デンソーロボット

取扱説明書 追補版

Ver. 2.2 の新機能

はじめに

VP-F シリーズロボットの新設に伴い、メインソフトウェアのバージョンを Ver.2.0*から Ver.2.2*に更新しました。この Ver.2.2*の更新に伴って追加および変更された機能について説明します。
なお、オプション品についての新機能については、別冊の「RC7J コントローラ　オプション機器説明書」を参照してください。

本書が扱う範囲

RC7J 型コントローラ

―　　メインソフトウェアバージョン Ver.2.2 に追加・変更された新機能についての説明　　―

目　次

1　USB メモリ機能の追加

USB フラッシュメモリを RC7 コントローラで操作できる機能を追加しました。

動作確認済み USB フラッシュメモリ

メーカ	型式（注）
㈱アイ・オー・データ機器（I-O DATA）	EDP-###M, EDC-###M
ロジテック㈱　　(Logitec)	LMC-###UDA
注：###は、容量を表す数値	

USBメモリご使用上の注意:　USBメモリをフォーマットする場合の「ファイルシステム」は、必ず「FAT」を選択して行ってください。

1.1　［ＵＳＢメモリ　メニュー］の表示機能

操作経路:　[F6 設定]—[F3 USBﾒﾓﾘ]

USBフラッシュメモリを操作するための[USBメモリメニュー]が表示されます。

USBフラッシュメモリを利用するには、次の条件を満たしておく必要があります。
- ロボットコントローラが手動モードであること。
- モータの電源が切れていること。
- プログラムが1つも動作していないこと。

注意: コントローラ電源がON状態でのUSBメモリの接続には、回数制限(１０回)があります。それを超えた場合は、コントローラ電源の再投入が必要となります。

注意: USBフラッシュメモリにアクセス中は、絶対にメモリを抜いたり、ロボットコントローラの電源を切らないでください。(USBフラッシュメモリへのアクセス状態は、各メモリの取扱説明書を参照してください。)

(1) [設定（メイン）]ウィンドウで[F3 USB ﾒﾓﾘ]を押すと、[USB メモリメニュー]が表示されます。

(2) 目的の機能を選択します。それに対応するウィンドウが表示されます。

USB メモリで扱うデータ

USBフラッシュメモリでは、下記の種類のデータを扱います。必要に応じて読み書きするデータを選択してください。

データ種類	内容	備考
プログラム	ソースファイル(PAC, H, PNL) 実行形式ファイル (NIC, MAP) 各種設定ファイル (DAT)	USBメモリに書き込めるのは、使用設定フラグが「使用」になっているファイルのみ。
変数データ	全グローバルデータ 変数使用個数情報	変数データをロボットコントローラに読み込むと、コントローラ内の変数使用個数も切り替わります
I/O データ	I/O 設定データ 拡張ボード用設定データ	
アーム データ	アームパラメータ ツール/ワーク/エリア座標系定義	・他のロボットのアームデータは、絶対読み込まないでください。 ・ツール、ワークデータについてはTOOL、WORKコマンドによる値変更が反映されていないデータをUSBメモリに書き込みます。変更を反映させたい場合は、システムパラメータの保存（操作ガイドP5-179参照）を行った後、書き込んでください。
視覚 データ	視覚機器設定データ	書き込み専用
ログ データ	通信設定 バージョン情報 各種ログデータ	

ロボットコントローラと WINCAPSⅡの間のデータの受け渡し

ロボットコントローラとWINCAPSⅡとのデータの受け渡しを、USBメモリにて行うことができます。

WINCAPSⅡでの操作方法については、WINCAPSⅡガイドの4.3.4節および4.3.5節を参照してください。

USB メモリデータの変更禁止

ロボットコントローラでUSBメモリに格納したデータは、決して変更しないでください。USBメモリデータには、データ破壊のチェックを行ったり、データの正確な読み書きを保証したりするためのチェックコードが入っています。そのため、データを変更すると、読み込むことができなくなります。

1.2　ロボットコントローラへの USB メモリの読み込み

操作経路:　[F6 設定]—[F3 USBメモリ]—[F1 読込み.]

USBフラッシュメモリに格納されているデータを、ロボットコントローラに読み込みます。

(1) [USB メモリメニュー]で[F1 読込み.]を押すと、[読込みファイル選択]ウィンドウが表示されます。

(2) 読み込むデータを選択し、[OK]ボタンを押します。

⚠注意: 他のロボットのアームデータは、絶対に読み込まないでください。ロボットが誤動作し、非常に危険です。

データの読み込みを、開始します。

(3) データの読み込みが完了したら、ロボットコントローラを再起動してください。

プログラムデータを読み込む場合は、次ページの「プログラムデータを読み込む場合の注意事項」を参照してください。

⚠注意: ロボットは、再起動しないと、正常に動作しない場合があります。

プログラムデータを読み込む場合の注意事項

プログラムデータをUSBメモリから読み込む場合は、ロボットコントローラに格納されているプログラムデータおよび実行形式データをすべて削除した後、プログラムデータが読み込まれます。

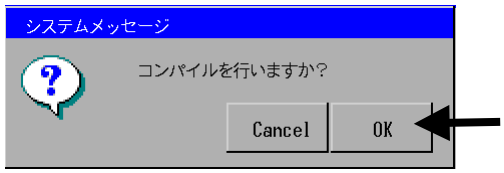
そのため、読み込み対象として選択したデータに、プログラムデータは格納されているものの、実行形式データが格納されていない場合は、読み込んだプログラムデータを下記の手順に従ってコンパイルし、ロードする必要があります。

1) ティーチングペンダントの基本画面で[F1 プログラム]を押し、[プログラム一覧]ウィンドウを表示します。

2) [プログラム一覧]ウィンドウの下部にある[使用設定]ボタンか、メニューバーの[F12 使用設定]を押します。下図の[システムメッセージ]ウィンドウで「全プログラム使用状態」を選択し、すべてのプログラムの使用設定フラグを「使用」にします。[OK]ボタンを押します。

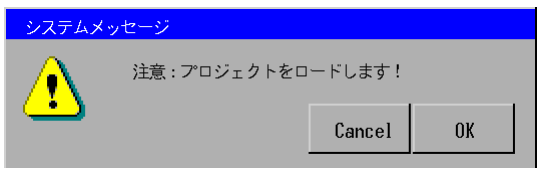
3) 基本画面で[F1 プログラム]-[F6 補助機能]-[F12 コンパイル]を押し、全プログラムをコンパイルします。

システムメッセージが表示されたら、[OK]ボタンを押します。



コンパイルが完了すると、次のようなシステムメッセージが表示されます。

[OK]ボタンを押します。



4) ロボットコントローラを再起動します。

変数データを読み込む場合の注意事項

USBメモリから新しい変数データを読み込むと、ロボットコントローラに格納されている現在の変数が、読み込んだ変数に書き換えられます。

たとえば、現在の整数型変数の個数が50個であり、USBメモリに格納されている整数型変数の個数が30個である場合は、USBメモリからの読み込み処理が完了すると、今までの31番目～50番目の変数は消失することになります。

1.3　ロボットコントローラから USB メモリへのデータ書き込み

操作経路:　[F6 設定]—[F3 USBメモリ]—[F2 書込み.]

ロボットコントローラに格納されているデータを、USBフラッシュメモリに書き込みます（保存されます）。

(1) [USB メモリメニュー]で[F2 書込み.]を押すと、[保存ファイル選択]ウィンドウが表示されます。

(2) フロッピーディスクに書き込むデータの種類を選択して[OK]ボタンを押すと、書き込み処理が開始されます。

書込みに失敗した場合の原因、対策内容を下表に示します。

失敗した時のファイル	原因	対策内容
プログラム	ロードした実行形式データとPACプログラムが一致していない可能性があります。	実行形式データの方が重要な場合は、プログラムは書き込まないようにしてください。PACプログラムと実行形式データを一致させている場合は、次の次のページ“実行形式データとプログラムの関係”を参照してください。
実行形式データ	実行形式データが存在しない可能性があります。	コンパイルをして、実行形式データを作成するか、WINCAPS Ⅱ およびUSBメモリから実行形式データをコントローラに転送してください。

USB メモリにプログラムデータを書き込む場合の注意事項

ロボットコントローラに格納されているプログラムデータを書き込む場合、[プログラム一覧]ウィンドウの「使用」欄（使用設定フラグ）が「使用」になっているプログラムのソースファイルだけが書き込まれます。

「使用設定フラグ」が「使用」になるのは、次の場合です。

1) ロボットコントローラにプロジェクトがロードされている場合（プロジェクト内のすべてのプログラムのソースファイルが「使用」になります)。

2) ティーチングペンダントの[プログラム一覧]ウィンドウでプログラムを選択し、その使用設定フラグを「使用」にした場合。

新たにロードしたプロジェクトの全プログラムのソースファイルをUSBメモリに書き込む場合は、プロジェクトをロードした後でそれらのファイルの使用設定フラグを「使用」から「未使用」に変更したり、プログラムを編集したりしないでください。詳細については、次ページの「実行形式データとプログラムデータの関係」の4)～7)を参照してください。

任意のプログラムのソースファイルをUSBメモリに書き込む場合は、次の操作を実行して使用設定フラグに「使用」または「未使用」を設定し、書き込むプログラムを選択してください。

1) ティーチングペンダントの基本画面で[F1 プログラム]を押し、[プログラム一覧]ウィンドウを表示します。

2) [プログラム一覧]ウィンドウの下部にある[使用設定]ボタンか、メニューバーの[F12 使用設定]を押します。次の[システムメッセージ]ウィンドウで、設定値を確認し、[OK]ボタンを押します。

実行形式データとプログラムデータの関係

実行形式データ（コンパイル済み）とプログラムデータ（プログラムのソースデータ）が矛盾していると、ロボットはプログラムしたとおりに動作しません。
次のような場合は、そのような矛盾が発生します。

1) パソコンからロボットコントローラに、実行形式データ（実行形式ファイルと相互参照テーブル）は送信したものの、そのソースプログラムデータ（インタプリタ表およびプログラムテーブルは除く）を送信しなかった場合。
2) パソコンからロボットコントローラに、ソースプログラムデータは送信したものの、実行形式データを送信しなかった場合。
3) USB メモリからロボットコントローラに、実行形式データは読み込んだものの、そのソースプログラムデータを読み込まなかった場合。
4) コンパイル後にソースプログラムを編集した場合。
5) 新規プロジェクトを作成した場合。
6) ロード済みのプログラムファイルと同名のプログラムを新たに追加した場合。
7) プログラムを削除した場合。
8) プログラムの使用設定フラグ（「使用」、「未使用」）を変更した場合。

注：1)および3)以外の場合は、コンパイルとロードを実行すると、実行形式データとソースプログラムデータが一致します。

1.4　制御ログの USB メモリ保存設定

操作経路:　[F6 設定]—[F3 USBメモリ]—[F12 補助機能.]—[F11 制御ログ]

USBメモリにデータを記録する際に、制御ログデータも記録するかどうかを設定します。

制御ログは、サイズが大きく、設備データのバックアップとしても不要です。そのため、制御ログの保存機能は、制御ログデータを保存する場合にのみ有効にしてください。

(1) [USB メモリメニュー]で[F12 補助機能.]—[F11 制御ログ]を押すと、[制御ログ保存設定]ウィンドウが表示されます。

(2) 「有効」か「無効」を選択し、[OK]を押します。

2　フォルダ機能の追加

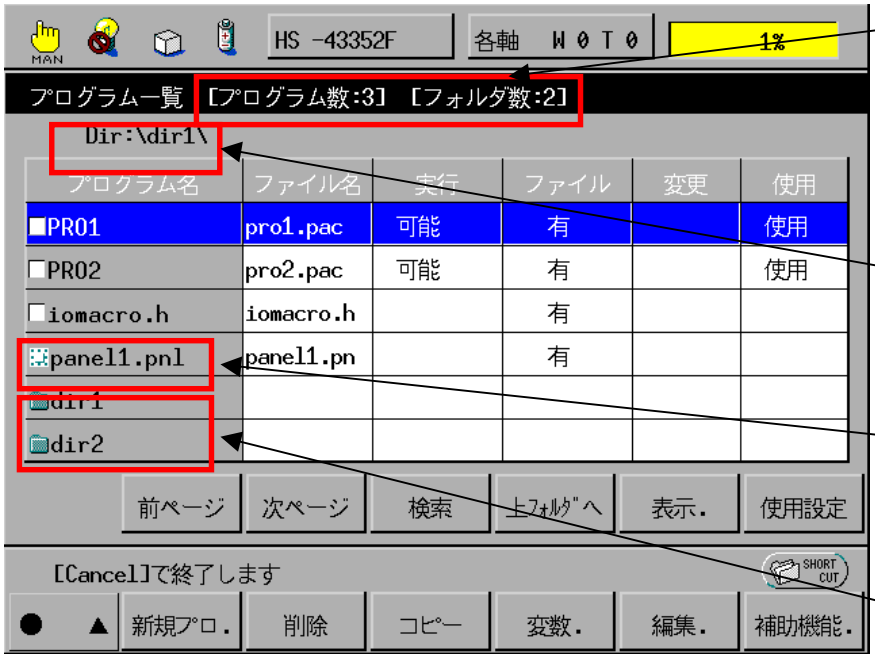
ユーザプログラムを、フォルダを使って階層で管理できるフォルダ機能を追加しました。

2.1　ティーチングペンダントにおける変更(拡張)

フォルダ機能の追加に伴い、ティーチングペンダントは次のように変更(拡張)されています。

プログラム一覧

表示上の変更点（手動モード）



現在開いているフォルダ内のプログラム数と、フォルダ数を表示します。プログラム数にはヘッダファイルも含みます。操作盤ファイルは数に含みません。

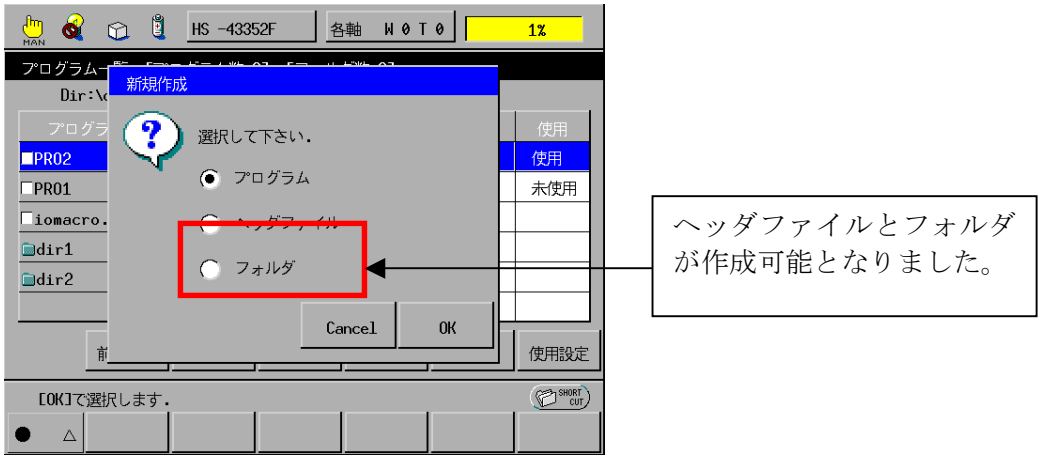
現在開いているフォルダのディレクトリ階層を表します。

操作盤ファイルを表します。操作盤ファイルに関しては、操作盤機能の項参照。

フォルダを表します。

ファンクションキーの機能変更点（手動モード）

①新規プログラム

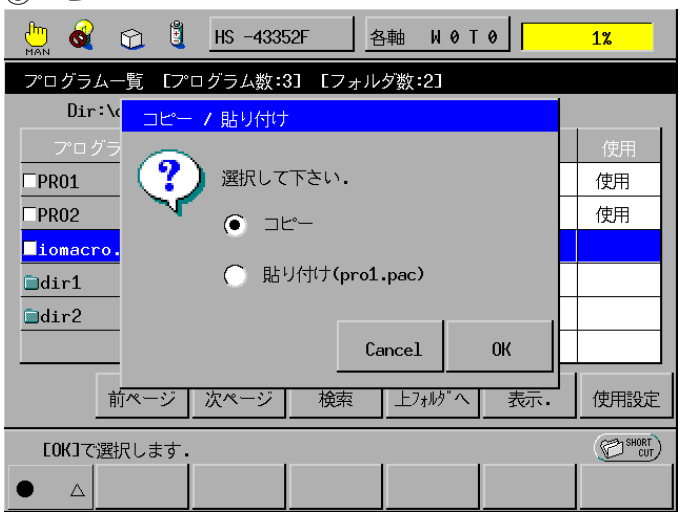


ヘッダファイルとフォルダが作成可能となりました。

○プログラム　「コントローラ内で最大２５６まで作成可能」
○ヘッダファイル　最大３２文字　「コントローラ内で”ヘッダファイル＋操作盤ファイル”で最大２５６まで作成可能。」
○フォルダ最大１６文字「コントローラ内で最大２５６まで作成可能」
「フォルダはルートを除いて最大４階層まで作成可能」

②削除
○PAC プログラム、ヘッダファイル、フォルダ、操作盤ファイルを削除可能です。
フォルダを削除した場合、削除したフォルダ内の要素はすべて削除されます。

③コピー



○コピー：PAC プログラム、ヘッダファイル、フォルダをコピーすることができます。１番最後にコピーしたものが貼り付けの対象となり貼り付けの括弧の中に表示されます。フォルダをコピーした場合、フォルダ内の要素もコピー対象となります。操作盤ファイルはコピーできません。ただし、操作盤ファイルがフォルダに含まれている場合で、フォルダをコピーした場合は操作盤ファイルもコピーされます。
○貼り付け：１番最後にコピーした要素を現在のフォルダ「Dir:で表示されているフォルダ」にコピーします。同名の要素がある場合は、別名でコピーか上書きかを選択します。
貼り付けした PAC プログラムの使用/未使用コンパイルフラグは未使用となります。ただし、上書きされた、PAC プログラムは現在の使用/未使用フラグを保持します。
フォルダを上書きする場合、コピー元フォルダ内にない要素を、コピー先のフォルダが持っている場合、その要素はそのまま残ります。

④検索
○プログラム名から PAC プログラムを検索します。現在のフォルダ「Dir:で表示されているフォルダ」内のみが対象です。ヘッダファイル、操作盤ファイル、フォルダは対象でありません。

⑤上ﾌｫﾙﾀﾞへ
現在の表示されているフォルダから上のフォルダへ移動します。ルートの場合はなにも行いません。

⑥表示
PAC プログラム、ヘッダファイルが選択されている状態で押された場合は、その内容を表示します。フォルダが選択されている状態で押された場合は、選択されているフォルダ内へ移動します。

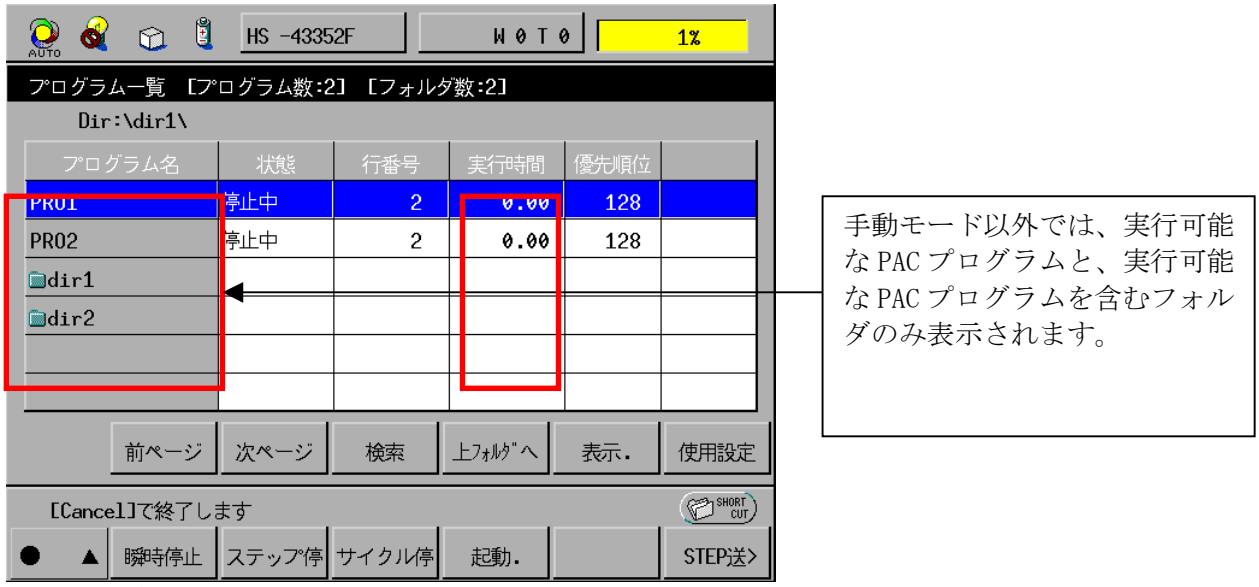
⑦使用設定

○コンパイルフラグ（使用／未使用）を設定します。設定対象は PAC プログラムのみです。使用になっている PAC プログラムはコンパイルの対象となります。
コンパイルフラグの設定には、次の５つがあります。
「このプログラムのみ使用状態に　/　このプログラムのみ未使用状態に」
　カーソルが PAC プログラムを選択している時のみ表示されます。選択された PAC プログラムの現在の状態を反転させます。
「現在フォルダ内プログラムのみ使用状態」
　現在のフォルダ「Dir:で表示されているフォルダ」内で表示されている PAC プログラムをすべて使用状態へ変更します。
「現在フォルダ内プログラムのみ未使用状態」
　現在のフォルダ「Dir:で表示されているフォルダ」内で表示されている PAC プログラムをすべて未使用状態へ変更します。
「全プログラム使用状態」
　コントローラ内 PAC プログラムをすべて使用状態へ変更します。
「全プログラム未使用状態」
　コントローラ内 PAC プログラムをすべて未使用状態へ変更します。

⑧新規 PRJ

ルート以下の PAC プログラム、ヘッダファイル、フォルダ、操作盤ファイルをすべて削除します。

表示上の変更点（手動モード以外）



手動モード以外では、実行可能な PAC プログラムと、実行可能な PAC プログラムを含むフォルダのみ表示されます。

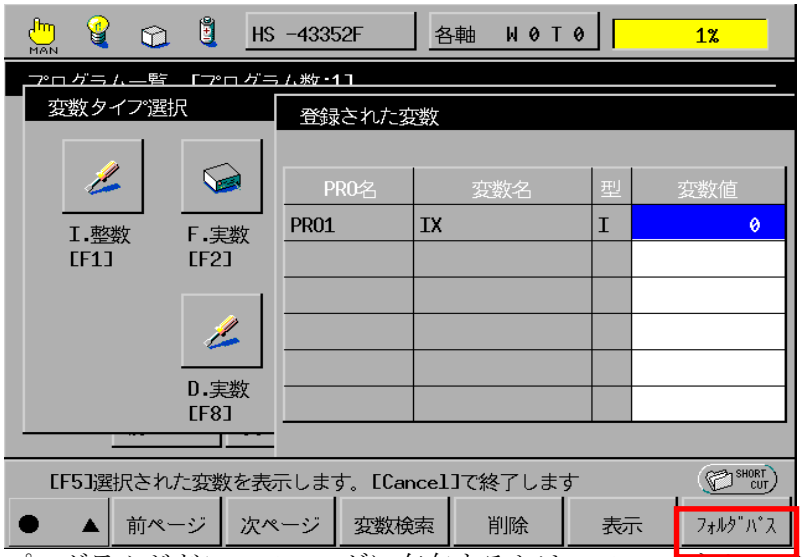
○フォルダ間の移動は手動モードと同様です。手動モードを参照下さい。

注意事項：
フォルダ内のプログラムの実行時間について
「F1　プログラム」－「F６　補助機能」－「F1　PRJ 設定」内のパラメータに
“１３：ｻｲｸﾙﾀｲﾑ算出ｺｰﾄﾞの削除”があります。パラメータの値の意味は
　　０：すべてのプログラムの時間を計る
　　１：I0 から呼び出せるプログラム（ルートで PR0*と言う名称）のみ時間を計る
　　２：すべてのプログラムの時間を計らない
です。
フォルダ内のプログラムの時間を表示したい場合は、“１３：ｻｲｸﾙﾀｲﾑ算出ｺｰﾄﾞの削除”
のパラメータの値を 0 にしてください。

ローカル変数の扱い

○フォルダ機能の追加に伴い、同名のプログラムでもフォルダが違えば作成可能です。
プログラム名、変数名が同じでも、フォルダが違う場合その変数の実体は別のものとして扱われます。

登録変数画面

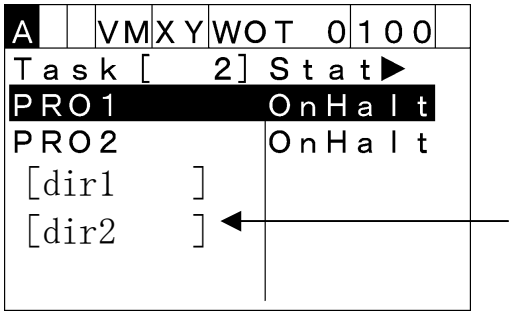


プログラムがどこのフォルダに存在するかは、フォルダパスで確認することができます。

2.2　ミニペンダントにおける変更点

ミニペンダントでフォルダは次のように表されます。

■操作キー：[PRO]



[OK]を押すと、次のような「機能一覧」が表示されます。

・SearchPRO プログラム名の検索　現在のフォルダからプログラムを検索します。
・Display プログラムの表示をします。フォルダを選択して実行した場合はフォルダ内へ移動します。
・UpFolder 上のフォルダへ移動します。すでにルートの場合はなにもしません。

・NowFolder 現在のフォルダを表示します。

2.3　I/O からの起動

I/O からの起動は、ルートにある PAC プログラムのみ可能です。PRO＊という名称であっても、フォルダ内にあるプログラムは起動できません。

2.4　特権タスク

フォルダ機能の追加に伴い特権タスクは、同じ名称であってもフォルダが違えば作成可能です。扱いは今までと同様です。ただし、名称は TSR０～TSR３１と拡張されています。
特権タスクは、PAC プログラムとは別に同時に３２まで起動させることができます。
３２を超えた場合は、エラー「0x7799 特権タスクの最大動作数を超えました」が発生し、
すべてのプログラムが停止します。またこのエラーが発生した場合は特権タスクも停止します。

2.5 WINCAPSⅡにおける変更点

2.5.1 機能概要

フォルダ機能とは、ユーザプログラムをディレクトリ構造を用いて階層管理する機能です。

プログラムを機能や種類別にフォルダに分類することにより構造が分かり易くなり、またフォルダ単位での操作できるためプログラムの転用性も向上させることができます。

以降、フォルダ機能に関して現行 WINCAPSⅡ からの変更点を記載します。

2.5.2 PAC マネージャ

2.5.2.1 メイン画面

PAC マネージャの画面左側に、フォルダ構造を表す「ツリー画面」が表示されました。

ツリーのルートには、プロジェクト名が表示されます。

また右側の「プログラム一覧画面」には、選択されているフォルダ内のプログラムが表示されます。

【ＰＡＣマネージャ 画面】

2.5.2.2 メニュー

フォルダ機能に伴い、追加、変更になったメニュー項目を記載します。

(1) ファイル メニュー

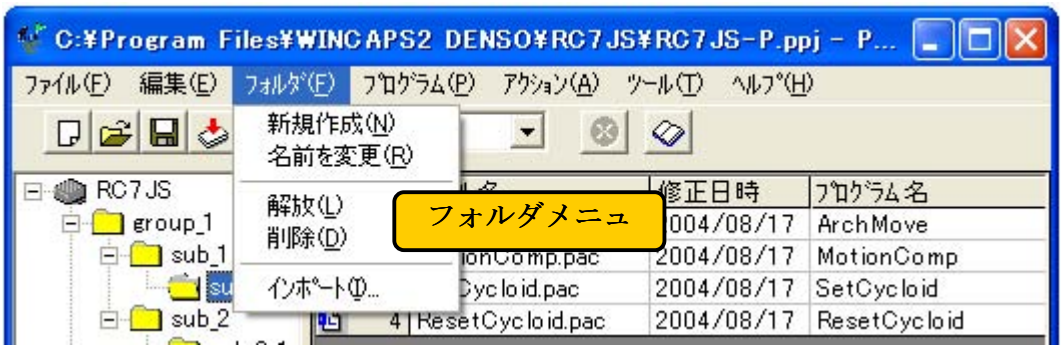
① 実行プログラムの作成

プロジェクト内に操作盤ファイルがある場合は、従来の「実行プログラムの作成」後、「操作盤ファイルのコンパイル」を行います。コンパイル結果は、従来同様にメッセージペインに表示されます。

(2) フォルダ メニュー（新規）

フォルダ構造を扱う専用メニューとして「フォルダ」メニューが追加されました。

ツリー画面の右クリックでも、同一のメニューが表示されます。



【フォルダメニュー】

① 新規作成

選択されているフォルダの下に、新規にフォルダを作成します。

※ 制限事項

作成するフォルダには、次の制限があります。

1. フォルダ名 ：英数字で始まり、最大 16 文字
2. 階層数 ：プロジェクトのルートから数えて、最大４階層
3. フォルダ数 ：最大２５６個

② 名前を変更

選択されているフォルダ名を変更します。

③ 開放

選択されているフォルダ以下をプログラム一覧から削除します。

> **※ 注意事項**
> プロジェクト情報からデータを削除するのみで、ファイル実体はそのまま残ります。

④ 削除

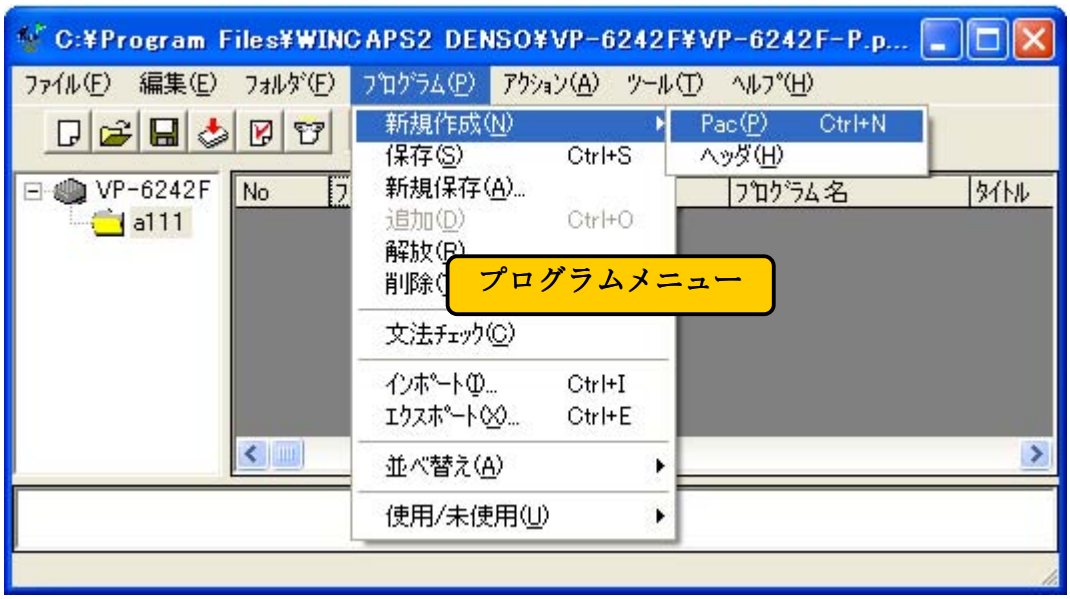
選択されているフォルダ以下をプログラム一覧から削除し、ファイル自体も削除します。

⑤インポート

選択されているフォルダの下に、他のフォルダ以下をコピーしてプログラム一覧に追加します。

> **※ 注意事項**
> インポートするフォルダが、プロジェクトのルートから４階層を超える場合は、メッセージが表示され、４階層までがインポートされます。

(3) プログラム メニュー



【プログラムメニュー】

Ver.2.2 の新機能

① 新規作成

従来の PAC プログラムだけでなく「ヘッダファイル」も作成可能になりました。

② 追加

ルートフォルダ選択時のみ、[追加]メニューは有効になります。

③ 削除（新規メニュー）

選択されているファイルをプログラム一覧から削除し、ファイル自体も削除します。

④ インポート／エクスポート

ヘッダファイル、操作盤ファイルも指定可能になりました。

⑤ 並べ替え

ファイルの種類別に並べ替えを行ないます。

プログラム一覧には、上から [PAC プログラム]→[ヘッダファイル] の順で表示されます。

2.5.3　変数マネージャ

2.5.3.1 フォルダ変数

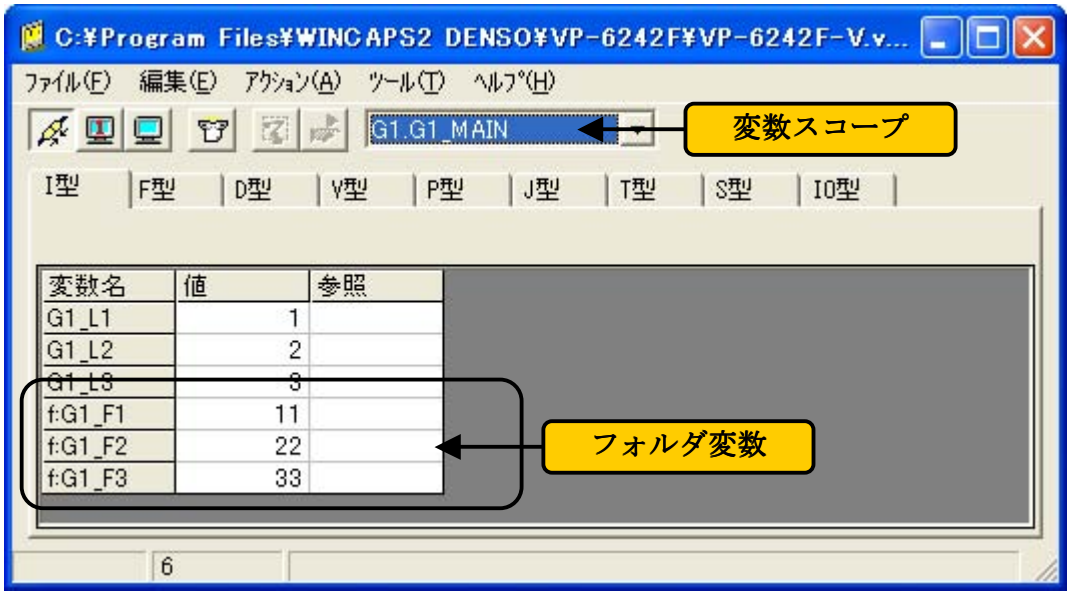
ローカル変数と同様に、「フォルダ変数」をモニタリングすることができます。

操作方法

① コントローラと接続します

② 変数スコープで任意のプログラムを選択します

> **※ 注意事項**
>
> フォルダ変数の場合は、変数名の前に[f:]が表示されます。



【変数マネージャ画面】

2.5.4　その他

2.5.4.1 プログラムのパス表示

フォルダ機能に伴い、プログラム名はフルパスで表示されます。(「プログラムモニタ画面」や「プログラム転送画面」等)

表示形式は、「フォルダ名．プログラム名」となります。

> **※ 注意事項**
>
> PAC マネージャのプログラム一覧画面は、選択フォルダ内のファイルのみ表示するため従来通り、プログラム名のみの表示となります。

3　パワーモジュール情報の表示機能

パワーモジュール情報の表示機能を追加しました。

操作経路：　[F2 アーム]—[F12 保守.]—[F5 PM情報.]

パワーモジュール情報が表示されます。

(1) ［保守機能（アーム)］ウィンドウで[F5 PM 情報.]を押すと、次のような
[パワーモジュール情報]ウィンドウが表示されます。

パワーモジュールの容量を表示します。
ハード設定　「SS、S、M、L、空き、不定」のいずれかが表示されます。
「SS、S、M、L」パワーモジュールの容量を表します。
「空き」パワーモジュールがついていない状態を表します。
「不定」パワーモジュールの状態が認識できない場合に表示されます。
パラメータ設定　「SS、S、M、L」のいずれかが表示されます。
「SS、S、M、L」コントローラ内データの、パワーモジュール容量設定値を表します。

使用しない軸はバックが灰色表示となります。
使用する軸(灰色表示でない)で、ハード設定と、パラメータ設定が一致していない軸は赤色表示されます。

赤色表示されている軸がある場合
コントローラ立ち上げ時と、モータ ON 実行時一致していない軸に対して
0x6149(J1　パワーモジュール容量異常)～0x6150(J8　パワーモジュール容量異常)が発生します。

(2) ［OK］または［Cancel］ボタンを押すと、[保守機能（アーム)］ウィンドウに戻ります。

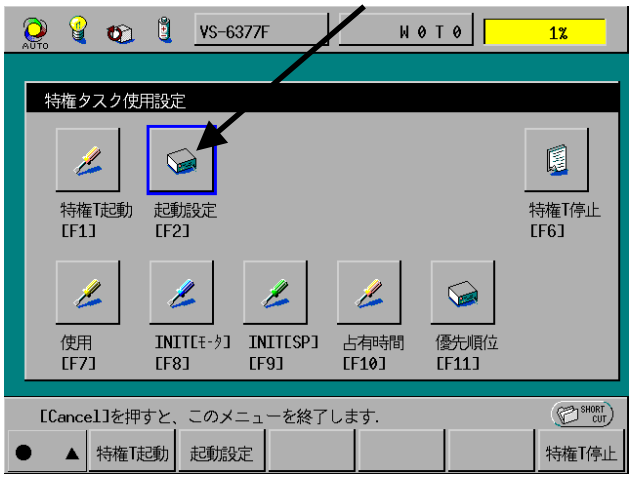
4　特権タスク起動モード設定機能の追加

Ver.2.2 以前のバージョンでは、次の操作実行時、特権タスクが自動で起動されます。
(1) コントローラ電源入りによる起動
(2) 手動モードから自動モードへの切り替えによる起動
(3) 「F8　特権タスク」－「F1　特権 T 起動」による起動

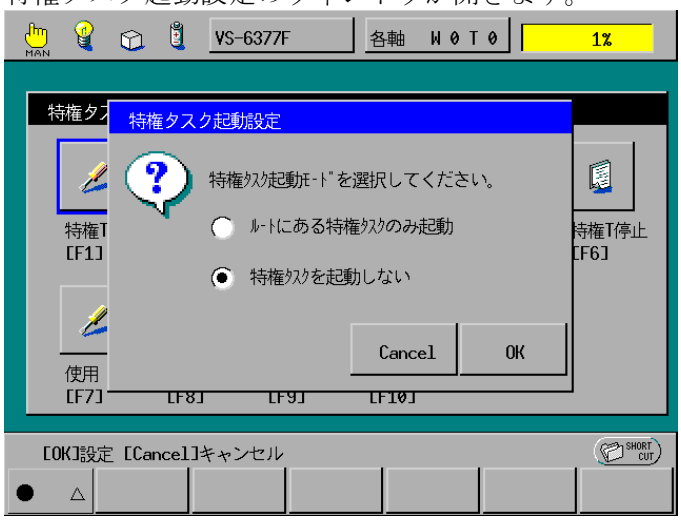
注:　Ver.2.2 以降は、フォルダ機能の追加に伴い、特権タスクをフォルダ内に置くことができます。
ただし、自動で起動するのは、ルートにある特権タスクのみです。

特権タスク起動モード設定で(1)～(3)の操作実行時の特権タスク自動起動を無効にすることができます。

操作経路：「F8　特権タスク」－「F2　起動設定」



特権タスク起動設定のウィンドウが開きます。



○ルートにある特権タスクのみ起動
　上記(1)～(3)の操作実行時、ルートにある特権タスクのみ起動されます。
　フォルダ内の特権タスクは起動されません。
○特権タスクを起動しない
　上記(1)～(3)の操作実行時、特権タスクを起動しません。

注意：いずれの特権タスク起動モードを選択した場合も、プログラム一覧からの特権タスクを選択しての起動、RUNによる特権タスクの起動の場合は通常に起動されます。

5　コマンドの追加と変更

Ver. 2.2から追加・変更されたコマンドについて説明します。

5.1　フォルダー機能関係のコマンド

5.1.1　フォルダー変数宣言

ＦＯＬＤＥＲ　（ステートメント）

機能　外部プログラムからアクセス可能なローカル変数を宣言する。

書式　FOLDER　DEFINT　＜変数名＞［＝＜定数＞］［、＜変数名＞［＝＜定数＞］・・・］
FOLDER　DEFSNG　＜変数名＞［＝＜定数＞］［、＜変数名＞［＝＜定数＞］・・・］
FOLDER　DEFDBL　＜変数名＞［＝＜定数＞］［、＜変数名＞［＝＜定数＞］・・・］
FOLDER　DEFSTR　＜変数名＞［＝＜定数＞］［、＜変数名＞［＝＜定数＞］・・・］
FOLDER　DEFVEC　＜変数名＞［＝＜ベクトル定数＞］［、＜変数名＞［＝＜ベクトル定数＞］・・・］
FOLDER　DEFPOS　＜変数名＞［＝＜ポジション定数＞］［、＜変数名＞［＝＜ポジション定数＞］・・・］
FOLDER　DEFJNT　＜変数名＞［＝＜ポジション定数＞］［、＜変数名＞［＝＜ポジション定数＞］・・・］
FOLDER　DEFTRN　＜変数名＞［＝＜同次変換型定数＞］［、＜変数名＞［＝＜同次変換型定数＞］・・・］
FOLDER　DIM〈変数名〉〈後置子〉［(〈要素数〉［、〈要素数〉［、〈要素数〉]])］[AS〈変数型〉][,〈変数名〉・・・]

説明　ＦＯＬＤＥＲステートメントでローカル変数を宣言すると、ローカル変数を外部のプログラムからＥＸＴＥＲＮステートメントを使用して参照可能となります。

１．プログラム内で同一変数名は定義不可
リスト１ではＡＡの宣言３行とも
変数２重定義エラーになります。

```
PRO1
 FOLDER DEFINT AA
 FOLDER DEFSNG AA
 DEFINT AA
END
```
リスト１

２．フォルダー内で、同一フォルダー変数名は定義不可

```
PRO1
 FOLDER DEFINT AA
END

PRO2
 FOLDER DEFSNG AA
END
```
リスト２

```
PRO1
 FOLDER DEFINT AA
END

PRO2
 FOLDER DEFINT AA
END
```
リスト３

リスト２は同一フォルダー内でＡＡフォルダー変数を別のプログラムで定義しているのでエラーになります。
リスト３は別々のフォルダーで同じＡＡフォルダー変数を定義可能（異なる変数となる）

３．別プログラムでフォルダー変数名とローカル変数名同一の時
宣言可能です。
ＰＲＯ１のＡＡとＰＲＯ２のＡＡは別変数になります。

```
ＰＲＯ１
 ＦＯＬＤＥＲ　ＤＥＦＩＮＴ　ＡＡ
ＥＮＤ

ＰＲＯ２
 ＤＥＦＩＮＴ　ＡＡ
ＥＮＤ
```

リスト４

ペンダントでプログラム編集や、WINCAPSⅡから＜マップ／実行プログラム＞をコントローラに送信すると FOLDER 変数はクリアされます。

関連項目

ＥＸＴＥＲＮ

用例

１．同一フォルダー内に下記３つのプログラムが存在する場合。

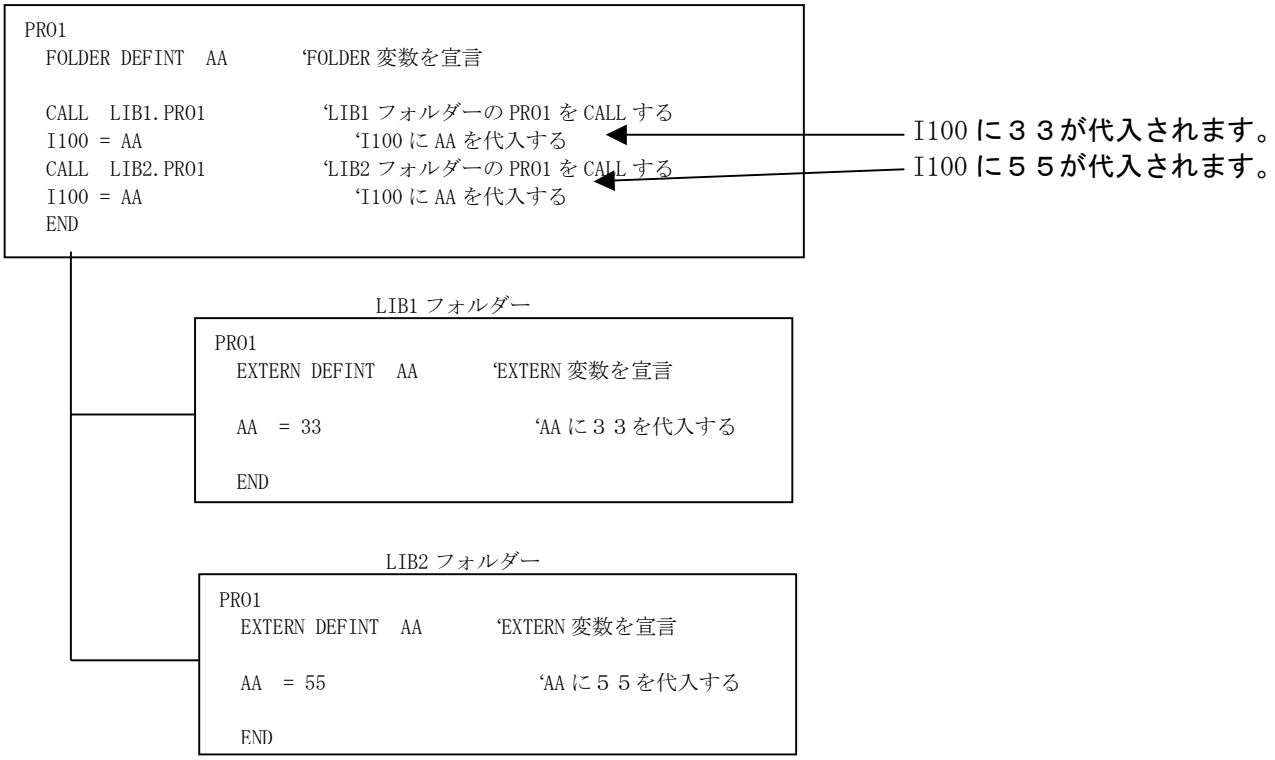
PRO1 を実行後 PRO3 を実行すると“STOP”（プログラム終了）します。

PRO2 を実行後 PRO3 を実行すると“HOLD“STOP””（プログラムを一時停止）します。

３つのプログラムで宣言されている AA 変数は３つとも同じ値になります。

```
PRO1
  FOLDER DEFINT  AA        'FOLDER 変数を宣言

  AA =1                          'AA 変数に１を代入
END


PRO2
  EXTERN DEFINT  AA        'EXTERN 変数を宣言

  AA=2                           'AA 変数に２を代入
END

PRO3
 EXTERN DEFINT  AA         'EXTERN 変数を宣言

 SELECT  CASE  AA          'AA の値が CASE 分と一致
                                'した場合そのコマンドが
                                '実行されます。
 CASE  1                      'AA が１の場合
     STOP                     'プログラムを終了します
 CASE  2                      'AA が２の場合
     HOLD  “STOP”           'プログラム実行を一時停止します。
 END SELECT                   '複数条件判断文の終了を宣言します。
END
```

２．下記のフォルダー構造にプログラムが存在する場合



３．FOLDER 変数で初期値を宣言した場合

(1) PRO1 実行後 PRO2 を実行した場合
I1 には３が入り、I2 にも３が入る。
(2) PRO2 実行後 PRO1 を実行した場合
I1 には３が入り、I2 には０が入る
（注）PRO1 を実行しないと初期値は代入されません。

```
PRO1
  FOLDER DEFINT AA =3
  I1 = AA
END

PRO2
  EXTERN DEFINT AA
  I2 = AA
END
```

5.1.2 ＥＸＴＥＲＮ変数宣言

ＥＸＴＥＲＮ （ステートメント）

機能

書式 EXTERN DEFINT ＜変数名＞［、＜変数名＞・・・］
EXTERN DEFSNG ＜変数名＞［、＜変数名＞・・・］
EXTERN DEFDBL ＜変数名＞［、＜変数名＞・・・］
EXTERN DEFSTR ＜変数名＞［、＜変数名＞・・・］
EXTERN DEFVEC ＜変数名＞［、＜変数名＞・・・］
EXTERN DEFPOS ＜変数名＞［、＜変数名＞・・・］
EXTERN DEFJNT ＜変数名＞［、＜変数名＞・・・］
EXTERN DEFTRN ＜変数名＞［、＜変数名＞・・・］
EXTERN DIM〈変数名〉〈後置子〉[(〈要素数〉[、〈要素数〉[、〈要素数〉]])][AS〈変数型〉][,〈変数名〉・・・]

説明

他のプログラムで定義されている FOLDER 変数を参照する事を宣言する。

FOLDER 変数探索方法

１．EXTERN が宣言されている PAC が存在するフォルダーに FOLDER 変数が存在するか探索する。
FOLDER 変数が存在する場合は探索終了。
存在しない場合は２．へ進む
２．１つ上のフォルダーに移り FOLDER 変数が存在するか探索する。
FOLDER 変数が存在する場合は探索終了。
存在しない場合は２．を繰り返す。
フォルダーのトップを探索しても FOLDER 変数が存在しない場合はコンパイルエラーになる。

```
ＰＲＯ１
 EXTERN  DEFINT  AA
 DEFINT  AA
ＥＮＤ
```
リスト１

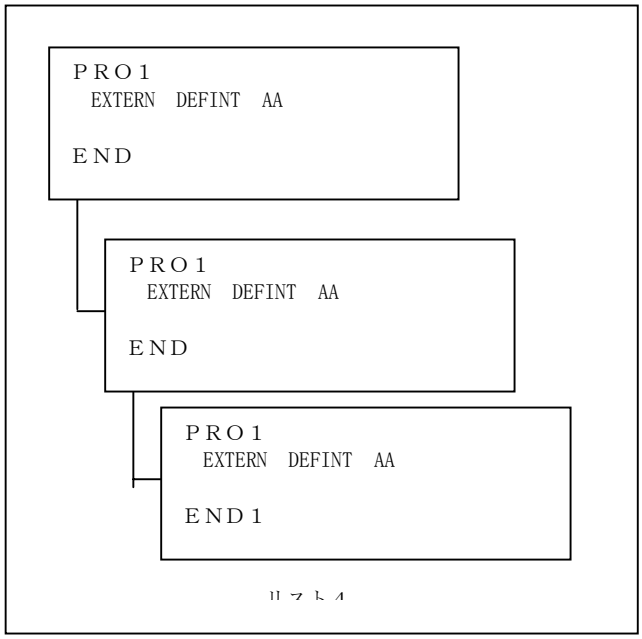
リスト１は同じ変数名が PAC プログラムの中で宣言されているのでコンパイルエラーになります。

```
ＰＲＯ１
 EXTERN  DEFINT  AA
 EXTERN  DEFINT  AA
ＥＮＤ
```
リスト２

リスト２は同じ EXTERN 変数名が同一の PAC ファイルに宣言されているのでコンパイルエラーになりす。

```
ＰＲＯ１
 EXTERN  DEFINT  AA
ＥＮＤ

ＰＲＯ２
 EXTERN  DEFINT  AA
ＥＮＤ
```
リスト３

リスト３で別の PAC ファイルに同じ EXTERN 変数が宣言されている場合は宣言可能です。



リスト４

リスト４で別のフォルダーで同じ EXTERN 宣言されている場合は宣言可能です。

ペンダントでプログラム編集や、WINCAPSⅡから＜マップ／実行プログラム＞をコントローラに送信すると EXTERN 変数はクリアされます。

関連項目

ＦＯＬＤＥＲ

用例

１．同一フォルダー内に下記３つのプログラムが存在する場合。

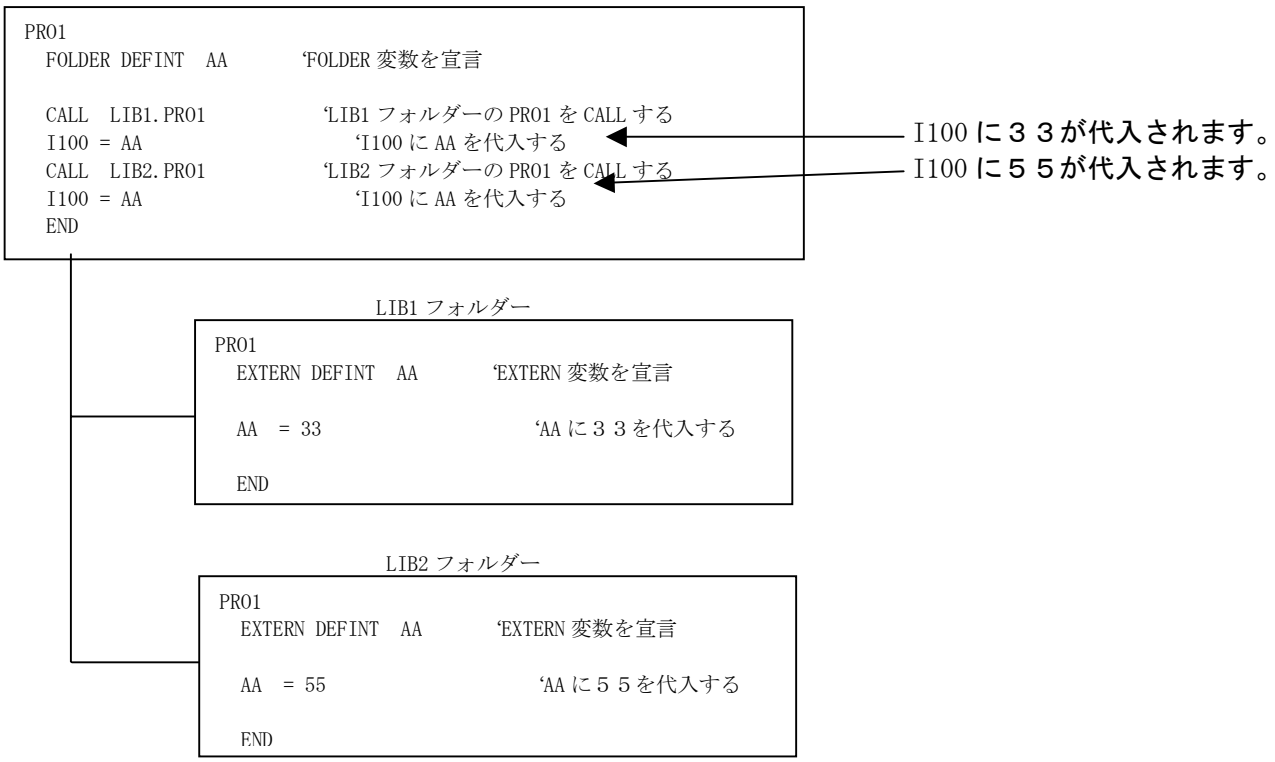
```
PRO1
  FOLDER DEFINT  AA        'FOLDER 変数を宣言

  AA =1                        'AA 変数に１を代入
END

PRO2
  EXTERN DEFINT  AA        'EXTERN 変数を宣言

  AA=2                         'AA 変数に２を代入
END

PRO3
 EXTERN DEFINT  AA         'EXTERN 変数を宣言

 SELECT  CASE  AA          'AA の値が CASE 分と一致
                               'した場合そのコマンドが
                               '実行されます。
 CASE  1                     'AA が１の場合
     STOP                    'プログラムを終了します
 CASE  2                     'AA が２の場合
     HOLD  "STOP"          'プログラム実行を一時停止します。
 END SELECT                  '複数条件判断文の終了を宣言します。
```

PRO1 を実行後 PRO3 を実行すると“STOP”（プログラム終了）します。

PRO2 を実行後 PRO3 を実行すると“HOLD “STOP””（プログラムを一時停止）します。

３つのプログラムで宣言されている AA 変数は３つとも同じ値になります。

２．下記のフォルダー構造にプログラムが存在する場合



３．FOLDER 変数で初期値を宣言した変数を EXTERN で参照する場合

```
PRO1
  FOLDER DEFINT AA =3
  I1 = AA
END

PRO2
  EXTERN DEFINT AA
  I2 = AA
END
```

PRO1 実行後 PRO2 を実行した場合

I1 には３が入り、I2 にも３が入る。

PRO2 実行後 PRO1 を実行した場合

I1 には３が入り、I2 には０が入る

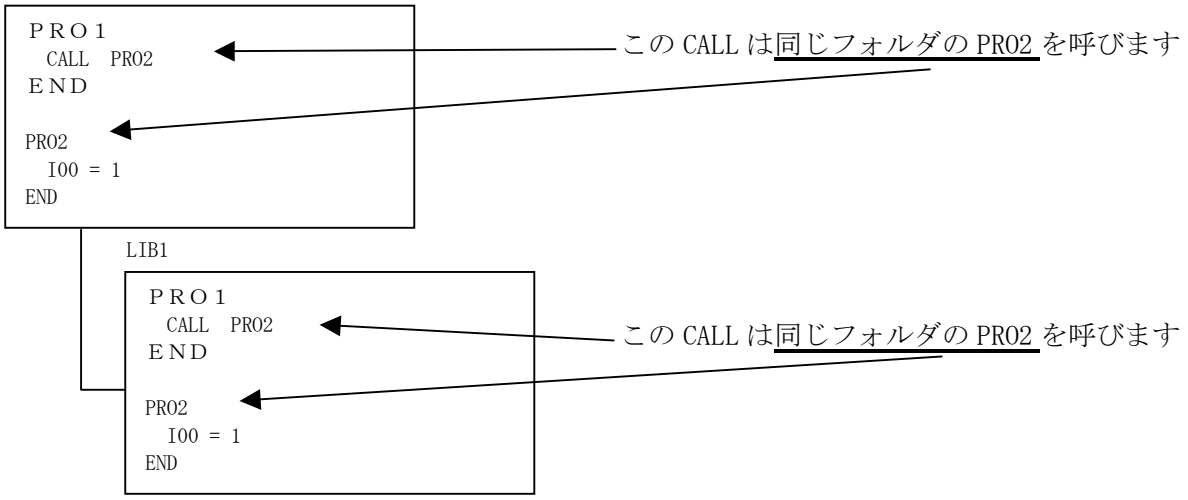
（注）PRO1 を実行しないと初期値は代入されません。

5.1.3　フォルダー構造でのプログラムの呼び出し方

１．CALL、RUN ステートメント

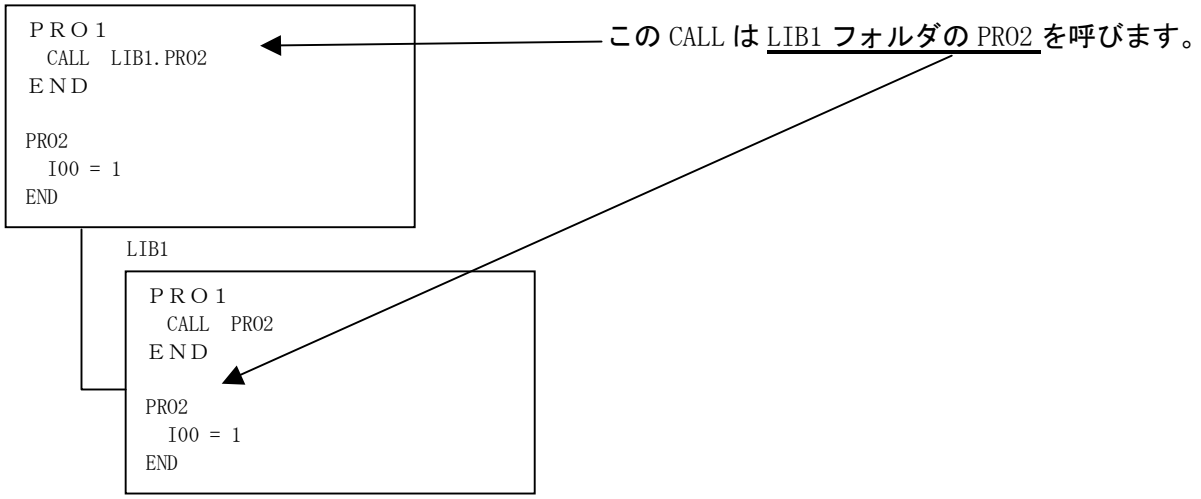
（１） プログラム名のみ時

呼び出す PAC があるフォルダー内のプログラムを呼び出します。



（２） フォルダ名を含めたプログラム名の時

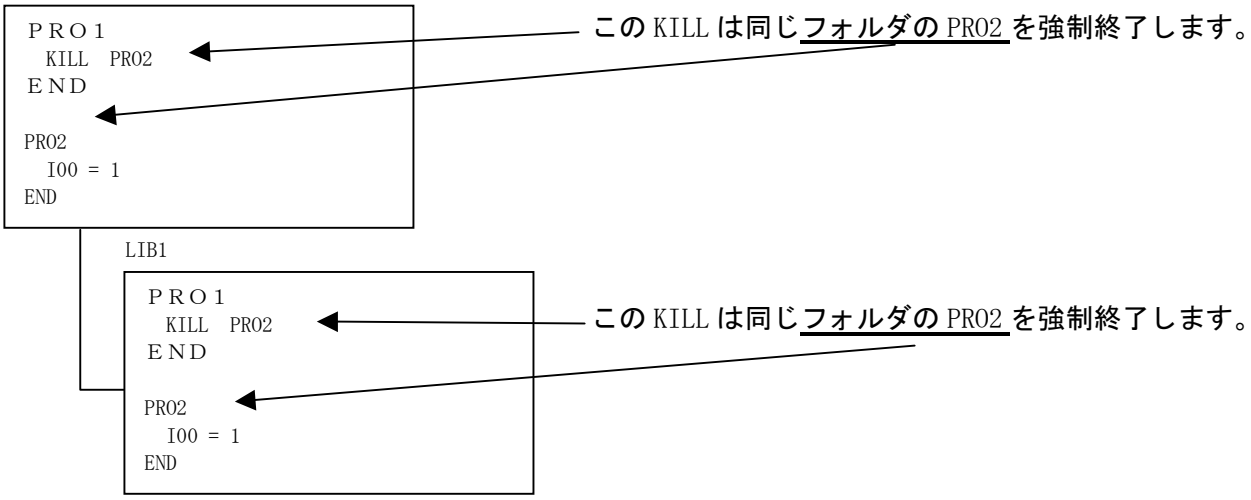
プログラムの指定方法は
CALL　（フォルダ名）.（プログラム名）　になります。
呼び出せるフォルダーの階層は１つしたのフォルダーまでです。
CALL　（フォルダ名）.（フォルダ名）.（プログラム名）はコンパイルエラーになります。



2. KILL、SUSPEND、STATUS ステートメント

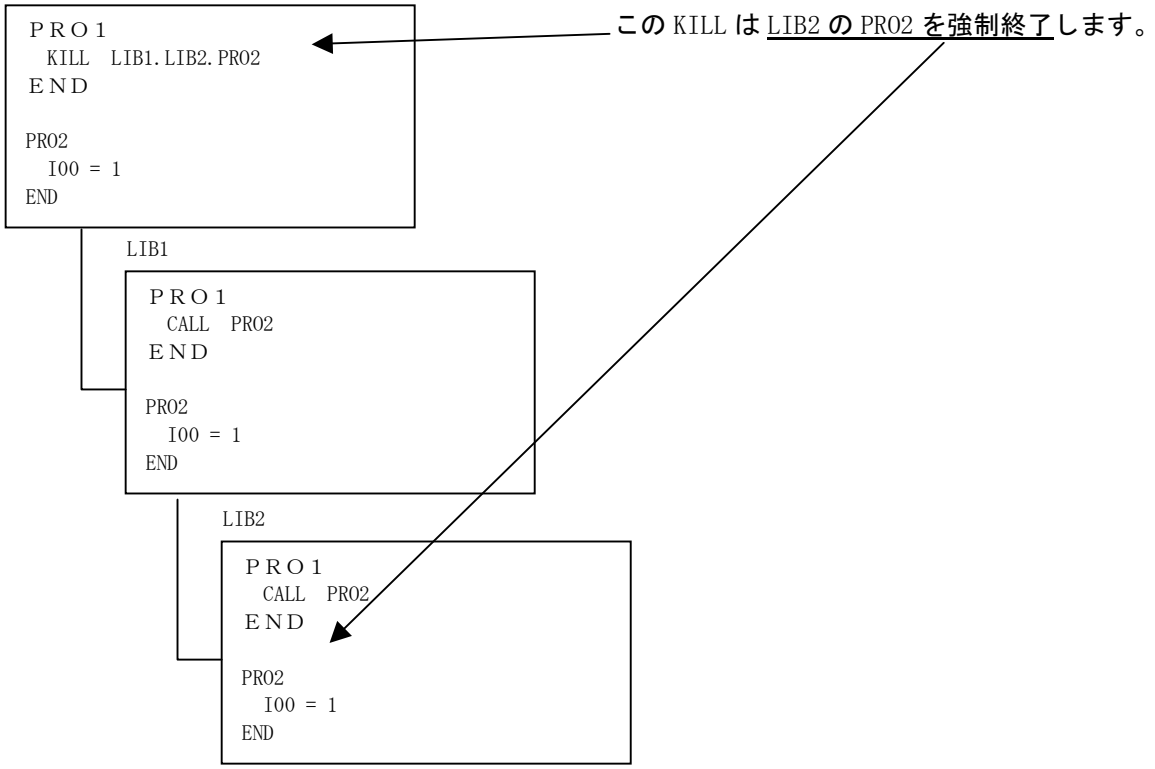
（１）プログラム名のみ時

呼び出す PAC があるフォルダー内のプログラムを処理します。。



（２）フォルダ名を含めたプログラム名の時

KILL （フォルダ名).（プログラム名）で指定します。
複数のフォルダの指定が可能です。 例）KILL （フォルダ名).（フォルダ名).（プログラム名）

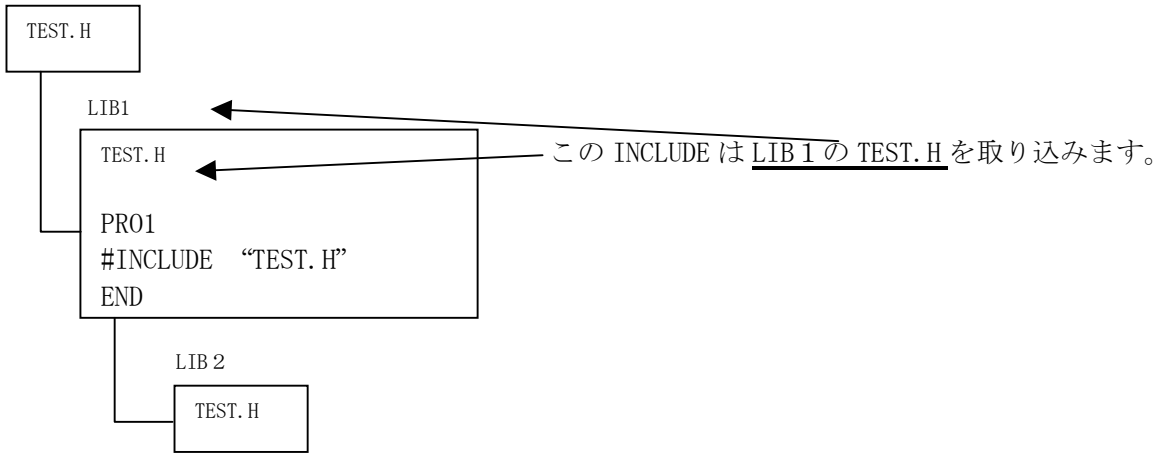


5.1.4 INCLUDE プリプロセッサステートメント

注意：フォルダ間の区切りを表す'¥'の表示は、表示系によって'＼'で表示されます。
例 ペンダントは'＼'表示 ミニペンダントは'¥'表示
これは表示上だけの違いであり、内部的には同じものとして扱われます。

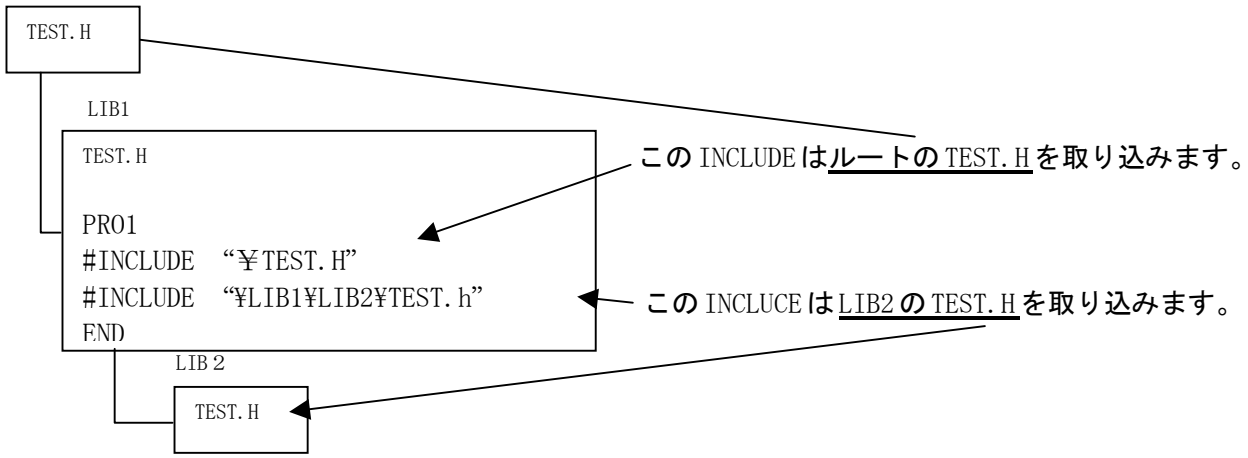
１．＃INCLUDE ＜ファイル名＞
システムフォルダを探します。

２．＃INCLUDE “ファイル名”
INCLUDE 文の記載してある PAC の存在するフォルダ内を探します。



３．＃INCLUDE “￥（フォルダパス）￥ファイル名”

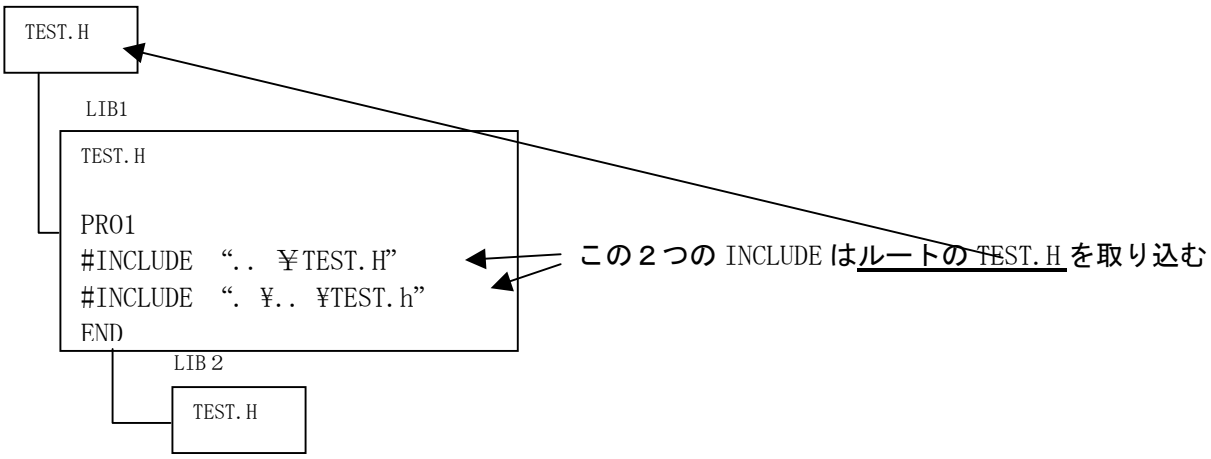
ルートからのフォルダパス先のフォルダ内を探します。



４．＃INCLUDE　“.. ￥ファイル名”　または　＃INCLUDE　“. ￥ファイル名”

.. ￥　　１つ上のフォルダ内を探します。
. ￥　　INCLUDE 文の記載してある PAC のフォルダ内を探します。

例）＃INCLUDE　“. ￥.. ￥TEST. H”　１つ上の TEST. H を探す。



5.2　その他のコマンド

GETLANGUAGE(関数)

機能
現在の言語設定を取得します。

書式
GETLANGUAGE

説明
現在の言語設定を取得します。０：日本語　１：英語を表します。

用例
defint li
li=GETLANGUAGE

PrintWarning(ステートメント)

機能
ペンダント警告メッセージ表示部分へメッセージを表示します。

書式
PRINTWARNING ＜表示メッセージ＞ ［, ＜メッセージ色＞］

説明
ペンダント警告メッセージ表示部分へメッセージを表示します。
表示メッセージの最大長は半角で５５文字までです。それ以上は切り捨てられます。
メッセージ色　０：黒　１：青　２：緑　３：赤　その他：黒　省略した場合も黒です。
メッセージのみの表示でありエラーログには記録されません。
ペンダントでのみの機能です。

用例
printWarning “Hello World!”,1

5.3 変更されたコマンド

Ver. 2.2 から内容変更のあるコマンドについて説明します。

【関連取扱説明書】:「プログラミングマニュアルⅠ、第 13 章　入出力制御文」

INPUT（ステートメント）【SLIM 準拠】

機能　RS232C および Ethernet ポートからのデータを得ます。

書式　INPUT [#<回線番号>,]<変数名>[,WTIME=<タイムアウト時間> [,RVAL=<格納変数>]]

[,<変数名>[,WTIME=<タイムアウト時間> [,RVAL=<格納変数>]]…]

説明　RS232C および Ethernet ポートに受信したデータを、<変数名>で指定した変数に格納します。

<回線番号>には、使用する RS232C および Ethernet ポートの回線番号を指定します。<回線番号>を省略するとデフォルト値の ch2 となります。ch1 はティーチングペンダント用に使用するため指定できません。(「2.4.1　回線番号」参照)

複数の情報を、それと同数の変数で受ける場合は、個々の変数間にカンマ(,)を付けます。

デリミタ（CR または CR+LF）の直前までのすべてを、指定した変数に格納します。デリミタは、変数には読み込まれません。

ボーレートは、システムパラメータで指定します。

＜**タイムアウト時間**＞　Ver2.2以降

変数毎にタイムアウト時間を指定できます。入力データがない場合、指定時間を経過した後、現在の変数に対しての入力待ちを終了し、次に制御が移ります。単位はmsですが、実際の遅延時間は1/60s単位で増減します。

＜**格納変数**＞　Ver2.2以降

＜タイムアウト時間＞の設定と対で使います。入力データを受けて正常に抜けた場合TRUE(1)が入り、タイムアウトで抜けたときは指定した変数にFALSE(0)が入ります。

注意①：	データの受信に先立って、受信データの入力バッファに残っているデータをクリアするために、FLUSHコマンドを実行してください。
②：	入力したデータが、代入する変数の型が表せる値の最大値を超えている場合、結果はその型の最大値となります。

関連項目　FLUSH、PRINT、WRITE、

用例

```
DIM li1 As Integer
defint li
DEFSTR ls1, ls2, ls3, ls4
INPUT #1, ls1                       'ch2 からのデータを ls1 に書きます。
INPUT #1, li1, ls2, ls3             'ch2 からのデータを li1, ls2, ls3 に書きます。
INPUT ls4                           'ch2 からのデータを ls4 に書きます。
INPUT ls4 ,WTIME=100,RVAL=li    '入力がない場合 100msec 時間後次のコマン
                                    'ドへ処理が移ります。li にはタイムアウト
                                    'で抜けたことを表す 0 が入ります。
```

LINEINPUT（ステートメント）

機能

RS232C および Ethernet ポートからデリミタまでのデータを読み込み、文字列型変数に代入します。

書式

LINEINPUT [#<回線番号>,] <文字列型変数名>[,WTIME=<タイムアウト時間> [,RVAL=<格納変数>]]

説明

RS232C および Ethernet ポートに受信したデータから、デリミタ（CR または CR+LF）の直前までのすべての文字を、<文字列型変数名>で指定した変数に格納します。デリミタは、変数には読み込まれません。

<回線番号>には、使用する RS232C および Ethernet ポートの回線番号を指定します。<回線番号>を省略するとデフォルト値の ch2 となります。ch1 はティーチングペンダント用に使用するため指定できません。（「2.4.1　回線番号」参照）

<文字列変数名>には、文字列型変数の名前を指定します。

＜タイムアウト時間＞　Ver2.2以降

指定時間内にデリミタの入力がない場合、lineinput文の実行を終了し、次のコマンドに制御が移ります。単位はmsですが、実際の遅延時間は1/60s単位で増減します。

＜格納変数＞　Ver2.2以降

＜タイムアウト時間＞の設定と対で使います。入力データを受けて正常に抜けた場合TRUE(1)が入り、タイムアウトで抜けたときは指定した変数にFALSE(0)が入ります。

用例

```
DEFSTR ls1,ls2,ls3,ls4
DEFINT li
LINEINPUT #0, ls1           'ポート 1 から文字列を受信し、ls1 へ代入します。
LINEINPUT #1, ls2           'ポート 2 から文字列を受信し、ls2 へ代入します。
LINEINPUT ls3               'ポート 2 から文字列を受信し、ls3 へ代入します。
LINEINPUT #1,ls4,WTIME=100,RVAL=li    'デリミタまでの入力がない場合、100msec
                                      '時間後次のコマンドへ処理が移ります。
                                      'li にはタイムアウトで抜けたことを表す
                                      '0 が入ります。
```

inputb

機能

１バイトデータを RS-232C および Ethernet ポートから入力します。

書式

inputb [#＜**ポート番号**＞,]＜**入力データ格納 I 変数**＞[,WTIME=<タイムアウト時間>[,RVAL=<格納変数>]]

＜**ポート番号**＞　入力ポート番号
（１：コントローラRS-232Cポート、-1：μVisionボードRS-232Cポート
4～7：Ethernetサーバポート、8～15：Ethernetクライアントポート）
省略時　1：コントローラRS-232Cポート

＜**入力データ格納I変数**＞
入力データが格納されるI型変数の番号

＜**タイムアウト時間**＞　Ver2.2以降
入力データがない場合、指定時間を経過した後にinputb文の実行を終了し、次のコマンドに制御が移ります。単位はmsですが、実際の遅延時間は1/60s単位で増減します。

＜**格納変数**＞　Ver2.2以降
＜タイムアウト時間＞の設定と対で使います。入力データを受けて正常に抜けた場合TRUE(1)が入り、タイムアウトで抜けたときは指定した変数にFALSE(0)が入ります。

説明

＜**入力データ格納 I 変数**＞で指定された I 型変数に、ポートから入力したデータを格納します。

注：　タイムアウト時間を設定せずに、このコマンドでデータ入力を行う場合、データがポートに存在しない場合、無限待ちとなります。データの有無の確認には com_state コマンドを使用してください。

用例

```
'!TITLE "<ﾀｲﾄﾙ>"
PROGRAM sample
        defint li
        ・
        ・
        ・
        inputb #1,I10       'I10 に RS-232C ポートから入力したデータを格納
        inputb #1,I10,WTIME=100,RVAL=li '入力がない場合 100msec 時間後次のコマン
                                        'ドへ処理が移ります。li にはタイムアウト
                                        'で抜けたことを表す 0 が入ります。
        ・
        ・
        ・
end
```

linputb [Ver.1.5 以降]

機能

複数バイトデータを RS-232C および Ethernet ポートから入力します。

書式

linputb [#＜**ポート番号**＞,]＜**格納配列先頭要素**＞,＜**入力バイト数**＞[,WTIME=<タイムアウト時間> [,RVAL=<格納変数>]]

＜**ポート番号**＞　入力ポート番号
（１：コントローラRS-232Cポート、-1：μVisionボードRS-232Cポート
4～7：Ethernetサーバポート、8～15：Ethernetクライアントポート）
省略時　1：コントローラRS-232Cポート

＜**格納配列先頭要素**＞
入力データが格納される配列の先頭要素

＜**入力バイト数**＞
入力するデータのバイト数

＜**タイムアウト時間**＞　Ver2.2以降
入力バイト数分の入力データがない場合、指定時間を経過した後にlinputb文の実行を終了し、次のコマンドに制御が移ります。単位はmsですが、実際の遅延時間は1/60s単位で増減します。

＜**格納変数**＞　Ver2.2以降
＜タイムアウト時間＞の設定と対で使います。入力データを受けて正常に抜けた場合TRUE(1)が入り、タイムアウトで抜けたときは指定した変数にFALSE(0)が入ります。

説明

配列の指定要素から、指定入力バイト数分だけ指定ポートからデータを入力します。

注：タイムアウト時間を設定せずに、このコマンドでデータ入力を行う場合、＜入力バイト数＞分のデータがポートに存在しない場合、無限待ちとなります。データの有無の確認には com_state コマンドを使用してください。

用例

```
'!TITLE "<ﾀｲﾄﾙ>"
PROGRAM sample
        defint li
        ・
        ・
        linputb #1,I64, 30      'I64 から I93 までにデータを連続して RS-232C ポート
                                'から入力
linputb #1,I64, 30,WTIME=100,RVAL=li '＜入力バイト数＞分の入力がない場合、100msec
                                '時間後次のコマンドへ処理が移ります。li には
                                'タイムアウトで抜けたことを表す 0 が入ります。
        ・
        ・
end
```

6　エラーコード表の変更

【関連取扱説明書】：「エラーコード表」

6.1　変更されたエラーコード

Ver.2.2 から変更されたエラーコードを下表に示します。

ｺｰﾄﾞ	メッセージ	Level	説明	復帰処置
1201	通信準備中(コネクション未確立)	4	DeviceNet モジュールは正常に動作しており、マスタデバイスとの明示的コネクションは確立していますが、I/O コネクションは確立していません。	マスタデバイスから、コネクションを確立させてください。 電源立上時にこのエラーが発生した後、最終的にネットワークが確立する場合は、ネットワーク異常検出待ち時間を長くしてください。
1203	通信準備中（通信アイドル状態）	4	DeviceNet モジュールは正常に動作していますが、規定時間内にマスタデバイスから空のデータしか受取れない状態です。	マスタデバイスから出力される、I/O データの内容を見直ししてください。 電源立上時にこのエラーが発生した後、最終的にネットワークが確立する場合は、ネットワーク異常検出待ち時間を長くしてください。
1204	通信準備中（I/O タイムアウト）	4	DeviceNet モジュールは正常に動作していますが、規定時間内にマスタデバイスからデータが受取れない状態です。	ネットワークケーブルの断線・コネクタの緩みがないか・ケーブル長は適切か・終端抵抗の位置は適切か確認してください。 電源立上時にこのエラーが発生した後、最終的にネットワークが確立する場合は、ネットワーク異常検出待ち時間を長くしてください。
1205	ロボット側 DPRAM リトライ異常	4	ロボット側から DeviceNet ボード又は CC-Link ボードの DPRAM へアクセスできない状態です。	コントローラのパワースイッチを一度切ってから再操作を行なってください。
1246	MACID の重複	4	自身のノードアドレスが他のノードと重複しています。	どちらかのノードアドレスを変更してください。
1249	CAN 送信のタイムアウト	4	DeviceNet ボードで CAN チップへの送信ができなくなりました。	ネットワーク上で発生している不具合を解決して下さい。 自分自身以外のノードがなく、ネットワーク電源は供給されていると、発生する場合があります。

ｺｰﾄﾞ	メッセージ	Level	説明	復帰処置
124A	DeviceNet RAM 異常	4	DeviceNet 通信部ソフトが RAM のハード異常を検出しました。	コントローラのパワースイッチを一度切ってから再操作を行なってください。
124D	通信側 DPRAM リトライ異常	4	DeviceNet 通信部ソフトが DPRAM へアクセス出来なくなりました。	コントローラのパワースイッチを一度切ってから再操作を行なってください。
21BA	干渉チェック実行エラー	4	干渉チェック実行時エラーが発生しました。	干渉チェックの設定を確認してください。 また、出力先の IO のポート番号が汎用出力、または内部 IO になっているか確認してください。
27AA	プログラムリセット信号が入っています	2	プログラムリセット信号が入っています	リセット信号を切った後、再度実行してください。 コンティニュ起動を行う時にプログラムリセット信号が入っているとこのエラーが発生します。 コンティニュ起動を行う場合はプログラムリセット信号は OFF にしてください。
6149	J１パワーモジュール容量異常	4	1 軸のパワーモジュールとパラメータでの設定値が一致していません。	1. パワーモジュールが接続するモータ出力に合ったものかご確認ください。 2. ロボット型式にあったアームファイルを使用しているかご確認ください。 3. 付加軸として使用する場合､付加軸設定にてモータ出力が正しく行われているかご確認ください。
614A	J2 パワーモジュール容量異常	4	2 軸のパワーモジュールとパラメータでの設定値が一致していません。	↑
614B	J3 パワーモジュール容量異常	4	3 軸のパワーモジュールとパラメータでの設定値が一致していません。	↑
614C	J4 パワーモジュール容量異常	4	4 軸のパワーモジュールとパラメータでの設定値が一致していません。	↑
614D	J5 パワーモジュール容量異常	4	5 軸のパワーモジュールとパラメータでの設定値が一致していません。	↑
614E	J6 パワーモジュール容量異常	4	6 軸のパワーモジュールとパラメータでの設定値が一致していません。	↑
614F	J7 パワーモジュール容量異常	4	7 軸のパワーモジュールとパラメータでの設定値が一致していません。	↑
6150	J8 パワーモジュール容量異常	4	8 軸のパワーモジュールとパラメータでの設定値が一致していません。	↑

6.2 追加されたエラーコード

Ver.2.2 から追加されたエラーコードを下表に示します。

ｺｰﾄﾞ	メッセージ	Level	説明	復帰処置
1218	FlashRom BCC 異常	4	DeviceNet ボードの FlashRom の BCC が異常です。	コントローラのパワースイッチを一度切ってから再操作してください。
1219	パラメータ情報エリア異常	4	DeviceNet ボードのパラメータ情報エリアのデータが異常です。	コントローラのパワースイッチを一度切ってから再操作してください。
121A	ロボット制御部コントロールエリア異常	4	DeviceNet ボードのロボット制御部コントロールエリアのデータ異常です。	コントローラのパワースイッチを一度切ってから再操作してください。
121D	スキャンリストデータテーブル異常	4	DeviceNet ボードのスキャンリストデータテーブルのデータが異常です。	コントローラのパワースイッチを一度切ってから再操作してください。
121E	スキャンリストマッピング情報エリア異常	4	DeviceNet ボードのスキャンリストマッピング情報エリアのデータが異常です。	コントローラのパワースイッチを一度切ってから再操作してください。
129D	CC-Link チェックサム異常	5	CC-Link リモートデバイスボードの FlashRom のチェックサムが異常です。	コントローラのパワースイッチを一度切ってから再操作してください。
129E	通信側 DPRAM リトライ異常 (CC-Link)	4	ロボット側から CC-Link リモートデバイスボードの DPRAM へアクセスできない状態です。	コントローラのパワースイッチを一度切ってから再操作してください。
2035	データ領域が未定義の状態です	2	I/O コマンドでデータ領域の内容が未定義の状態です。	1. データ領域の状態を修正後、再度実行してください。 2. ストローブ信号を立ち上げる際に、データ領域の状態が確定しているか確認してください。
220F	I/O デバイスが変更されました	5	I/O デバイスの状態が前回の設定から変更されました。(例：DeviceNet Slave ボードを新たに実装した時)	I/O の割付設定をした後にコントローラ再立上してください。
7396	整数型変数は使用できません	3	近似比較演算子では整数型は使用できません。	整数型を単精度実数､倍精度実数に変更してください。
7799	特権タスクの最大動作数を超えました。	3	特権タスクの動作数が最大動作数（32 個）を超えました。	特権タスクが同時期に 32 個を超えて動作しないようプログラムを修正してください。

ｺｰﾄﾞ	メッセージ	Level	説明	復帰処置
779B	ファイルの作成できる最大数を超えました	3	コントローラ内のファイルの最大数を超えました。 PAC ファイルは 256 まで、 ヘッダファイル、操作盤ファイルは、あわせて 256 までです。	ファイル数を減らして再操作してください。
7780	フォルダの作成できる最大数を超えました	3	フォルダの作成できる最大数を超えました。フォルダの最大数は 256 です。	フォルダ数を最大数を超えない数に減らして再操作してください。

取扱説明書 追補版

Ver. 2.2 の新機能

初　版　2004 年 11 月

株式会社デンソーウェーブ

4G50C