡し）** nCid　　通信ハンドラの ID 番号 groupno　設定する制御グループ情報
	OUT（戻り） 無し
	リターン値 0　　：正常終了 その他：エラーコード
解 説	**制約事項** 本関数は MRC に対して伝送を行う場合に有効です。 YRC1000、YRC1000micro、DX200、DX100 に対して伝送を行う場合は、BscSetCtrlGroupDX を参照してください。 FS100、NX100、XRC に対して伝送を行う場合は、BscSetCtrlGroupXrc を参照してください。 コントローラの電源立ち上げ時には、ロボット 1、及びベース 1、ステーション 1（ベース、ステーションが存在する場合）が指定されています。ベース軸（走行軸など）を有するシステムでは、そのベース軸を有するマニピュレータが指定された場合、自動的にそのベース軸も指定されます。 ただしこの関数では以下の設定はできません。 ・存在しない制御軸の指定（選択） ・複数のロボット制御グループの指定（選択） ・複数のステーション制御グループの指定（選択）
	制御グループ情報 制御グループ情報は、以下のビットデータを 10 進数にした値となります。 D7 D6 D5 D4 D3 D2 D1 D0  D0：R1（ロボット1） D1：R2（ロボット2） D2：S1（ステーション1） D3：S2（ステーション2） D4：S3（ステーション3） D5：S4（ステーション4） D6：S5（ステーション5） D7：S6（ステーション6）
コントローラ	MRC（シリアルポート、Ethernet）

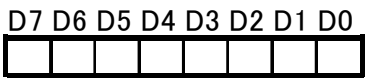
参 照	「BscGetCtrlGroupDX」 「BscSetCtrlGroupDX」 「BscIsCtrlGroupDX」 「BscSetCtrlGroupXrc」 「BscGetCtrlGroupXrc」 「BscIsCtrlGroupXrc」 「BscIsTaskInfXrc」 「BscGetCtrlGroup」 「BscIsCtrlGroup」 「BscIsTaskInf」 「BscChangeTask」

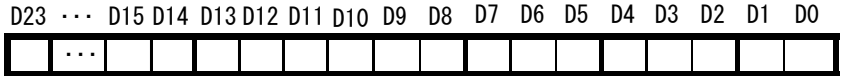
■ BscSetCtrlGroupXrc

機 能	制御グループの設定を行います。
書 式	_declspec(dllexport) short APIENTRY BscSetCtrlGroupXrc(short nCid,short groupno1,short groupno2);
引 数	**IN（引き渡し）** nCid　　通信ハンドラの ID 番号 groupno1　設定する制御グループ情報（ロボット軸） groupno2　設定する制御グループ情報（ステーション軸）
	OUT（戻り） groupno1　設定する制御グループ情報（ロボット軸） groupno2　設定する制御グループ情報（ステーション軸）
	リターン値 0　　：正常終了 その他：エラーコード
解 説	**制約事項** 本関数は FS100、NX100、XRC に対して伝送を行う場合に有効です。 YRC1000、YRC1000micro、DX200、DX100、に対しては伝送を行う場合は、BscSetCtrlGroupDX を参照してください。 MRC に対しては伝送を行う場合は、BscSetCtrlGroup を参照してください。 コントローラの電源立ち上げ時には、ロボット 1、及びベース 1、ステーション 1（ベース、ステーションが存在する場合）が指定されています。ベース軸（走行軸など）を有するシステムでは、そのベース軸を有するマニピュレータが指定された場合、自動的にそのベース軸も指定されます。ただしこの関数では以下の設定はできません。 ・存在しない制御軸の指定（選択） ・複数のロボット制御グループの指定（選択） ・複数のステーション制御グループの指定（選択）
	制御グループ情報（ロボット軸） 制御グループ情報は、以下のビットデータを 10 進数にした値となります。  D0：R1（ロボット1） D1：R2（ロボット2） D2：R3（ロボット3） D3：R4（ロボット4）

解 説	**制御グループ情報（ステーション軸）** 制御グループ情報は、以下のビットデータを 10 進数にした値となります。  D0：S1（ステーション1） D1：S2（ステーション2） D2：S3（ステーション3） D3：S4（ステーション4） D4：S5（ステーション5） D5：S6（ステーション6） D6：S7（ステーション7） D7：S8（ステーション8） D8：S9（ステーション9） D9：S10（ステーション10） D10：S11（ステーション11） D11：S12（ステーション12）
コントローラ	FS100、NX100、XRC（シリアルポート、Ethernet）
参 照	「BscGetCtrlGroupDX」 「BscSetCtrlGroupDX」 「BscIsCtrlGroupDX」 「BscGetCtrlGroupXrc」 「BscIsCtrlGroupXrc」 「BscIsTaskInfXrc」 「BscSetCtrlGroup」 「BscGetCtrlGroup」 「BscIsCtrlGroup」 「BscIsTaskInf」 「BscChangeTask」

■ BscSetCtrlGroupDX

機 能	制御グループの設定を行います。
書 式	_declspec(dllexport) short APIENTRY BscSetCtrlGroupDX(short nCid,long groupno1,long groupno2);
引 数	**IN（引き渡し）** nCid　　通信ハンドラの ID 番号 groupno1　設定する制御グループ情報（ロボット軸） groupno2　設定する制御グループ情報（ステーション軸）
	OUT（戻り） groupno1　設定する制御グループ情報（ロボット軸） groupno2　設定する制御グループ情報（ステーション軸）
	リターン値 0　　：正常終了 その他：エラーコード
解 説	**制約事項** 本関数は YRC1000、YRC1000micro、DX200、DX100、に対して伝送を行う場合に有効です。 FS100、NX100、XRC に対しては伝送を行う場合は、BscSetCtrlGroupXrc を参照してください。 MRC に対しては伝送を行う場合は、BscSetCtrlGroup を参照してください。 コントローラの電源立ち上げ時には、ロボット 1、及びベース 1、ステーション 1（ベース、ステーションが存在する場合）が指定されています。 ベース軸（走行軸など）を有するシステムでは、そのベース軸を有するマニピュレータが指定された場合、自動的にそのベース軸も指定されます。 ただしこの関数では以下の設定はできません。 ・存在しない制御軸の指定（選択） ・複数のロボット制御グループの指定（選択） ・複数のステーション制御グループの指定（選択）
	制御グループ情報（ロボット軸） 制御グループ情報は、以下のビットデータを 10 進数にした値となります。 D7 D6 D5 D4 D3 D2 D1 D0  D0：R1（ロボット1） D1：R2（ロボット2） D2：R3（ロボット3） D3：R4（ロボット4） ： ： D7：R8（ロボット8）

解 説	**制御グループ情報（ステーション軸）** 制御グループ情報は、以下のビットデータを 10 進数にした値となります。  D0：S1（ステーション1） D1：S2（ステーション2） D2：S3（ステーション3） D3：S4（ステーション4） D4：S5（ステーション5） D5：S6（ステーション6） D6：S7（ステーション7） D7：S8（ステーション8） D8：S9（ステーション9） D9：S10（ステーション10） D10：S11（ステーション11） D11：S12（ステーション12） ： ： D23：S24（ステーション 24）
コントローラ	YRC1000、YRC1000micro、DX200、DX100、（シリアルポート、Ethernet）
参 照	「BscIsCtrlGroupDX」 「BscIsCtrlGroupDX」 「BscSetCtrlGroupXrc」 「BscGetCtrlGroupXrc」 「BscIsCtrlGroupXrc」 「BscIsTaskInfXrc」 「BscSetCtrlGroup」 「BscGetCtrlGroup」 「BscIsCtrlGroup」 「BscIsTaskInf」 「BscChangeTask」

■ BscServoOff

機 能	サーボ電源オフを送信します。
書 式	_declspec(dllexport) short APIENTRY BscServoOff(short nCid);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号
	OUT（戻り） 無し
	リターン値 0　　：正常終了 その他：エラーコード
コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC（シリアルポート、Ethernet） ERC（シリアルポート）
参 照	「BscServoOn」「BscIsServo」

■ BscServoOn

機 能	サーボ電源オンを送信します。
書 式	_declspec(dllexport) short APIENTRY BscServoOn(short nCid);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号
	OUT（戻り） 無し
	リターン値 0　　：正常終了 その他：エラーコード
コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC（シリアルポート、Ethernet） ERC（シリアルポート）
参 照	「BscServoOff」「BscIsServo」

7.4 DCI 機能

ロボット側がプレイバックしている時にそのインストラクションに対応した関数を準備することによって、ジョブのセーブ・ロードあるいは変数のロード・セーブを自動的に行います。
ロボットのステータス（現在位置、アラーム・エラー・サーボ等の状態）のリードや、システムのコントロール（スタート、ホールド、ジョブの呼び出し等）を行います。
以下の関数が使用できます。

BscDCILoadSave
BscDCILoadSaveOnce
BscDCIGetPos
BscDCIGetPos2
BscDCIGetVarData
BscDCIGetVarDataEx
BscDCIPutPos
BscDCIPutPos2
BscDCIPutVarData
BscDCIPutVarDataEx

■ BscDCILoadSave

機 能	DCI インストラクションによるジョブの送受信を行います。
書 式	_declspec(dllexport) short APIENTRY BscDCILoadSave(short nCid,short timec);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号 timec　送受信待ち時間（秒）
	OUT（戻り） 無し
	リターン値 -1　　：送受信失敗 その他：受信ジョブ数
解 説	**送受信回数** 本関数は指定の待ち時間以内にロボットから送受信信号が入ってくる間、何度も通信を行います。
コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC（シリアルポート、Ethernet） ERC（シリアルポート）
参 照	「BscDCILoadSaveOnce」「BscDCIGetPos」「BscDCIPutPos」「BscDCIGetPos2」「BscDCIPutPos2」「BscDCIGetVarData」「BscDCIPutVarData」

■ BscDCILoadSaveOnce

機 能	DCI インストラクションによるジョブの送受信を行います。
書 式	_declspec(dllexport) short APIENTRY BscDCILoadSaveOnce(short nCid);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号
	OUT（戻り） 無し
	リターン値 -1　　：送受信失敗 その他：受信ジョブ数
解 説	**送受信回数** 本関数はロボットから送受信信号が入ってくるまで、無限に待ちます。信号が来れば通信を 1 回のみ行います。
コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC（シリアルポート、Ethernet） ERC（シリアルポート）
参 照	「BscDCILoadSave」「BscDCIGetPos」「BscDCIPutPos」「BscDCIGetPos2」「BscDCIPutPos2」「BscDCIGetVarData」「BscDCIPutVarData」

■ BscDCIGetPos

機 能	DCI インストラクションによる変数の受信を行います。
書 式	_declspec(dllexport) short APIENTRY BscDCIGetPos(short nCid,short *type,short *rconf,double *p);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号 *type　変数タイプ番号ポインタ *rconf　形態データ格納ポインタ *p　数値変数格納エリアの先頭ポインタ
	OUT（戻り） *rconf　形態データ格納ポインタ *p　数値変数格納エリアの先頭ポインタ
	リターン値 -1　：受信失敗 その他：変数タイプ番号
解 説	**変数タイプ番号** 変数タイプ番号の意味は以下のようになります。

コントローラ / タイプ番号	YRC1000/YRC1000micro/DX200/DX100 FS100/NX100/XRC/MRC	ERC
1	バイト型変数	
2	整数型変数	
3	倍精度型変数	
4	実数型変数	
5	ロボット軸位置型変数（パルス型）	
6	ロボット軸位置型変数（直交型）	
7	ベース軸位置型変数（パルス型）	
8	ベース軸位置型変数（直交型）	
9	ステーション軸位置型変数（パルス型）	ー

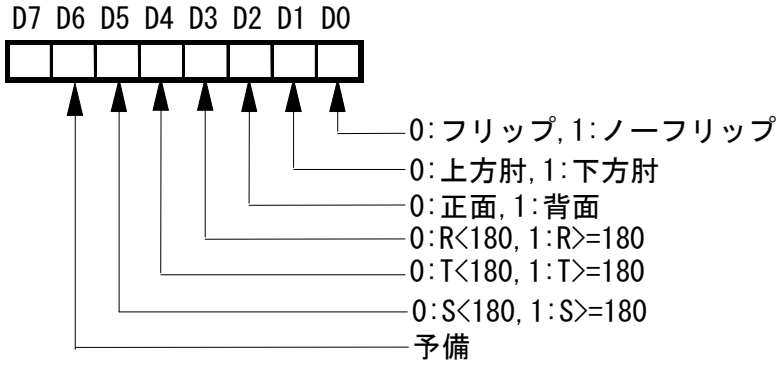
変数タイプ番号と使用する格納エリア、配列数

タイプ番号	使用する格納エリア	必要な配列数
1～9	p	6

解 説

形態

形態のデータは、以下のビットデータを 10 進数にした値となります。



*ERC、ERC2 の場合は D3 － D7 データは無視されます。
*MRC、MRC2 の場合は D5 － 7D データは無視されます。

数値変数格納エリアの内容

数値変数格納エリアの内容、有効配列数は変数タイプによって以下のようになります。

変数ﾀｲﾌﾟ番号	有効配列数	内 容					
		p[0]	p[1]	p[2]	p[3]	p[4]	p[5]
1	1	ﾊﾞｲﾄ値	－	－	－	－	－
2	1	整数値	－	－	－	－	－
3	1	倍精度整数値	－	－	－	－	－
4	1	実数値	－	－	－	－	－
5	6	S 軸ﾊﾟﾙｽ数	L 軸ﾊﾟﾙｽ数	U 軸ﾊﾟﾙｽ数	R 軸ﾊﾟﾙｽ数	B 軸ﾊﾟﾙｽ数	T 軸ﾊﾟﾙｽ数
6	6	X 軸座標値(mm)	Y 軸座標値(mm)	Z 軸座標値(mm)	手首姿勢 Rx 座標値(deg)	手首姿勢 Ry 座標値(deg)	手首姿勢 Rz 座標値(deg)
7	6	ﾍﾞｰｽ第 1 軸ﾊﾟﾙｽ数	ﾍﾞｰｽ第 2 軸ﾊﾟﾙｽ数	ﾍﾞｰｽ第 3 軸ﾊﾟﾙｽ数	ﾍﾞｰｽ第 4 軸ﾊﾟﾙｽ数	ﾍﾞｰｽ第 5 軸ﾊﾟﾙｽ数	ﾍﾞｰｽ第 6 軸ﾊﾟﾙｽ数
8	6	ﾍﾞｰｽ第 1 軸直交値(mm)	ﾍﾞｰｽ第 2 軸直交値(mm)	ﾍﾞｰｽ第 3 軸直交値(mm)	ﾍﾞｰｽ第 4 軸直交値(mm)	ﾍﾞｰｽ第 5 軸直交値(mm)	ﾍﾞｰｽ第 6 軸直交値(mm)
9	6	ｽﾃｰｼｮﾝ第 1 ﾊﾟﾙｽ数	ｽﾃｰｼｮﾝ第 2 ﾊﾟﾙｽ数	ｽﾃｰｼｮﾝ第 3 ﾊﾟﾙｽ数	ｽﾃｰｼｮﾝ第 4 ﾊﾟﾙｽ数	ｽﾃｰｼｮﾝ第 5 ﾊﾟﾙｽ数	ｽﾃｰｼｮﾝ第 6 ﾊﾟﾙｽ数

コントローラ

YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC（シリアルポート、Ethernet）
ERC（シリアルポート）

参 照

「BscDCILoadSave」 「BscDCILoadSaveOnce」 「BscDCIPutPos」 「BscDCIPutPos2」 「BscDCIPutVarData」

■ BscDCIGetPos2

機 能	DCI インストラクションによる変数の受信を行います。(7 軸以上対応)
書 式	_declspec(dllexport) short APIENTRY BscDCIGetPos2(short nCid,short *type,short *rconf,double *p,short *axisNum);
引 数	**IN（引き渡し）** nCid 通信ハンドラの ID 番号 *type 変数タイプ番号ポインタ *rconf 形態データ格納ポインタ *p 数値変数格納エリアの先頭ポインタ *axisNum 軸数格納エリアの先頭ポインタ
	OUT（戻り） *rconf 形態データ格納ポインタ *p 数値変数格納エリアの先頭ポインタ
	リターン値 -1 ：受信失敗 その他：変数タイプ番号
解 説	**変数タイプ番号** 変数タイプ番号の意味は以下のようになります。

コントローラ / タイプ番号	YRC1000/YRC1000micro/DX200/DX100 FS100/NX100/XRC/MRC	ERC
1	バイト型変数	
2	整数型変数	
3	倍精度型変数	
4	実数型変数	
5	ロボット軸位置型変数（パルス型）	
6	ロボット軸位置型変数（直交型）	
7	ベース軸位置型変数（パルス型）	
8	ベース軸位置型変数（直交型）	
9	ステーション軸位置型変数（パルス型）	ー

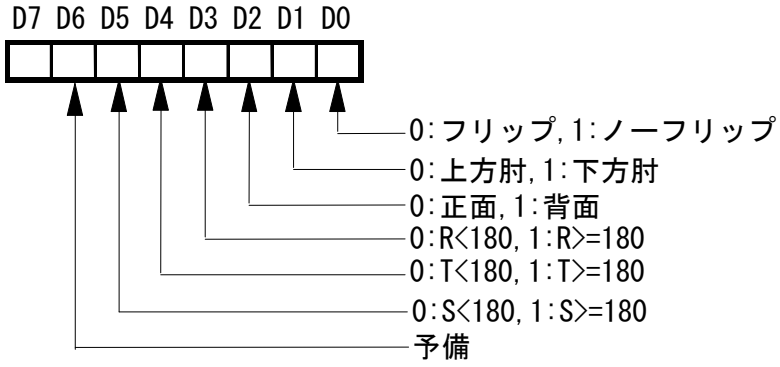
変数タイプ番号と使用する格納エリア、配列数

タイプ番号	使用する格納エリア	必要な配列数
1～9	p	8

解 説

形態

形態のデータは、以下のビットデータを 10 進数にした値となります。



*ERC、ERC2 の場合は D3 － D7 データは無視されます。
*MRC、MRC2 の場合は D5 － D7 データは無視されます。

数値変数格納エリアの内容

数値変数格納エリアの内容、有効配列数は変数タイプによって以下のようになります。

変数ﾀｲﾌﾟ番号	有効配列数	内 容					
		p[0]	p[1]	p[2]	p[3]	p[4]	p[5]
1	1	ﾊﾞｲﾄ値	－	－	－	－	－
2	1	整数値	－	－	－	－	－
3	1	倍精度整数値	－	－	－	－	－
4	1	実数値	－	－	－	－	－
5	8	S 軸ﾊﾟﾙｽ数	L 軸ﾊﾟﾙｽ数	U 軸ﾊﾟﾙｽ数	R 軸ﾊﾟﾙｽ数	B 軸ﾊﾟﾙｽ数	T 軸ﾊﾟﾙｽ数
6	6	X 軸座標値(mm)	Y 軸座標値(mm)	Z 軸座標値(mm)	手首姿勢 Rx 座標値(deg)	手首姿勢 Ry 座標値(deg)	手首姿勢 Rz 座標値(deg)
7	6	ﾍﾞｰｽ第 1 軸ﾊﾟﾙｽ数	ﾍﾞｰｽ第 2 軸ﾊﾟﾙｽ数	ﾍﾞｰｽ第 3 軸ﾊﾟﾙｽ数	ﾍﾞｰｽ第 4 軸ﾊﾟﾙｽ数	ﾍﾞｰｽ第 5 軸ﾊﾟﾙｽ数	ﾍﾞｰｽ第 6 軸ﾊﾟﾙｽ数
8	6	ﾍﾞｰｽ第 1 軸直交値(mm)	ﾍﾞｰｽ第 2 軸直交値(mm)	ﾍﾞｰｽ第 3 軸直交値(mm)	ﾍﾞｰｽ第 4 軸直交値(mm)	ﾍﾞｰｽ第 5 軸直交値(mm)	ﾍﾞｰｽ第 6 軸直交値(mm)
9	6	ｽﾃｰｼｮﾝ第 1 ﾊﾟﾙｽ数	ｽﾃｰｼｮﾝ第 2 ﾊﾟﾙｽ数	ｽﾃｰｼｮﾝ第 3 ﾊﾟﾙｽ数	ｽﾃｰｼｮﾝ第 4 ﾊﾟﾙｽ数	ｽﾃｰｼｮﾝ第 5 ﾊﾟﾙｽ数	ｽﾃｰｼｮﾝ第 6 ﾊﾟﾙｽ数

変数ﾀｲﾌﾟ番号	有効配列数	内 容	
		p[6]	p[7]
1	1	－	－
2	1	－	－
3	1	－	－
4	1	－	－
5	8	7 軸ﾊﾟﾙｽ数	8 軸ﾊﾟﾙｽ数
6	6	－	－
7	6	－	－
8	6	－	－
9	6	－	－

コントローラ

YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC（シリアルポート、Ethernet）
ERC（シリアルポート）

参 照

「BscDCILoadSave」 「BscDCILoadSaveOnce」 「BscDCIPutPos」 「BscDCIPutPos2」 「BscDCIPutVarData」

■ BscDCIGetVarData

機 能	DCI インストラクションによる変数の受信を行います。
書 式	_declspec(dllexport) short APIENTRY BscDCIGetVarData(short nCid,short *type,short *rconf,double *p, char *str);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号 *type　変数タイプ番号ポインタ *rconf　形態データ格納ポインタ *p　数値変数格納エリアの先頭ポインタ *str　文字変数格納エリアの先頭ポインタ
	OUT（戻り） *rconf　形態データ格納ポインタ *p　数値変数格納エリアの先頭ポインタ *str　文字変数格納エリアの先頭ポインタ
	リターン値 -1　：受信失敗 その他：変数タイプ番号
解 説	**制約事項** 本関数はコントローラが YRC1000 及び YRC1000micro、DX200、DX100、FS100、NX100 Ver3.0 以降の場合に限り文字型変数を取り扱うことが可能です。
	変数タイプ番号 変数タイプ番号の意味は以下のようになります。

コントローラ / タイプ番号	YRC1000/YRC1000micro/DX200/ DX100/FS100/NX100 V3.0 以降	NX100(V3.0 未満)/ XRC/MRC	ERC
1	バイト型変数		
2	整数型変数		
3	倍精度型変数		
4	実数型変数		
5	ロボット軸位置型変数（パルス型）		
6	ロボット軸位置型変数（直交型）		
7	ベース軸位置型変数（パルス型）		
8	ベース軸位置型変数（直交型）		
9	ステーション軸位置型変数（パルス型）		－
10	文字型変数	－	－

重要　文字型変数は YRC1000 及び YRC1000micro、DX200、DX100、FS100、NX100 Ver3.0 以降のバージョンで有効です。

解 説

変数タイプ番号と使用する格納エリア、配列数

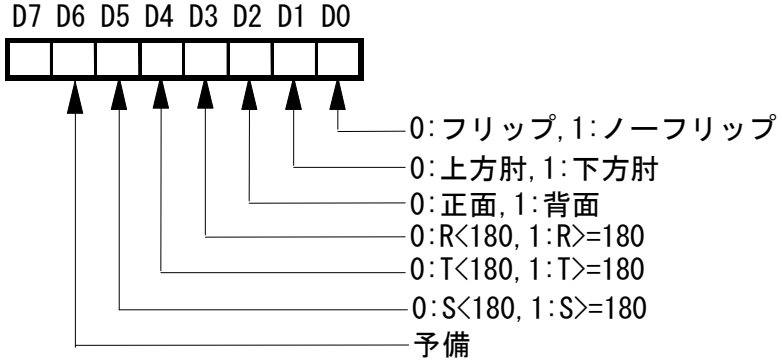
タイプ番号	使用する格納エリア	必要な配列数
1～9	p	6
10	str	16

重要

文字型変数の送受信で本関数をコールする場合、変数格納エリアは以下の例のように 17 文字分の変数エリアを確保してください。
Visual Basic で使用する場合の変数宣言例：
Dim S_Variable As String *17
Visual C++ で使用する場合の変数宣言例：char S_Variable[17];

形態

形態のデータは、以下のビットデータを 10 進数にした値となります。



＊ ERC, ERC2 の場合は D3 － D7 の値は無視されます。
＊ MRC, MRC2 の場合は D5 － D7 の値は無視されます。

数値変数格納エリアの内容

数値変数格納エリアの内容、有効配列数は変数タイプによって以下のようになります。

変数ﾀｲﾌﾟ番号	有効配列数	内 容					
		p[0]	p[1]	p[2]	p[3]	p[4]	p[5]
1	1	ﾊﾞｲﾄ値	－	－	－	－	－
2	1	整数値	－	－	－	－	－
3	1	倍精度整数値	－	－	－	－	－
4	1	実数値	－	－	－	－	－
5	6	S 軸ﾊﾟﾙｽ数	L 軸ﾊﾟﾙｽ数	U 軸ﾊﾟﾙｽ数	R 軸ﾊﾟﾙｽ数	B 軸ﾊﾟﾙｽ数	T 軸ﾊﾟﾙｽ数
6	6	X 軸座標値(mm)	Y 軸座標値(mm)	Z 軸座標値(mm)	手首姿勢 Rx 座標値(deg)	手首姿勢 Ry 座標値(deg)	手首姿勢 Rz 座標値(deg)
7	6	ﾍﾞｰｽ第 1 軸ﾊﾟﾙｽ数	ﾍﾞｰｽ第 2 軸ﾊﾟﾙｽ数	ﾍﾞｰｽ第 3 軸ﾊﾟﾙｽ数	ﾍﾞｰｽ第 4 軸ﾊﾟﾙｽ数	ﾍﾞｰｽ第 5 軸ﾊﾟﾙｽ数	ﾍﾞｰｽ第 6 軸ﾊﾟﾙｽ数
8	6	ﾍﾞｰｽ第 1 軸直交値(mm)	ﾍﾞｰｽ第 2 軸直交値(mm)	ﾍﾞｰｽ第 3 軸直交値(mm)	ﾍﾞｰｽ第 4 軸直交値(mm)	ﾍﾞｰｽ第 5 軸直交値(mm)	ﾍﾞｰｽ第 6 軸直交値(mm)
9	6	ｽﾃｰｼｮﾝ第 1 ﾊﾟﾙｽ数	ｽﾃｰｼｮﾝ第 2 ﾊﾟﾙｽ数	ｽﾃｰｼｮﾝ第 3 ﾊﾟﾙｽ数	ｽﾃｰｼｮﾝ第 4 ﾊﾟﾙｽ数	ｽﾃｰｼｮﾝ第 5 ﾊﾟﾙｽ数	ｽﾃｰｼｮﾝ第 6 ﾊﾟﾙｽ数

変数ﾀｲﾌﾟ番号	有効配列数	内 容
		str
10	16	文字列

コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC（シリアルポート、Ethernet） ERC（シリアルポート）
参 照	「BscDCILoadSave」「BscDCILoadSaveOnce」「BscDCIPutPos」「BscDCIPutPos2」「BscDCIPutVarData」

■ BscDCIGetVarDataEx

機 能	DCI インストラクションによる変数の受信を行います。(7 軸以上対応)
書 式	_declspec(dllexport) short APIENTRY BscDCIGetVarDataEx(short nCid,short *type,long *rconf,double *p, char *str,short *axisNum);
引 数	**IN（引き渡し）** nCid　　通信ハンドラの ID 番号 *type　　変数タイプ番号ポインタ *rconf　　形態データ格納ポインタ *p　　数値変数格納エリアの先頭ポインタ *str　　文字変数格納エリアの先頭ポインタ *axisNum　軸数格納ポインタ
	OUT（戻り） *rconf　　形態データ格納ポインタ *p　　数値変数格納エリアの先頭ポインタ *str　　文字変数格納エリアの先頭ポインタ *axisNum　軸数格納ポインタ
	リターン値 -1　　：受信失敗 その他：変数タイプ番号
解 説	**制約事項** 本関数はコントローラが YRC1000 及び YRC1000micro、DX200、DX100、FS100、NX100 Ver3.0 以降の場合に限り文字型変数を取り扱うことが可能です。
	変数タイプ番号 変数タイプ番号の意味は以下のようになります。

コントローラ / タイプ番号	YRC1000/YRC1000micro/DX200 DX100/FS100/NX100 V3.0 以降	NX100(V3.0 未満)/ XRC/MRC	ERC
1	バイト型変数		
2	整数型変数		
3	倍精度型変数		
4	実数型変数		
5	ロボット軸位置型変数（パルス型）		
6	ロボット軸位置型変数（直交型）		
7	ベース軸位置型変数（パルス型）		
8	ベース軸位置型変数（直交型）		
9	ステーション軸位置型変数（パルス型）		―
10	文字型変数	―	―

重要

- コントローラに7軸ロボットが1台でも登録されていれば、P変数は7軸用となります。
- 文字型変数はYRC1000及びYRC1000micro、DX200、DX100、FS100、NX100 Ver3.0以降のバージョンで有効です。

解 説

変数タイプ番号と使用する格納エリア、配列数

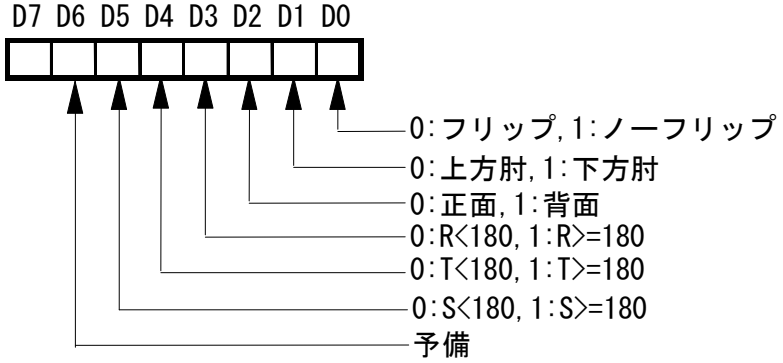
タイプ番号	使用する格納エリア	必要な配列数
1～9	p	7
10	str	16

重要

文字型変数の送受信で本関数をコールする場合、変数格納エリアは以下の例のように 17 文字分の変数エリアを確保してください。
Visual Basic で使用する場合の変数宣言例：
Dim S_Variable As String *17
Visual C++ で使用する場合の変数宣言例：char S_Variable[17];

形態

形態のデータは、以下のビットデータを 10 進数にした値となります。



＊ ERC, ERC2 の場合は D3 － D7 の値は無視されます。
＊ MRC, MRC2 の場合は D5 － D7 の値は無視されます。

数値変数格納エリアの内容

数値変数格納エリアの内容、有効配列数は変数タイプによって以下のようになります。

変数ﾀｲﾌﾟ番号	有効配列数	内 容						
		p[0]	p[1]	p[2]	p[3]	p[4]	p[5]	p[6]
1	1	ﾊﾞｲﾄ値	－	－	－	－	－	－
2	1	整数値	－	－	－	－	－	－
3	1	倍精度整数値	－	－	－	－	－	－
4	1	実数値	－	－	－	－	－	－
5	7	S 軸ﾊﾟﾙｽ数	L 軸ﾊﾟﾙｽ数	U 軸ﾊﾟﾙｽ数	R 軸ﾊﾟﾙｽ数	B 軸ﾊﾟﾙｽ数	T 軸ﾊﾟﾙｽ数	E 軸ﾊﾟﾙｽ数
6	7	X 軸座標値 (mm)	Y 軸座標値 (mm)	Z 軸座標値 (mm)	手首姿勢 Rx 座標値 (deg)	手首姿勢 Ry 座標値 (deg)	手首姿勢 Rz 座標値 (deg)	Re (deg)
7	6	ﾍﾞｰｽ第 1 軸ﾊﾟﾙｽ数	ﾍﾞｰｽ第 2 軸ﾊﾟﾙｽ数	ﾍﾞｰｽ第 3 軸ﾊﾟﾙｽ数	ﾍﾞｰｽ第 4 軸ﾊﾟﾙｽ数	ﾍﾞｰｽ第 5 軸ﾊﾟﾙｽ数	ﾍﾞｰｽ第 6 軸ﾊﾟﾙｽ数	－
8	6	ﾍﾞｰｽ第 1 軸直交値 (mm)	ﾍﾞｰｽ第 2 軸直交値 (mm)	ﾍﾞｰｽ第 3 軸直交値 (mm)	ﾍﾞｰｽ第 4 軸直交値 (mm)	ﾍﾞｰｽ第 5 軸直交値 (mm)	ﾍﾞｰｽ第 6 軸直交値 (mm)	－
9	6	ｽﾃｰｼｮﾝ第 1 ﾊﾟﾙｽ数	ｽﾃｰｼｮﾝ第 2 ﾊﾟﾙｽ数	ｽﾃｰｼｮﾝ第 3 ﾊﾟﾙｽ数	ｽﾃｰｼｮﾝ第 4 ﾊﾟﾙｽ数	ｽﾃｰｼｮﾝ第 5 ﾊﾟﾙｽ数	ｽﾃｰｼｮﾝ第 6 ﾊﾟﾙｽ数	－

変数ﾀｲﾌﾟ番号	有効配列数	内 容
		str
10	16	文字列

コントローラ

YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC（シリアルポート、Ethernet）
ERC（シリアルポート）

参 照	「BscDCILoadSave」 「BscDCILoadSaveOnce」 「BscDCIPutPos」 「BscDCIPutPos2」 「BscDCIPutVarData」 「BscDCIPutVarDataEx」

■ BscDCIPutPos

機 能	DCI インストラクションによる変数の送信を行います。
書 式	_declspec(dllexport) short APIENTRY BscDCIPutPos(short nCid,short type,short rconf,double *p);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号 type　変数タイプ番号 rconf　形態データ *p　数値変数格納エリアの先頭ポインタ
	OUT（戻り） 無し
	リターン値 -1　：送信失敗 その他：正常終了
解 説	**変数タイプ番号** 変数タイプ番号の意味は以下のようになります。

コントローラ / タイプ番号	YRC1000/YRC1000micro/DX200/DX100 FS100/NX100/XRC/MRC	ERC
1	バイト型変数	
2	整数型変数	
3	倍精度型変数	
4	実数型変数	
5	ロボット軸位置型変数（パルス型）	
6	ロボット軸位置型変数（直交型）	
7	ベース軸位置型変数（パルス型）	
8	ベース軸位置型変数（直交型）	
9	ステーション軸位置型変数（パルス型）	ー

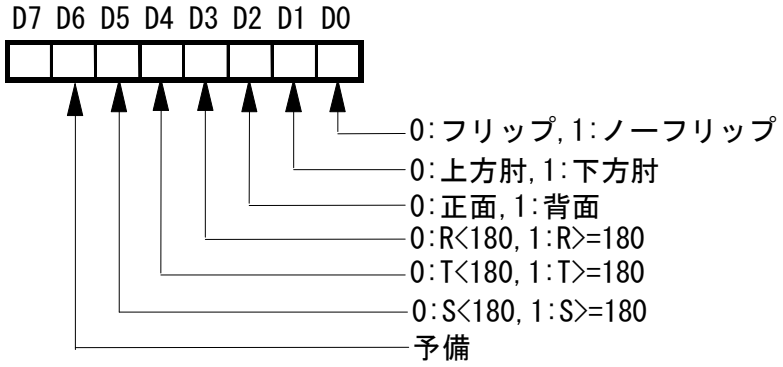
変数タイプ番号と使用する格納エリア、配列数

タイプ番号	使用する格納エリア	必要な配列数
1～9	p	6

解 説

形態

形態のデータは、以下のビットデータを 10 進数にした値となります。



＊ ERC、ERC2 の場合は D3 － D7 データは無視されます。
＊ MRC、MRC2 の場合は D5 － D7 データは無視されます。

数値変数格納エリアの内容

数値変数格納エリアの内容、有効配列数は変数タイプによって以下のようになります。

変数ﾀｲﾌﾟ番号	有効配列数	内 容					
		p[0]	p[1]	p[2]	p[3]	p[4]	p[5]
1	1	ﾊﾞｲﾄ値	－	－	－	－	－
2	1	整数値	－	－	－	－	－
3	1	倍精度整数値	－	－	－	－	－
4	1	実数値	－	－	－	－	－
5	6	S 軸ﾊﾟﾙｽ数	L 軸ﾊﾟﾙｽ数	U 軸ﾊﾟﾙｽ数	R 軸ﾊﾟﾙｽ数	B 軸ﾊﾟﾙｽ数	T 軸ﾊﾟﾙｽ数
6	6	X 軸座標値(mm)	Y 軸座標値(mm)	Z 軸座標値(mm)	手首姿勢 Rx 座標値(deg)	手首姿勢 Ry 座標値(deg)	手首姿勢 Rz 座標値(deg)
7	6	ﾍﾞｰｽ第 1 軸ﾊﾟﾙｽ数	ﾍﾞｰｽ第 2 軸ﾊﾟﾙｽ数	ﾍﾞｰｽ第 3 軸ﾊﾟﾙｽ数	ﾍﾞｰｽ第 4 軸ﾊﾟﾙｽ数	ﾍﾞｰｽ第 5 軸ﾊﾟﾙｽ数	ﾍﾞｰｽ第 6 軸ﾊﾟﾙｽ数
8	6	ﾍﾞｰｽ第 1 軸直交値(mm)	ﾍﾞｰｽ第 2 軸直交値(mm)	ﾍﾞｰｽ第 3 軸直交値(mm)	ﾍﾞｰｽ第 4 軸直交値(mm)	ﾍﾞｰｽ第 5 軸直交値(mm)	ﾍﾞｰｽ第 6 軸直交値(mm)
9	6	ｽﾃｰｼｮﾝ第 1 ﾊﾟﾙｽ数	ｽﾃｰｼｮﾝ第 2 ﾊﾟﾙｽ数	ｽﾃｰｼｮﾝ第 3 ﾊﾟﾙｽ数	ｽﾃｰｼｮﾝ第 4 ﾊﾟﾙｽ数	ｽﾃｰｼｮﾝ第 5 ﾊﾟﾙｽ数	ｽﾃｰｼｮﾝ第 6 ﾊﾟﾙｽ数

コントローラ

YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC（シリアルポート、Ethernet）
ERC（シリアルポート）

参 照

「BscDCILoadSave」「BscDCILoadSaveOnce」「BscDCIGetPos」「BscDCIGetPos2」「BscDCIGetVarData」

■ BscDCIPutPos2

機 能	DCI インストラクションによる変数の送信を行います。(7 軸以上対応)
書 式	_declspec(dllexport) short APIENTRY BscDCIPutPos2(short nCid,short type,short rconf,double *p,short axisNum);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号 type　変数タイプ番号 rconf　形態データ *p　数値変数格納エリアの先頭ポインタ axisNum　軸数
	OUT（戻り） 無し
	リターン値 -1　：送信失敗 その他：正常終了
解 説	**変数タイプ番号** 変数タイプ番号の意味は以下のようになります。

タイプ番号 ＼ コントローラ	YRC1000/YRC1000micro/DX200 DX100/FS100/NX100/XRC/MRC	ERC
1	バイト型変数	
2	整数型変数	
3	倍精度型変数	
4	実数型変数	
5	ロボット軸位置型変数（パルス型）	
6	ロボット軸位置型変数（直交型）	
7	ベース軸位置型変数（パルス型）	
8	ベース軸位置型変数（直交型）	
9	ステーション軸位置型変数（パルス型）	ー

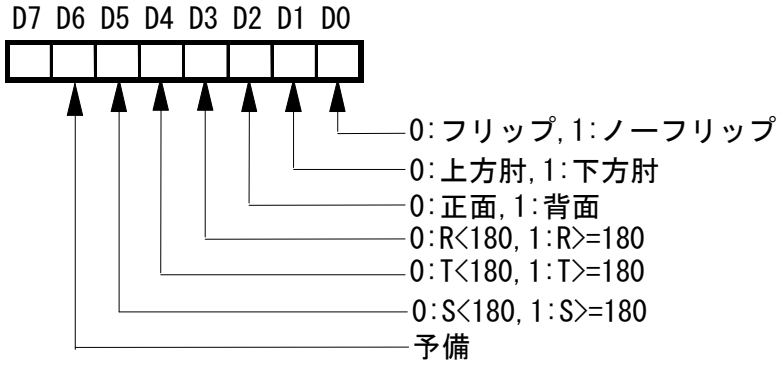
変数タイプ番号と使用する格納エリア、配列数

タイプ番号	使用する格納エリア	必要な配列数
1～9	p	8

解 説

形態

形態のデータは、以下のビットデータを 10 進数にした値となります。



*ERC、ERC2 の場合は D3 － D7 データは無視されます。
*MRC、MRC2 の場合は D5 － D7 データは無視されます。

数値変数格納エリアの内容

数値変数格納エリアの内容、有効配列数は変数タイプによって以下のようになります。

変数ﾀｲﾌﾟ番号	有効配列数	内 容					
		p[0]	p[1]	p[2]	p[3]	p[4]	p[5]
1	1	ﾊﾞｲﾄ値	－	－	－	－	－
2	1	整数値	－	－	－	－	－
3	1	倍精度整数値	－	－	－	－	－
4	1	実数値	－	－	－	－	－
5	8	S 軸ﾊﾟﾙｽ数	L 軸ﾊﾟﾙｽ数	U 軸ﾊﾟﾙｽ数	R 軸ﾊﾟﾙｽ数	B 軸ﾊﾟﾙｽ数	T 軸ﾊﾟﾙｽ数
6	6	X 軸座標値(mm)	Y 軸座標値(mm)	Z 軸座標値(mm)	手首姿勢 Rx 座標値(deg)	手首姿勢 Ry 座標値(deg)	手首姿勢 Rz 座標値(deg)
7	6	ﾍﾞｰｽ第 1 軸ﾊﾟﾙｽ数	ﾍﾞｰｽ第 2 軸ﾊﾟﾙｽ数	ﾍﾞｰｽ第 3 軸ﾊﾟﾙｽ数	ﾍﾞｰｽ第 4 軸ﾊﾟﾙｽ数	ﾍﾞｰｽ第 5 軸ﾊﾟﾙｽ数	ﾍﾞｰｽ第 6 軸ﾊﾟﾙｽ数
8	6	ﾍﾞｰｽ第 1 軸直交値(mm)	ﾍﾞｰｽ第 2 軸直交値(mm)	ﾍﾞｰｽ第 3 軸直交値(mm)	ﾍﾞｰｽ第 4 軸直交値(mm)	ﾍﾞｰｽ第 5 軸直交値(mm)	ﾍﾞｰｽ第 6 軸直交値(mm)
9	6	ｽﾃｰｼｮﾝ第 1 ﾊﾟﾙｽ数	ｽﾃｰｼｮﾝ第 2 ﾊﾟﾙｽ数	ｽﾃｰｼｮﾝ第 3 ﾊﾟﾙｽ数	ｽﾃｰｼｮﾝ第 4 ﾊﾟﾙｽ数	ｽﾃｰｼｮﾝ第 5 ﾊﾟﾙｽ数	ｽﾃｰｼｮﾝ第 6 ﾊﾟﾙｽ数

変数ﾀｲﾌﾟ番号	有効配列数	内 容	
		p[6]	p[7]
1	1	－	－
2	1	－	－
3	1	－	－
4	1	－	－
5	8	7 軸ﾊﾟﾙｽ数	8 軸ﾊﾟﾙｽ数
6	6	－	－
7	6	－	－
8	6	－	－
9	6	－	－

コントローラ

YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC（シリアルポート、Ethernet）

参 照

「BscDCILoadSave」「BscDCILoadSaveOnce」「BscDCIGetPos」「BscDCIGetPos2」「BscDCIGetVarData」

■ BscDCIPutVarData

機 能	DCI インストラクションによる変数の送信を行います。
書 式	_declspec(dllexport) short APIENTRY BscDCIPutVarData(short nCid,short *type,short rconf,double *p, char *str);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号 type　変数タイプ番号 rconf　形態データ *p　数値変数格納エリアの先頭ポインタ *str　文字変数格納エリアの先頭ポインタ
	OUT（戻り） 無し
	リターン値 -1　　：送信失敗 その他：正常終了
解 説	**制約事項** 本関数はコントローラが YRC1000 及び YRC1000micro、DX200、DX100、FS100、NX100 Ver3.0 以降の場合に限り文字型変数を取り扱うことが可能です。
	変数タイプ番号 変数タイプ番号の意味は以下のようになります。

コントローラ / タイプ番号	YRC1000/YRC1000micro/DX200 DX100/FS100/NX100 V3.0 以降	NX100(V3.0 未満)/ XRC/MRC	ERC
1	バイト型変数		
2	整数型変数		
3	倍精度型変数		
4	実数型変数		
5	ロボット軸位置型変数（パルス型）		
6	ロボット軸位置型変数（直交型）		
7	ベース軸位置型変数（パルス型）		
8	ベース軸位置型変数（直交型）		
9	ステーション軸位置型変数（パルス型）		―
10	文字型変数	―	―

重要　文字型変数は YRC1000 及び YRC1000micro、DX200、DX100、FS100、NX100 Ver3.0 以降のバージョンで有効です。

解 説

変数タイプ番号と使用する格納エリア、配列数

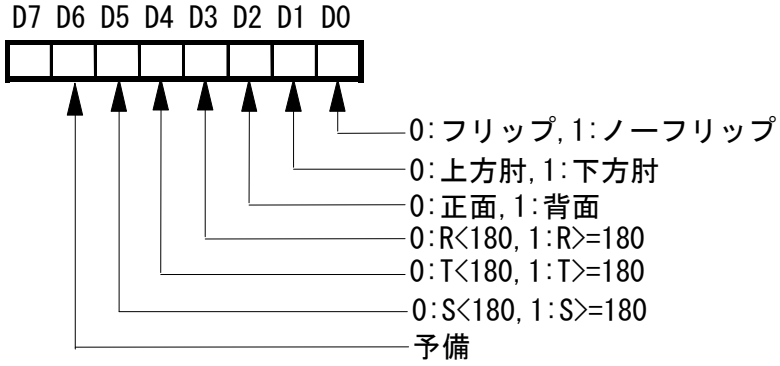
タイプ番号	使用する格納エリア	必要な配列数
1～9	p	6
10	str	16

重要

文字型変数の送受信で本関数をコールする場合、変数格納エリアは以下の例のように 17 文字分の変数エリアを確保してください。
Visual Basic で使用する場合の変数宣言例：
Dim S_Variable As String *17
Visual C++ で使用する場合の変数宣言例：char S_Variable[17];

形態

形態のデータは、以下のビットデータを 10 進数にした値となります。



*ERC、ERC2 の場合は D3 － D7 の値は無視されます。
*MRC、MRC2 の場合は D5 － D7 の値は無視されます。

数値変数格納エリアの内容

数値変数格納エリアの内容、有効配列数は変数タイプによって以下のようになります。

変数ﾀｲﾌﾟ番号	有効配列数	内 容					
		p[0]	p[1]	p[2]	p[3]	p[4]	p[5]
1	1	ﾊﾞｲﾄ値	－	－	－	－	－
2	1	整数値	－	－	－	－	－
3	1	倍精度整数値	－	－	－	－	－
4	1	実数値	－	－	－	－	－
5	6	S 軸ﾊﾟﾙｽ数	L 軸ﾊﾟﾙｽ数	U 軸ﾊﾟﾙｽ数	R 軸ﾊﾟﾙｽ数	B 軸ﾊﾟﾙｽ数	T 軸ﾊﾟﾙｽ数
6	6	X 軸座標値(mm)	Y 軸座標値(mm)	Z 軸座標値(mm)	手首姿勢 Rx 座標値(deg)	手首姿勢 Ry 座標値(deg)	手首姿勢 Rz 座標値(deg)
7	6	ﾍﾞｰｽ第 1 軸ﾊﾟﾙｽ数	ﾍﾞｰｽ第 2 軸ﾊﾟﾙｽ数	ﾍﾞｰｽ第 3 軸ﾊﾟﾙｽ数	ﾍﾞｰｽ第 4 軸ﾊﾟﾙｽ数	ﾍﾞｰｽ第 5 軸ﾊﾟﾙｽ数	ﾍﾞｰｽ第 6 軸ﾊﾟﾙｽ数
8	6	ﾍﾞｰｽ第 1 軸直交値(mm)	ﾍﾞｰｽ第 2 軸直交値(mm)	ﾍﾞｰｽ第 3 軸直交値(mm)	ﾍﾞｰｽ第 4 軸直交値(mm)	ﾍﾞｰｽ第 5 軸直交値(mm)	ﾍﾞｰｽ第 6 軸直交値(mm)
9	6	ｽﾃｰｼｮﾝ第 1 ﾊﾟﾙｽ数	ｽﾃｰｼｮﾝ第 2 ﾊﾟﾙｽ数	ｽﾃｰｼｮﾝ第 3 ﾊﾟﾙｽ数	ｽﾃｰｼｮﾝ第 4 ﾊﾟﾙｽ数	ｽﾃｰｼｮﾝ第 5 ﾊﾟﾙｽ数	ｽﾃｰｼｮﾝ第 6 ﾊﾟﾙｽ数

変数ﾀｲﾌﾟ番号	有効配列数	内 容
		str
10	16	文字列

コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC（シリアルポート、Ethernet） ERC（シリアルポート）
参 照	「BscDCILoadSave」「BscDCILoadSaveOnce」「BscDCIGetPos」 「BscDCIGetPos2」「BscDCIGetVarData」

■ BscDCIPutVarDataEx

機 能	DCI インストラクションによる変数の送信を行います。（7 軸以上対応）
書 式	_declspec(dllexport) short APIENTRY BscDCIPutVarDataEx(short nCid,short *type,long rconf,double *p, char *str,short axisNum);
引 数	**IN（引き渡し）** nCid　　通信ハンドラの ID 番号 type　　変数タイプ番号 rconf　　形態データ *p　　数値変数格納エリアの先頭ポインタ *str　　文字変数格納エリアの先頭ポインタ axisNum　軸数
	OUT（戻り） 無し
	リターン値 -1　　：送信失敗 その他：正常終了
解 説	**制約事項** 本関数はコントローラが YRC1000 及び YRC1000micro、DX200、DX100、FS100、NX100 Ver3.0 以降の場合に限り文字型変数を取り扱うことが可能です。
	変数タイプ番号 変数タイプ番号の意味は以下のようになります。

コントローラ / タイプ番号	YRC1000/YRC1000micro/DX200/ DX100/FS100/NX100 V3.0 以降	NX100(V3.0 未満)/ XRC/MRC	ERC
1	バイト型変数		
2	整数型変数		
3	倍精度型変数		
4	実数型変数		
5	ロボット軸位置型変数（パルス型）		
6	ロボット軸位置型変数（直交型）		
7	ベース軸位置型変数（パルス型）		
8	ベース軸位置型変数（直交型）		
9	ステーション軸位置型変数（パルス型）		―
10	文字型変数	―	―

重要

- コントローラに7軸ロボットが1台でも登録されていれば、P変数は7軸用となります。
- 文字型変数はYRC1000及びYRC1000micro、DX200、DX100、FS100、NX100 Ver3.0以降のバージョンで有効です。

解 説

変数タイプ番号と使用する格納エリア、配列数

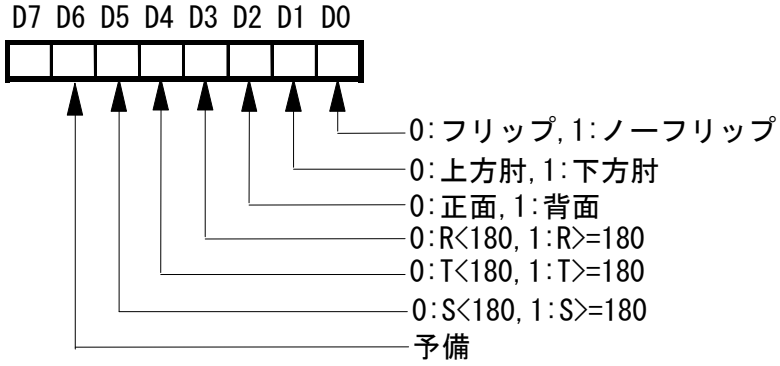
タイプ番号	使用する格納エリア	必要な配列数
1～9	p	7
10	str	16

重要

文字型変数の送受信で本関数をコールする場合、変数格納エリアは以下の例のように 17 文字分の変数エリアを確保してください。
Visual Basic で使用する場合の変数宣言例：
Dim S_Variable As String *17
Visual C++ で使用する場合の変数宣言例：char S_Variable[17];

形態

形態のデータは、以下のビットデータを 10 進数にした値となります。



*ERC、ERC2 の場合は D3 － D7 の値は無視されます。
*MRC、MRC2 の場合は D5 － D7 の値は無視されます。

数値変数格納エリアの内容

数値変数格納エリアの内容、有効配列数は変数タイプによって以下のようになります。

変数ﾀｲﾌﾟ番号	有効配列数	内 容						
		p[0]	p[1]	p[2]	p[3]	p[4]	p[5]	p[6]
1	1	ﾊﾞｲﾄ値	－	－	－	－	－	－
2	1	整数値	－	－	－	－	－	－
3	1	倍精度整数値	－	－	－	－	－	－
4	1	実数値	－	－	－	－	－	－
5	7	S 軸ﾊﾟﾙｽ数	L 軸ﾊﾟﾙｽ数	U 軸ﾊﾟﾙｽ数	R 軸ﾊﾟﾙｽ数	B 軸ﾊﾟﾙｽ数	T 軸ﾊﾟﾙｽ数	E 軸ﾊﾟﾙｽ数
6	7	X 軸座標値 (mm)	Y 軸座標値 (mm)	Z 軸座標値 (mm)	手首姿勢 Rx 座標値(deg)	手首姿勢 Ry 座標値(deg)	手首姿勢 Rz 座標値(deg)	Re (deg)
7	6	ﾍﾞｰｽ第 1 軸ﾊﾟﾙｽ数	ﾍﾞｰｽ第 2 軸ﾊﾟﾙｽ数	ﾍﾞｰｽ第 3 軸ﾊﾟﾙｽ数	ﾍﾞｰｽ第 4 軸ﾊﾟﾙｽ数	ﾍﾞｰｽ第 5 軸ﾊﾟﾙｽ数	ﾍﾞｰｽ第 6 軸ﾊﾟﾙｽ数	－
8	6	ﾍﾞｰｽ第 1 軸直交値(mm)	ﾍﾞｰｽ第 2 軸直交値(mm)	ﾍﾞｰｽ第 3 軸直交値(mm)	ﾍﾞｰｽ第 4 軸直交値(mm)	ﾍﾞｰｽ第 5 軸直交値(mm)	ﾍﾞｰｽ第 6 軸直交値(mm)	－
9	6	ｽﾃｰｼｮﾝ第 1 ﾊﾟﾙｽ数	ｽﾃｰｼｮﾝ第 2 ﾊﾟﾙｽ数	ｽﾃｰｼｮﾝ第 3 ﾊﾟﾙｽ数	ｽﾃｰｼｮﾝ第 4 ﾊﾟﾙｽ数	ｽﾃｰｼｮﾝ第 5 ﾊﾟﾙｽ数	ｽﾃｰｼｮﾝ第 6 ﾊﾟﾙｽ数	－

変数ﾀｲﾌﾟ番号	有効配列数	内 容
		str
10	16	文字列

コントローラ

YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC（シリアルポート、Ethernet）
ERC（シリアルポート）

参 照	「BscDCILoadSave」 「BscDCILoadSaveOnce」 「BscDCIGetPos」 「BscDCIGetPos2」 「BscDCIGetVarData」 「BscDCIGetVarDataEx」

7.5 I/O 信号のリード・ライト機能

Ｉ／Ｏ信号のリード（読み込み）・ライト（書き込み）を行います。
以下の関数が使用できます。

BscReadIO
BscReadIO2
BscWriteIO
BscWriteIO2

■ BscReadIO

機 能	指定されたコイル番号から指定された個数だけ、連続した番号のコイル状態を読み込みます。指定できる個数の最大値は 256 個です。
書 式	_declspec(dllexport) short APIENTRY BscReadIO(short nCid,short add,short num,short *stat);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号 add　読み込み開始アドレス番号 num　読み込み信号個数（Max.256 個） *stat　コイルの状態
	OUT（戻り） *stat　コイルの状態
	リターン値 -1　　：ヘッダ番号異常 0　　：正常終了 その他：エラーコード
解 説	**制約事項** 本関数は MRC、XRC に対して有効です。 YRC1000、YRC1000micro、DX200、DX100、FS100、NX100 に対しては、BscReadIO2 を参照してください。
	不要信号 読み込み個数が 8 の倍数でない場合、不要信号は全て 0 がセットされます。

I/O 番号一覧

MRC

信号	信号の範囲	分類名	読み込み	書き込み
0xxx	0010-0167(128 個)	ロボット汎用入力	○	×
1xxx	1010-1167(128 個)	ロボット汎用出力	○	×
2xxx	2010-2187(144 個)	外部入力	○	×
3xxx	3010-3187(144 個)	外部出力	○	×
4xxx	4010-4167(128 個)	ロボット専用入力	○	×
5xxx	5010-5247(192 個)	ロボット専用出力	○	×
6xxx	6010-6047 (32 個)	タイマ／カウンタ	×	×
7xxx	7010-7327(256 個)	補助リレー	○	×
8xxx	8010-8087 (64 個)	制御状態信号	○	×
82xx	8210-8247 (32 個)	擬似入力信号	○	×
9xxx	9010-9167(128 個)	DL 入力	○	○

XRC

信号	信号の範囲	分類名	読み込み	書き込み
0xxx	0010-0247(192 個)	ロボット汎用入力	○	×
1xxx	1010-1247(192 個)	ロボット汎用出力	○	×
2xxx	2010-2327(256 個)	外部入力	○	×
3xxx	3010-3327(256 個)	外部出力	○	×
4xxx	4010-4287(224 個)	ロボット専用入力	○	×
5xxx	5010-5387(304 個)	ロボット専用出力	○	×
6xxx	ー	ー	×	×
7xxx	7010-7887(704 個)	補助リレー	○	×
8xxx	8010-8127 (96 個)	制御状態信号	○	×
82xx	8210-8247 (32 個)	擬似入力信号	○	×
9xxx	9010-9167(128 個)	ネットワーク入力	○	○

コントローラ	XRC、MRC（シリアルポート、Ethernet）
参 照	「BscWriteIO」「BscReadIO2」「BscWriteIO2」

■ BscReadIO2

機 能	指定されたコイル番号から指定された個数だけ、連続した番号のコイル状態を読み込みます。指定できる個数の最大値は 256 個です。
書 式	_declspec(dllexport) short APIENTRY BscReadIO2(short nCid,DWORD add,short num,short *stat);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号 add　読み込み開始アドレス番号 num　読み込み信号個数（Max.256 個） *stat　コイルの状態
	OUT（戻り） *stat　コイルの状態
	リターン値 -1　　：ヘッダ番号異常 0　　：正常終了 その他：エラーコード
解 説	**制約事項** 本関数は YRC1000、YRC1000micro、DX200、DX100、FS100、NX100 に対して有効です。 MRC、XRC に対しては、BscReadIO を参照してください。
	不要信号 読み込み個数が 8 の倍数でない場合、不要信号は全て 0 がセットされます。
	I/O 番号一覧 **YRC1000**

信号	信号の範囲	分類名	読み込み	書き込み
0xxx	00010-05127(4096 個)	汎用入力	○	×
1xxx	10010-15127(4096 個)	汎用出力	○	×
2xxx	20010-25127(4096 個)	外部入力	○	×
27xx	27010-29567(2048 個)	ネットワーク入力	○	○
3xxx	30010-35127(4096 個)	外部出力	○	×
37xx	37010-39567(2048 個)	ネットワーク出力	○	×
4xxx	40010-42567(2048 個)	専用入力（システム）	○	×
5xxx	50010-55127(4096 個)	専用出力（システム）	○	×
6xxx	ー	ー	×	×
7xxx	70010-79997(7992 個)	補助リレー（システム）	○	×
8xxx	80010-85127(4096 個)	制御状態信号(システム)	○	×
87xx	87010-87207（160 個）	擬似入力信号(システム)	○	×

解 説

YRC1000micro

信号	信号の範囲	分類名	読み込み	書き込み
0xxx	00010-05127(4096 個)	汎用入力	○	×
1xxx	10010-15127(4096 個)	汎用出力	○	×
2xxx	20010-21287(1024 個)	外部入力	○	×
27xx	27010-29567(2048 個)	ネットワーク入力	○	○
3xxx	30010-31287(1024 個)	外部出力	○	×
37xx	37010-39567(2048 個)	ネットワーク出力	○	×
4xxx	40010-42567(2048 個)	専用入力（システム）	○	×
5xxx	50010-55127(4096 個)	専用出力（システム）	○	×
6xxx	―	―	×	×
7xxx	70010-79997(7992 個)	補助リレー（システム）	○	×
8xxx	80010-85127(4096 個)	制御状態信号(システム)	○	×
87xx	87010-87207（160 個）	擬似入力信号(システム)	○	×

DX200

信号	信号の範囲	分類名	読み込み	書き込み
0xxx	00010-05127(4096 個)	汎用入力	○	×
1xxx	10010-15127(4096 個)	汎用出力	○	×
2xxx	20010-25127(4096 個)	外部入力	○	×
27xx	27010-29567(2048 個)	ネットワーク入力	○	○
3xxx	30010-35127(4096 個)	外部出力	○	×
37xx	37010-39567(2048 個)	ネットワーク出力	○	×
4xxx	40010-41607(1280 個)	専用入力（システム）	○	×
5xxx	50010-53007(2400 個)	専用出力（システム）	○	×
6xxx	―	―	×	×
7xxx	70010-79997(7992 個)	補助リレー（システム）	○	×
8xxx	80010-80647（512 個）	制御状態信号(システム)	○	×
82xx	82010-82207（160 個）	擬似入力信号(システム)	○	×

DX100

信号	信号の範囲	分類名	読み込み	書き込み
0xxx	00010-02567(2048 個)	汎用入力	○	×
1xxx	10010-12567(2048 個)	汎用出力	○	×
2xxx	20010-22567(2048 個)	外部入力	○	×
25xx	25010-27567(2048 個)	ネットワーク入力	○	○
3xxx	30010-32567(2048 個)	外部出力	○	×
35xx	35010-37567(2048 個)	ネットワーク出力	○	×
4xxx	40010-41607(1280 個)	専用入力（システム）	○	×
5xxx	50010-52007(1600 個)	専用出力（システム）	○	×
6xxx	―	―	×	×
7xxx	70010-79997(7992 個)	補助リレー（システム）	○	×
8xxx	80010-80647（512 個）	制御状態信号(システム)	○	×
82xx	82010-82207（160 個）	擬似入力信号(システム)	○	×

解 説

FS100

信号	信号の範囲	分類名	読み込み	書き込み
0xxx	00010-01287(1024 個)	汎用入力	○	×
1xxx	10010-11287(1024 個)	汎用出力	○	×
2xxx	20010-21287(1024 個)	外部入力	○	×
25xx	25010-26287(1024 個)	ネットワーク入力	○	○
3xxx	30010-31287(1024 個)	外部出力	○	×
35xx	35010-36287(1024 個)	ネットワーク出力	○	×
4xxx	40010-41607(1280 個)	専用入力（システム）	○	×
5xxx	50010-52007(1600 個)	専用出力（システム）	○	×
6xxx	―	―	×	×
7xxx	70010-79997(7992 個)	補助リレー（システム）	○	×
8xxx	80010-80647（512 個）	制御状態信号(システム)	○	×
82xx	82010-82207（160 個）	擬似入力信号(システム)	○	×

NX100

信号	信号の範囲	分類名	読み込み	書き込み
0xxx	00010-01287(1024 個)	汎用入力	○	×
1xxx	10010-11287(1024 個)	汎用出力	○	×
2xxx	20010-21287(1024 個)	外部入力	○	×
22xx	22010-23287(1024 個)	ネットワーク入力	○	○
3xxx	30010-31287(1024 個)	外部出力	○	×
32xx	32010-33287(1024 個)	ネットワーク出力	○	×
4xxx	40010-40807（640 個）	専用入力（システム）	○	×
5xxx	50010-51007（800 個）	専用出力（システム）	○	×
6xxx	―	―	×	×
7xxx	70010-79997(7992 個)	補助リレー（システム）	○	×
8xxx	80010-80647（512 個）	制御状態信号(システム)	○	×
82xx	82010-82127（96 個）	擬似入力信号(システム)	○	×

コントローラ

YRC1000、YRC1000micro、DX200、DX100、FS100、NX100（シリアルポート、Ethernet）

参 照

「BscWriteIO2」「BscReadIO」「BscWriteIO」

■ BscWriteIO

機 能	指定されたコイル番号から指定された個数だけ、連続した番号のコイル状態を書き込みます。指定できる個数の最大値は 256 個です。書き込み可能なアドレス番号は 9000 番台のみです。
書 式	_declspec(dllexport) short APIENTRY BscWriteIO(short nCid,short add,short num,short *stat);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号 add　書き込み開始アドレス番号 num　書き込み信号個数（Max.256 個） *stat　コイルの状態
	OUT（戻り） *stat　コイルの状態
	リターン値 -1　　：ヘッダ番号異常 0　　：正常終了 その他：エラーコード
解 説	**制約事項** 本関数は MRC、XRC に対して有効です。 YRC1000、YRC1000micro、DX200、DX100、FS100、NX100 に対しては、BscWriteIO2 を参照してください。
	不要信号 書き込み個数が 8 の倍数でない場合、最後のデータ中に不要データが存在することになります。

解 説

I/O 番号一覧

MRC

信号	信号の範囲	分類名	読み込み	書き込み
0xxx	0010-0167(128 個)	ロボット汎用入力	○	×
1xxx	1010-1167(128 個)	ロボット汎用出力	○	×
2xxx	2010-2187(144 個)	外部入力	○	×
3xxx	3010-3187(144 個)	外部出力	○	×
4xxx	4010-4167(128 個)	ロボット専用入力	○	×
5xxx	5010-5247(192 個)	ロボット専用出力	○	×
6xxx	6010-6047 (32 個)	タイマ／カウンタ	×	×
7xxx	7010-7327(256 個)	補助リレー	○	×
8xxx	8010-8087 (64 個)	制御状態信号	○	×
82xx	8210-8247 (32 個)	擬似入力信号	○	×
9xxx	9010-9167(128 個)	DL 入力	○	○

XRC

信号	信号の範囲	分類名	読み込み	書き込み
0xxx	0010-0247(192 個)	ロボット汎用入力	○	×
1xxx	1010-1247(192 個)	ロボット汎用出力	○	×
2xxx	2010-2327(256 個)	外部入力	○	×
3xxx	3010-3327(256 個)	外部出力	○	×
4xxx	4010-4287(224 個)	ロボット専用入力	○	×
5xxx	5010-5387(304 個)	ロボット専用出力	○	×
6xxx	―	―	×	×
7xxx	7010-7887(704 個)	補助リレー	○	×
8xxx	8010-8127 (96 個)	制御状態信号	○	×
82xx	8210-8247 (32 個)	擬似入力信号	○	×
9xxx	9010-9167(128 個)	ネットワーク入力	○	○

コントローラ

XRC、MRC（シリアルポート、Ethernet）

参 照

「BscReadIO」「BscReadIO2」「BscWriteIO2」

■ BscWriteIO2

機 能	指定されたコイル番号から指定された個数だけ、連続した番号のコイル状態を書き込みます。指定できる個数の最大値は 256 個です。書き込み可能なアドレス番号は YRC1000、YRC1000micro、DX200 : 27000 番台、DX100、FS100、: 25000 番台、NX100 : 22000 番台のみです。
書 式	_declspec(dllexport) short APIENTRY BscWriteIO2(short nCid,DWORD add,short num,short *stat);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号 add　書き込み開始アドレス番号 num　書き込み信号個数（Max.256 個） *stat　コイルの状態
	OUT（戻り） *stat　コイルの状態
	リターン値 -1　　：ヘッダ番号異常 0　　：正常終了 その他：エラーコード
解 説	**制約事項** 本関数は YRC1000、YRC1000micro、DX200、DX100、FS100、NX100 に対して有効です。 MRC、XRC に対しては、BscWriteIO を参照してください。
	不要信号 書き込み個数が 8 の倍数でない場合、最後のデータ中に不要データが存在することになります。
	I/O 番号一覧 **YRC1000**

信号	信号の範囲	分類名	読み込み	書き込み
0xxx	00010-05127(4096 個)	汎用入力	○	×
1xxx	10010-15127(4096 個)	汎用出力	○	×
2xxx	20010-25127(4096 個)	外部入力	○	×
27xx	27010-29567(2048 個)	ネットワーク入力	○	○
3xxx	30010-35127(4096 個)	外部出力	○	×
37xx	37010-39567(2048 個)	ネットワーク出力	○	×
4xxx	40010-42567(2048 個)	専用入力（システム）	○	×
5xxx	50010-55127(4096 個)	専用出力（システム）	○	×
6xxx	―	―	×	×
7xxx	70010-79997(7992 個)	補助リレー（システム）	○	×
8xxx	80010-85127(4096 個)	制御状態信号(システム)	○	×
87xx	87010-87207（160 個）	擬似入力信号(システム)	○	×

解 説

YRC1000micro

信号	信号の範囲	分類名	読み込み	書き込み
0xxx	00010-05127(4096 個)	汎用入力	○	×
1xxx	10010-15127(4096 個)	汎用出力	○	×
2xxx	20010-21287(1024 個)	外部入力	○	×
27xx	27010-29567(2048 個)	ネットワーク入力	○	○
3xxx	30010-31287(1024 個)	外部出力	○	×
37xx	37010-39567(2048 個)	ネットワーク出力	○	×
4xxx	40010-42567(2048 個)	専用入力（システム）	○	×
5xxx	50010-55127(4096 個)	専用出力（システム）	○	×
6xxx	―	―	×	×
7xxx	70010-79997(7992 個)	補助リレー（システム）	○	×
8xxx	80010-85127(4096 個)	制御状態信号(システム)	○	×
87xx	87010-87207（160 個）	擬似入力信号(システム)	○	×

DX200

信号	信号の範囲	分類名	読み込み	書き込み
0xxx	00010-05127(4096 個)	汎用入力	○	×
1xxx	10010-15127(4096 個)	汎用出力	○	×
2xxx	20010-25127(4096 個)	外部入力	○	×
27xx	27010-29567(2048 個)	ネットワーク入力	○	○
3xxx	30010-35127(4096 個)	外部出力	○	×
37xx	37010-39567(2048 個)	ネットワーク出力	○	×
4xxx	40010-41607(1280 個)	専用入力（システム）	○	×
5xxx	50010-53007(2400 個)	専用出力（システム）	○	×
6xxx	―	―	×	×
7xxx	70010-79997(7992 個)	補助リレー（システム）	○	×
8xxx	80010-80647（512 個）	制御状態信号(システム)	○	×
82xx	82010-82207（160 個）	擬似入力信号(システム)	○	×

DX100

信号	信号の範囲	分類名	読み込み	書き込み
0xxx	00010-02567(2048 個)	汎用入力	○	×
1xxx	10010-12567(2048 個)	汎用出力	○	×
2xxx	20010-22567(2048 個)	外部入力	○	×
25xx	25010-27567(2048 個)	ネットワーク入力	○	○
3xxx	30010-32567(2048 個)	外部出力	○	×
35xx	35010-37567(2048 個)	ネットワーク出力	○	×
4xxx	40010-41607(1280 個)	専用入力（システム）	○	×
5xxx	50010-52007(1600 個)	専用出力（システム）	○	×
6xxx	―	―	×	×
7xxx	70010-79997(7992 個)	補助リレー（システム）	○	×
8xxx	80010-80647（512 個）	制御状態信号(システム)	○	×
82xx	82010-82207（160 個）	擬似入力信号(システム)	○	×

解 説	**FS100** **NX100**

FS100

信号	信号の範囲	分類名	読み込み	書き込み
0xxx	00010-01287(1024 個)	汎用入力	○	×
1xxx	10010-11287(1024 個)	汎用出力	○	×
2xxx	20010-21287(1024 個)	外部入力	○	×
25xx	25010-26287(1024 個)	ネットワーク入力	○	○
3xxx	30010-31287(1024 個)	外部出力	○	×
35xx	35010-36287(1024 個)	ネットワーク出力	○	×
4xxx	40010-41607(1280 個)	専用入力（システム）	○	×
5xxx	50010-52007(1600 個)	専用出力（システム）	○	×
6xxx	―	―	×	×
7xxx	70010-79997(7992 個)	補助リレー（システム）	○	×
8xxx	80010-80647（512 個）	制御状態信号(システム)	○	×
82xx	82010-82207（160 個）	擬似入力信号(システム)	○	×

NX100

信号	信号の範囲	分類名	読み込み	書き込み
0xxx	00010-01287(1024 個)	汎用入力	○	×
1xxx	10010-11287(1024 個)	汎用出力	○	×
2xxx	20010-21287(1024 個)	外部入力	○	×
22xx	22010-23287(1024 個)	ネットワーク入力	○	○
3xxx	30010-31287(1024 個)	外部出力	○	×
32xx	32010-33287(1024 個)	ネットワーク出力	○	×
4xxx	40010-40807（640 個）	専用入力（システム）	○	×
5xxx	50010-51007（800 個）	専用出力（システム）	○	×
6xxx	―	―	×	×
7xxx	70010-79997(7992 個)	補助リレー（システム）	○	×
8xxx	80010-80647（512 個）	制御状態信号(システム)	○	×
82xx	82010-82127（96 個）	擬似入力信号(システム)	○	×

コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100（シリアルポート、Ethernet）
参 照	「BscReadIO2」「BscReadIO」「BscWriteIO」

7.6 その他の機能

その他の機能として、以下の関数が使用できます。

BscClose
BscCommand
BscConnect
BscDisConnect
BscDiskFreeSizeGet
BscEnforcedClose
BscGets
BscInBytes
BscOpen
BscOutBytes
BscPuts
BscReConnect
BscReStartJob
BscSetBreak
BscSetCom
BscSetCondBSC
BscSetEServerMode
BscSetEther
BscStatus

■ BscClose

機 能	通信ハンドラを開放します。
書 式	_declspec(dllexport) short APIENTRY BscClose(short nCid);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号
	OUT（戻り） 無し
	リターン値 0　　：正常終了 その他：開放失敗
解 説	**呼び出し条件** 本関数を呼び出す前にBscDisConnect関数で通信回線の切断を行う必要があります。
コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC（シリアルポート、Ethernet） ERC（シリアルポート）
参 照	「BscOpen」「BscDisConnect」

■ BscCommand

機 能	伝送コマンドの送信処理を行います。
書 式	_declspec(dllexport) short APIENTRY BscCommand(short nCid,char *h_buf,char *d_buf,short fforever);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号 *h_buf　ヘッダ文字列ポインタ *d_buf　データ文字列ポインタ fforever　ロボットの応答 0：待ち無し、1：待ち有り
	OUT（戻り） 無し
	リターン値 -1　：送信失敗 その他：正常終了
解 説	**ヘッダ文字列** ヘッダ文字列はヘッダ番号、サブコード番号の順で表わされます。
	データ文字列 データ文字列はデータの文字列とその最後に「￥r（キャリッジリターン)」を付けます。 ＜例＞”サーボオン”コマンドを送信する場合 ヘッダ番号　01 サブコード番号　000 の場合 ヘッダ文字列 01,000 データ文字列 SVON 1\r
コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC（シリアルポート、Ethernet） ERC（シリアルポート）

■ BscConnect

機 能	通信回線を接続します。
書 式	_declspec(dllexport) short APIENTRY BscConnect(short nCid);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号
	OUT（戻り） 無し
	リターン値 0：エラー 1：正常終了
解 説	**呼び出し条件** 本関数を呼び出す前に BscOpen 関数及び BscSetCom 関数（シリアルポート）、BscSetEther 関数（Ethernet）、または BscSetEServerMode 関数（Ethernet Server）で通信回線の設定をしておく必要があります。
コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC（シリアルポート、Ethernet） ERC（シリアルポート）
参 照	「BscOpen」「BscSetCom」（シリアルポート）「BscSetEther」（Ethernet） 「BscDisConnect」

■ BscDisConnect

機 能	通信回線を切断します。
書 式	_declspec(dllexport) short APIENTRY BscDisConnect(short nCid);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号
	OUT（戻り） 無し
	リターン値 0：エラー 1：正常終了
コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC（シリアルポート、Ethernet） ERC（シリアルポート）
参 照	「BscClose」「BscConnect」

■ BscDiskFreeSizeGet

機 能	指定ドライブの空き容量を取得します。
書 式	_declspec(dllexport) short APIENTRY BscDiskFreeSizeGet(short nCid,short dno,long *dsize);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号 dno　ドライブ番号 1:A ～ 26:Z dsize　空き容量のポインタ
	OUT（戻り） dsize　空き容量のポインタ
	リターン値 0：エラー 1：正常終了
コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC（シリアルポート、Ethernet） ERC（シリアルポート）

■ BscEnforcedClose

機 能	強制クローズ処理を行います。
書 式	_declspec(dllexport) short APIENTRY BscEnforcedClose(short nCid);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号
	OUT（戻り） 無し
	リターン値 0　　：正常終了 その他：開放失敗
コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC（シリアルポート、Ethernet） ERC（シリアルポート）
参 照	「BscOpen」　「BscDisConnect」

■ BscGets

機 能	TTY レベルの伝送で文字列を受信します。
書 式	_declspec(dllexport) short APIENTRY BscGets(short nCid,char *bufptr,WORD bsize,WORD *plengets);
引 数	**IN（引き渡し）** nCid　　通信ハンドラの ＩＤ番号 *bufptr　受信文字列ポインタ bsize　　最大文字数 *plengets　受信文字数
	OUT（戻り） *bufptr　受信文字列ポインタ
	リターン値 受信文字数
コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC（シリアルポート、Ethernet） ERC（シリアルポート）
参 照	「BscPuts」「BscInBytes」「BscOutBytes」

■ BscInBytes

機 能	TTY レベルの伝送で受信中の文字数を返します。
書 式	_declspec(dllexport) short APIENTRY BscInBytes(short nCid);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号
	OUT（戻り） 無し
	リターン値 受信文字数
コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC（シリアルポート、Ethernet） ERC（シリアルポート）
参 照	「BscGets」「BscPuts」「BscOutBytes」

■ BscOpen

機 能	通信ハンドラを取得します。
書 式	_declspec(dllexport) short APIENTRY BscOpen(char *path,short mode);
引 数	**IN（引き渡し）** *path　通信カレントディレクトリ格納ポインタ mode　通信タイプ　1 (=0x01)　: シリアルポート 16 (=0x10)　:Ethernet 256 (=0x100):Ethernet Server
	OUT（戻り） 無し
	リターン値 -1　　：取得失敗 -8　　：ライセンスエラー その他：通信ハンドラの ID 番号
解 説	**呼び出し条件** 本関数を呼び出した後、BscSetCom　関数（シリアルポート）または、BscSetEther(Ethernet)、BscSetEServerMode(Ethernet Server)、及び BscConnect 関数を呼び出すことにより、通信を開始することが可能となります。
	通信タイプ 通信タイプは、1 (=0x01): シリアルポート、16 (=0x10):Ethernet、 256 (=0x100):Ethernet Server 機能 を使用できます。それ以外の値は、エラーになります。
コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC（シリアルポート、Ethernet） ERC（シリアルポート）
参 照	「BscClose」「BscSetCom」（シリアルポート）「BscSetEther」（Ethernet） 「BscSetEServerMode」（Ethernet Server）「BscConnect」

■ BscOutBytes

機 能	TTY レベルの伝送で送信中の残り文字数を返します。
書 式	_declspec(dllexport) short APIENTRY BscOutBytes(short nCid);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号
	OUT（戻り） 無し
	リターン値 送信文字数
コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC（シリアルポート、Ethernet） ERC（シリアルポート）
参 照	「BscGets」「BscPuts」「BscInBytes」

■ BscPuts

機 能	TTY レベルの伝送で文字列を送信します。
書 式	_declspec(dllexport) short APIENTRY BscPuts(short nCid,char *bufptr,WORD length);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号 *bufptr　送信文字列ポインタ length　送信文字数
	OUT（戻り） 無し
	リターン値 送信文字数
コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC（シリアルポート、Ethernet） ERC（シリアルポート）
参 照	「BscGets」「BscInBytes」「BscOutBytes」

■ BscReConnect

機 能	通信回線を再接続します。
書 式	_declspec(dllexport) short APIENTRY BscReConnect(short nCid);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号
	OUT（戻り） 無し
	リターン値 0：エラー 1：正常終了
解 説	**呼び出し条件** 本関数を呼び出す前に BscOpen 関数及び BscSetCom 関数（シリアルポート）、BscSetEther 関数（Ethernet）、または BscSetEServerMode 関数（Ethernet Server）で通信回線の設定をしておく必要があります。
コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC（シリアルポート、Ethernet） ERC（シリアルポート）
参 照	「BscOpen」「BscSetCom」（シリアルポート）「BscSetEther」（Ethernet） 「BscDisConnect」

■ BscReStartJob

機 能	再始動をかけます。
書 式	_declspec(dllexport) short APIENTRY BscReStartJob(short nCid);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号
	OUT（戻り） 無し
	リターン値 0　　：正常終了 1　　：カレントジョブ名称指定無し その他：エラーコード
解 説	**呼び出し条件** 本関数を実行する事前に BscSelectJob 関数をコールして、カレントジョブ名の指定をしておく必要があります。 BscHoldOn 関数で始動中のジョブがホールドされ、これを再スタートさせたい場合は、BscHoldOff 関数でホールドを解除し、BscContinueJob 関数をコールして、再スタートをかけて下さい。
コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC（シリアルポート、Ethernet） ERC（シリアルポート）
参 照	「BscContinueJob」「BscSelectJob」

■ BscSetBreak

機 能	伝送を中止します。
書 式	_declspec(dllexport) short APIENTRY BscSetBreak(short nCid,short flg);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号 flg　　強制終了フラグ 0：強制終了無し、1：強制終了有り
	OUT（戻り） 無し
	リターン値 -1：通信ハンドラ異常 0：正常終了
コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC（シリアルポート、Ethernet） ERC（シリアルポート）

■ BscSetCom

機 能	シリアルポートの通信パラメータの設定を行います。
書 式	_declspec(dllexport) short APIENTRY BscSetCom(short nCid, short port,DWORD baud,short parity,short clen,short stp);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号 port　通信ポート番号　1:COM1, 2:COM2, 3:COM3, 4:COM4, •••, 255:COM255 baud　ボーレート　150, 300, 600, 1200, 2400, 4800, 9600, 19200 parity　パリティ　0: なし , 1: 奇数 , 2: 偶数 clen　データ長　7:7 ビット , 8:8 ビット stp　ストップビット　0:1 ビット , 1:1.5 ビット , 2:2 ビット
	OUT（戻り） 無し
	リターン値 0：エラー 1：正常終了
解 説	**呼び出し条件** 本関数を呼び出す前に BscOpen 関数でシリアルポートの通信ハンドラを取得しておく必要があります。また、本関数呼び出し後に BscConnect 関数を使用することにより、通信を行うことが可能となります。
コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC、ERC（シリアルポート）
参 照	「BscOpen」「BscConnect」

■ BscSetCondBSC

機 能	伝送コントロールタイマー・リトライカウンタの値を設定します。
書 式	_declspec(dllexport) short APIENTRY BscSetCondBSC(short nCid,short timerA,short timerB,short rtyR,short rtyW);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号 timerA　タイマー A（コントロールタイマー、ミリセカンド単位） timerB　タイマー B（テキストタイマー、ミリセカンド単位） rtyR　シーケンスリトライカウンタ rtyW　テキストリトライカウンタ
	OUT（戻り） 無し
	リターン値 -1：通信ハンドラ異常 0：正常終了
解 説	**初期値** timerA　10000(msec) timerB　30000(msec) rtyR　3 rtyW　3
	アプリケーションの注意事項 本機能は、パソコンだけの『MOTOCOM32』のパラメータを変えるのに使用されます。 ロボットコントローラの伝送パラメータ（コントロールタイマー、リトライカウンタ）を変える場合は、ロボットコントローラのプログラミングペンダントを使用しなければいけません。
コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC（シリアルポート、Ethernet） ERC（シリアルポート）

■ BscSetEServerMode

機 能	イーサネットサーバ機能による通信のパラメータ設定を行います。
書 式	_declspec(dllexport) short APIENTRY BscSetEServerMode(short nCid, char *IPaddr, short Mode);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号 *IPaddr　通信先の IP アドレス Mode　サーバの通信モードの指定 1：サーバモード、-1：占有モード
	OUT（戻り） 無し
	リターン値 0：エラー 1：正常終了
解 説	**呼び出し条件** 本関数を呼び出す前に BscOpen 関数で通信ハンドラを取得しておく必要があります。また、本関数呼び出し後に BscConnect 関数を使用することにより、通信を行うことが可能となります。
	モード指定 本関数にてイーサネットサーバのモードを指定します。指定可能なモードには以下のものがあります。 ・サーバモード 各コマンド毎にコントローラとアプリケーションとのネットワーク接続を終了させます。これにより、一台のコントローラに対して複数のアプリケーションが同時にネットワーク接続可能となります。 ・占有モード 一つのアプリケーションでコントローラとのネットワーク接続を占有します。ネットワーク接続は BscConnect 関数が呼ばれてから BscDisConnect 関数が呼ばれるまで継続します。 これらのモード間でのネットワーク接続処理の違いはMOTOCOM32内で処理されますので、アプリケーション側で本関数でのモード指定以外に接続処理方法を変える必要はありません。
	制限事項 イーサネットサーバ機能の通信では DCI 機能はサポートしていません。 本関数は YRC1000、YRC1000micro、DX200、DX100、FS100、NX100 に対して伝送を行う場合に有効です。
コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100（Ethernet）
参 照	「BscOpen」「BscConnect」

■ BscSetEther

機 能	イーサネット通信のパラメータの設定を行います。
書 式	_declspec(dllexport) short APIENTRY BscSetEther(short nCid, char *IPaddr, short mode, HWND hWnd);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号 *IPaddr　通信先の IP アドレス mode　実行モード 0：Client、1：Stand alone hWnd　ウィンドウハンドル
	OUT（戻り） 無し
	リターン値 0：エラー 1：正常終了
解 説	**呼び出し条件** 本関数を呼び出す前に BscOpen 関数で通信ハンドラを取得しておく必要があります。また、本関数呼び出し後に BscConnect 関数を使用することにより、通信を行うことが可能となります。
	実行モードと通信先の IP アドレス 使用する通信機能にあった『mode』引数を選びます。その『mode』引数によって、パソコンで動作するアプリケーションが、クライアントまたは、サーバであることが決まります。 <table><tr><th>機能</th><th>Mode(パソコン)</th><th>IPaddr</th></tr><tr><td>Host Control</td><td>0 (Client)</td><td>必ず設定</td></tr><tr><td>DCI</td><td rowspan="2">1 (Server)</td><td rowspan="2">省略可能</td></tr><tr><td>Stand Alone</td></tr></table> パソコンがサーバ（mode＝1）の場合に、IP アドレス（IPadder）に YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC の IP アドレスを設定するかどうかで特定のクライアントサーバかどうかが決まります。 ＜例＞ クライアントの場合 ： 　BscSetEther(nCid, "192.168.10.10", 0, hWnd); 特定のクライアントサーバの場合（必ず IP アドレスを記述します。）： 　BscSetEther(nCid, "192.168.10.10", 1, hWnd); いくつかのクライアントサーバの場合（ＩＰアドレスを記述しません。）： 　BscSetEther(nCid, "", 1, hWnd);
コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC（Ethernet）
参 照	「BscOpen」「BscConnect」

■ BscStatus

機 能	ロボットからのステータスの取得を行います。
書 式	_declspec(dllexport) short APIENTRY BscStatus(short nCid,char *hpt,char *dpt,unsigned short sz,char *rbuf);
引 数	**IN（引き渡し）** nCid　通信ハンドラの ID 番号 *hpt　ヘッダ文字列ポインタ *dpt　送信データ文字列ポインタ sz　送信データ文字列サイズ *rbuf　受信データ文字列ポインタ
	OUT（戻り） *rbuf　受信データ文字列ポインタ
	リターン値 0　：正常終了 その他：エラーコード
コントローラ	YRC1000、YRC1000micro、DX200、DX100、FS100、NX100、XRC、MRC（シリアルポート、Ethernet） ERC（シリアルポート）
参 照	「BscGetStatus」

7.7 伝送関連キーワードに対応した DLL 関数

7.7.1 伝送コマンドに関連する DLL 関数

■ リード・監視系

RALARM	BscGetError BscGetError2 BscGetFirstAlarm BscGetNextAlarm BscIsErrorCode
RPOS	BscIsLoc
RPOSJ	BscIsLoc BscGetPulsePos
RSTATS	BscGetStatus BscIsCycle BscIsServo BscIsTeachMode BscIsPlayMode BscIsRemoteMode BscIsHold BscIsAlarm BscIsError
RJSEQ	BscIsJobName BscIsJobLine BscIsJobStep
RPOSC	BscIsRobotPos BscGetCartPos
JWAIT	BscJobWait
RGROUP	BscGetCtrlGroupDX BscIsCtrlGroupDX BscGetCtrlGroupXrc BscIsCtrlGroupXrc BscIsTaskInfXrc BscGetCtrlGroup BscIsCtrlGroup BscIsTaskInf
RTRQ	BscGetTorque
RMAXTRQ	BscGetMaxTorque
RENCTMP	BscGetEncTmp
RSYSTM	BscGetSysTime

■ リード・データアクセス系

RJDIR	BscFindFirst BscFindFirstMaster BscFindNext BscFindNextMaster
RUFRAME	BscGetUFrame
UPLOAD	BscUpload BscUploadEx
SAVEV	BscGetVarData BscGetVarData2 BscHostGetVarData BscHostGetVarDataM BscGetVarDataEx

■ 操作系

HOLD	BscHoldOn BscHoldOff
RESET	BscReset
CANCEL	BscCancel
MODE	BscSelectMode
CYCLE	BscSelStepCycle BscSelOneCycle BscSelLoopCycle
HLOCK	BscOPLock BscOPUnLock
MDSP	BscMDSP
SVON	BscServoOn BscServoOff
CGROUP	BscSetCtrlGroupDX BscSetCtrlGroupXrc BscSetCtrlGroup
CTASK	BscChangeTask

■ 編集系

DELETE	BscDeleteJob
WUFRAME	BscPutUFrame BscPutUFrameEx2
CVTRJ	BscConvertJobP2R
DOWNLOAD	BscDownload BscDownloadEx
CVTSJ	BscConvertJobR2P
LOADV	BscPutVarData BscPutVarData2 BscHostPutVarData BscHostPutVarDataM BscPutVarDataEx

■ ジョブ選択系

SETMJ	BscSetMasterJob
JSEQ	BscSetLineNumber

■ 起動系

START	BscStartJob BscContinueJob
MOVJ	BscMovj BscMovjEx BscMov BscMovEx BscMovEx2
MOVL	BscMovl BscMovlEx BscMov BscMovEx BscMovEx2
IMOV	BscImov BscImovEx BscImovEx2 BscMov BscMovEx BscMovEx2

PMOVJ	BscPMov BscPMovjEx BscPMov BscPMovEx
PMOVL	BscPMovl BscPMovlEx BscPMov BscPMovEx

■ その他の DLL 関数

BscCommand
BscStatus

7.7.2 DCI 機能に関連する DLL 関数

LOADJ	BscDCILoadSave BscDCILoadSaveOnce
SAVEJ	BscDCILoadSave BscDCILoadSaveOnce
LOADV	BscDCIGetPos BscDCIGetPos2 BscDCIGetVarData BscDCIGetVarDataEx
SAVEV	BscDCIPutPos BscDCIPutPos2 BscDCIPutVarData BscDCIPutVarDataEx

7.7.3 I/O のリード・ライトに関連する DLL 関数

I/O のリード	BscReadIO BscReadIO2
I/O のライト	BscWriteIO BscWriteIO2

7.7.4 パソコン通信ポートに関連する DLL 関数

ポートの接続	BscOpen BscSetCom BscSetEServerMode BscSetEther BscConnect BscReConnect
ポートの切断	BscClose BscDisConnect BscEnforcedClose
伝送パラメータ設定	BscSetCondBSC

7.7.5 その他の DLL 関数

BscDiskFreeSizeGet
BscGets
BscInBytes
BscOutBytes
BscPuts
BscSetBreak

8 付録

8.1 よくあるお問い合わせ

■ USB タイプキーをパソコンへ取り付けた状態でドライバをインストールした場合の対処方法

1. USB タイプキーをパソコンへ取り付けた状態でデバイスマネージャから「不明なデバイス」として登録されているドライブを削除します。
2. ［アプリケーションの追加と削除］でドライバ（Sentinel System Driver）をアンインストールします。
3. **USB タイプキーをパソコンから取り外した状態**でドライバをインストールします。これで正常に動作するようになります。

（インストールについては、「1.4 ハードウェアキーについて」を参照してください。）

■ インストールしているドライバより古いバージョンをインストールした場合の対処方法

この状態ですと、稀に問題が起こる場合があります。［アプリケーションの追加と削除］でドライバをアンインストールしてから、再インストールしてください。

（インストールについては、「1.4 ハードウェアキーについて」を参照してください。）

MOTOCOM32
操作要領書

本製品に関するご不明点及びアフターサービスに関するお問合せは、下記のYASKAWAコンタクトセンタまでご用命ください。

技術・アフターサービスに関するお問い合わせ（YASKAWAコンタクトセンタ）

TEL **0120-502-495**
FAX **0120-394-094**
E-mail **robotcc@yaskawa.co.jp**

●技術相談　●資料請求
月～金（祝日及び当社休業日は除く）
9:00～12:00,13:00～17:00

●アフターサービス
24時間365日

製品・技術情報サイト　e-メカサイト

eメカ　検索　**www.e-mechatronics.com**
安川電機製品の最新情報をご覧いただけます。

海外アフターサービス (Overseas Offices for After-sales Service)

海外アフターサービス拠点については下記URLをご参照ください。

www.e-mechatronics.com/contact/afterservice/robot/index.html

製造･販売

株式会社 安川電機　www.yaskawa.co.jp

この資料の内容についてのお問い合わせは、
当社代理店もしくは、上記の営業部門にお尋ねください。

本製品の最終使用者が軍事関係であったり、用途が兵器などの製造用である場合には、「外国為替及び外国貿易管理法」の定める輸出規制の対象となることがありますので、輸出される際には十分な審査及び必要な輸出手続きをお取りください。

YASKAWA

株式会社 安川電機

資料番号
HW9481253