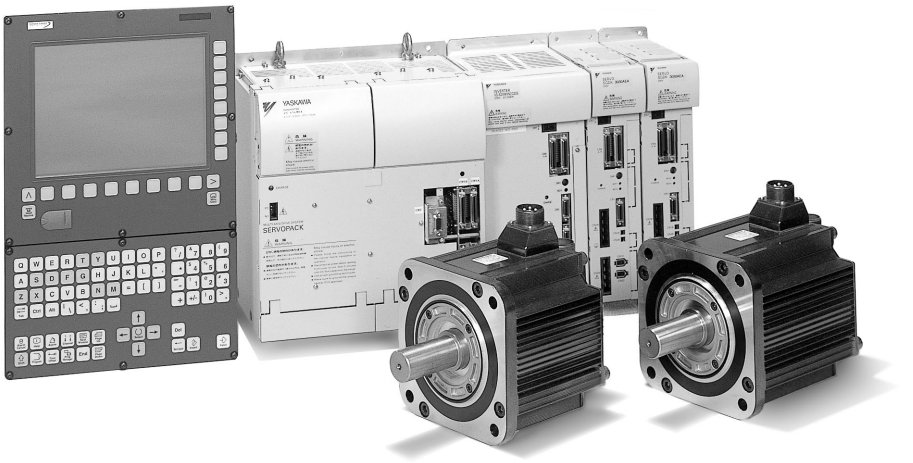
Yaskawa Siemens CNC シリーズ

SINCOMコンピュータリンク説明書



資料番号　NCSI-SP02-17

本マニュアルは Yaskawa Siemens 840DI、Yaskawa Siemens 830DI 両モデル用に作成されています。

本文中の記述では両モデルの機能差は区別されておりませんので、それぞれのモデルにどの機能が標準装備されているか、どの機能がオプションで装備可能かについては別途、機能一覧表をご参照下さい。また、本文中に 840DI と言った表現が出て来ますが、830DI も意味している事があるとご理解下さい。

Yaskawa Siemens 840DI

SINCOM コンピュータリンク説明書

製造業者／サービス文書

対象制御装置

制御装置
Yaskawa Siemens 840DI

04.01 版

Yaskawa Siemens 文書

版の履歴

今回の版の概略説明および今までに作成された版を下記に示します。

「備考」欄のコードが、各版のステータスを示しています。

「備考」欄のステータスコードの意味は次のとおりです。

A . . . . . 新規作成

B . . . . . 新しいオーダ番号で印刷し直した未改訂の文書

C . . . . . 新しいステータスの改訂版

前回の版以降に実際に変更があったページには、そのページのヘッダ部分に新しい版のコードが示されています。

版	オーダ番号	備考
04.01	NCSI-SP02-17	A

本文書に説明のない他の機能でも制御装置で実行できる場合がありますが、そのような機能は新しい制御装置やサービス時に利用できるとは限りません。

本文書の記述と、対象となるハードウェアおよびソフトウェアとが一致しているかどうかは十分に確認されています。しかし相違点がまったくないとは言えず、完全に一致しているとは保証できません。本文書に記載されている情報は定期的に検討され、必要な変更は次の版に反映されます。さらなる改善のために皆様のご意見をお待ちしています。

本内容は予告なしに変更されることがあります。

はじめに

文書の構成

Yaskawa Siemens 文書は次の 3 つのレベルで構成されています。

- 一般文書
- ユーザー文書
- 製造業者／サービス文書

対象読者

本書は Yaskawa Siemens 840DI（以降 YS 840DI と略す）を使用する工作機械を製作される方を対象としています。

目的

本書は YS 840DI のコンピュータリンクソフトウェアおよび接続されるホストコンピュータ側で準備が必要なソフトウェアについて説明しています。
本機能で使用するインタフェースについても記述しています。

説明範囲

本書は以下の 2 部で構成されています。

- パート 1：MMC とホストコンピュータ間のインタフェース
- パート 2：PLC とホストコンピュータ間の MMC 経由のインタフェース

シンボル表示　次のマークには特別な意味があり，本文書中で使用されています。

（注）　本文書中に（注）マークを使用して関連する情報に注意を喚起します。

!　重要
重要な情報を読者に伝えたいときに使用されています。

警告表示　本書では，危険度に応じて次の警告表示が使用されています。

危険
この記号は，適切な注意が払われないと，致死，重傷，あるいは物的損害が発生することを表します。

注意
この記号は，適切な注意が払われないと，軽傷，あるいは物的損害が発生する恐れがあることを表します。

警告
この記号は，適切な注意が払われないと，致死，重傷，あるいは物的損害が発生する恐れがあることを表します。

参照　このマークは他の文書に具体的な情報がある場合に使用します。

登録商標　MS-DOS® および WINDOWS™ は米国の Microsoft Corporation の登録商標です。

関連マニュアル

■ 関連するマニュアルについては，下表に示すものがあります。必要に応じてご覧ください。

■ 製品の仕様，使用制限などの条件を十分ご理解いただいたうえで，製品をご活用ください。

マニュアル名称	資料番号
Yaskawa Siemens 840DI 結合説明書 ハード編	NCSI-SP02-01
Yaskawa Siemens 840DI 結合説明書 機能編	DE0400309
Yaskawa Siemens 840DI PLC トレーニングマニュアル	DE0400515
Yaskawa Siemens 840DI ユーザーズマニュアル 操作編	NCSI-SP02-04
Yaskawa Siemens 840DI ShopMill セットアップマニュアル	NCSI-SP02-05
Yaskawa Siemens 840DI ユーザーズマニュアル プログラミング編 ISO 互換 G コード説明書（マシニングセンタ用）	NCSI-SP02-20
Yaskawa Siemens 840DI ユーザーズマニュアル プログラミング編 ISO 互換 G コード説明書（旋盤用）　（制作中）	NCSI-SP02-21
Yaskawa Siemens 840DI ユーザーズマニュアル プログラミング編 基本説明書	NCSI-SP02-06
Yaskawa Siemens 840DI ユーザーズマニュアル プログラミング編 上級説明書	NCSI-SP02-07
Yaskawa Siemens 840DI ユーザーズマニュアル プログラミング編 サイクル説明書	NCSI-SP02-08
Yaskawa Siemens 840DI ユーザーズマニュアル プログラミング編 計測サイクル説明書	NCSI-SP02-09
Yaskawa Siemens 840DI 保守説明書	NCSI-SP02-10
Yaskawa Siemens 840DI 保守説明書 サービスマンハンドブック　（制作中）	NCSI-SP02-19
Yaskawa Siemens 840DI 保守説明書 別冊付録 一覧表	NCSI-SP02-11
Yaskawa Siemens 840DI 保守説明書 別冊付録 アラーム診断ガイド	NCSI-SP02-12
Yaskawa Siemens 840DI API 取扱説明書 HMI プログラミングパッケージ 基礎編	NCSI-SP02-13
Yaskawa Siemens 840DI API 取扱説明書 HMI プログラミングパッケージ COM および OPC クライアント編	NCSI-SP02-14
Yaskawa Siemens 840DI API 取扱説明書 HMI プログラミングパッケージ インストールガイド	NCSI-SP02-15
Yaskawa Siemens 840DI シンクロナイズドアクション説明書	NCSI-SP02-16
Yaskawa Siemens 840DI SINCOM コンピュータリンク説明書　（本書）	**NCSI-SP02-17**
Yaskawa Siemens 840DI ツールマネージメント説明書	NCSI-SP02-18

パート 1 ホストコンピュータインタフェース

1 概要

1.1 一般説明

ホストコンピュータと YS 840 DI との間のインタフェースの説明。

- これらのシステム間の通信はイーサネットと TCP/IP プロトコルに基づいています。
- データ転送は，ホストコンピュータと MMC から開始することができます。
- コンピュータ通信インタフェースは, DCE 規格を使用したファイル転送と RPC（リモートプロシージャコール）に基づいています。

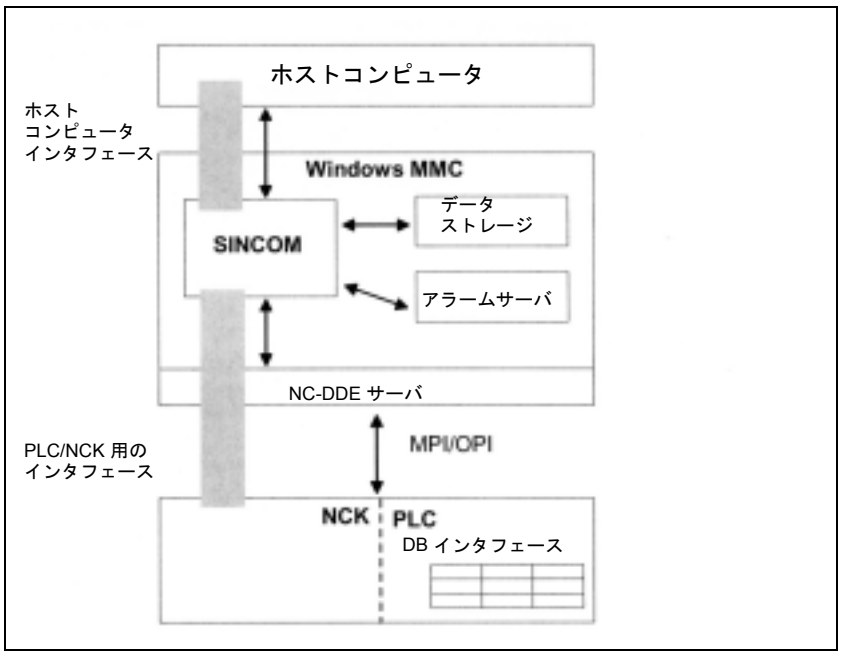


図 1.1 インタフェース

データ通信

大量のデータが転送される場合，そのデータは RPC ファンクションコールで送られます。すでにファイルフォーマットで存在している NC プログラムなどのデータも同様にファイルとして転送されます。ツールデータなどのより大量のデータは，ファイルに書込まれ，同様にファイルとして転送されます。

ファイルの名称などのパラメータは，RPC ファンクションコールで通信パートナに渡されます。

ホストコンピュータと Windows MMC との間のデータ交換で利用できる機能に基づいて，ファイルの転送が FTP を使用して行われるか，ネットワークシステム上にある場合，コピーすることによって行われるかが決まります。

RPC ファンクション

本書では，ホストコンピュータおよびユーザー用の RPC ファンクションについて説明します。ホストコンピュータによって受信された RPC コールのことを以降は " コールされたファンクション " と呼びます。ホストコンピュータに到着した RPC コールは，ホストコンピュータ中の対応するファンクションをコールします。

ホストコンピュータによってコールされる RPC コールのことを以降は " ファンクションコール " と呼びます。ホストコンピュータのファンクションコールは，SINCOM 中の対応するファンクションをコールします。

2 システムインストレーション

2.1 システム要件

2.1.1 ソフトウェア

YS 840DI

YS 840DI では次のソフトウェアが必要となります。

- MMC, SW 4.x
- イーサネットインタフェース用のドライバおよび，TCP/IP
- ツールマネージメント
- コンピュータリンクソフトウェアパッケージ

ホストコンピュータ

- インストールされたイーサネットインタフェースモジュール用のドライバ。DCE 規格に基づいた rpc と FTP または NFS サービスを有する TCP/IP

2.1.2 ハードウェア

YS 840DI

- イーサネット用のインタフェースモジュール : 3COM Etherlink III Combo (3C509)

ホストコンピュータ

- イーサネット用のインタフェースモジュール : 3COM Etherlink III Combo (3C509)

YS 840DI ／ホストコンピュータ接続

ネットワークケーブルがイーサネットインタフェースで使用されます。

2.2 インストレーション

インストレーションルーチンの間に，イーサネットインタフェースが MMC 上にインストールされ，TCP/IP を含む関連ドライバがロードされ，IP アドレスが入力されなければなりません。

（注）ドライバのインストレーションおよびセットアップの正確な手順については，イーサネットインタフェースのマニュアルを参照してください。
コンピュータリンクソフトウェアには WORD ファイル (Liesmich.doc) が提供されています。
このファイルには提供されるバージョンについてのインストレーションおよびセットアップインストラクションが含まれています。

コンピュータ通信ソフトウェアが MMC にロードされ，このソフトウェアの Ini ファイルが構成されます。

DB インタフェース用のデータブロックが PLC 上にセットアップされなければなりません。これは通常，機械メーカの PLC がインストールされると実行されます。

データブロック構造については，1.1「一般説明」で説明します。

3 動作モードおよびステータス

3.1 動作モード

YS 840DI は，自動，MDA，JOG および TEACH IN モードで動作します。コンピュータリンクにはこれらのモードに加えて自身の動作モードが必要となります。SINCOM の動作は，PLC と共に，ホストコンピュータ通信モードによって決まります。ホストコンピュータ通信モードを操作および表示するために個別のダイアログが MMC に提供されます。ホストコンピュータ通信モードには次のものがあります。

- ホストコンピュータモード（無人）
- ホストコンピュータモード（有人）
- 手動モード
- 特殊モード
- オフライン

3.1.1 ホストコンピュータモード（無人／有人）

この 2 つのホストコンピュータ通信モード（有人および無人）の場合，ホストコンピュータ上で NC プログラムを指定し，自動モードと関連して，PLC からこのプログラムをスタートさせることが可能です。有人モードと無人モードの相違により，故障の場合にそれぞれ異なる応答を開始することができます。

有人生産中に故障が生じた場合，ワークキャリアはマシン内に留まります。加工処理を再開する前にオペレータは故障を修復しなければなりません。

無人生産中に故障が生じた場合，ワークキャリアの加工を終了して，別のワークキャリアで継続することが可能です。

3.1.2 手動モード

NC プログラムはホストコンピュータによって指定されます。NC プログラムはコンピュータリンクを介して選択されます。選択された NC プログラムは，ユーザーインタフェースのヘッダ行に表示されます。

オペレータがプログラムをスタートさせなければなりません。

（注）ホストコンピュータ通信モードおよび手動モードの場合，マシン上の材料の流れは自動的です。つまり，トランスポートシステムが自動的にワークキャリアを運び，加工後にワークキャリアを回収します。

3.1.3 特殊モード

特殊モードの場合，ワークキャリアを自動的に移動させることはできません。またホストコンピュータはプログラム割当てを転送しません。NC プログラムを自動的にスタートさせることもできません。マシンはオペレータによって制御され，ホストコンピュータとの完全な通信が実行できます。特殊モードは通常，NC プログラムをテストするか計画されていないワークを手動で加工するのに使用されます。

3.1.4 オフライン

オフラインは，ホストコンピュータと MMC との間の接続が中断されたことを示します。MMC は，ホストコンピュータとの接続が中断されたことを検出すると，オフラインに切換ります。ホストコンピュータも同様に，接続の中断を検出すると直ちにそのステータスデータ中およびプラントディスプレイ中にマシンがオフラインであると示します。

オフライン後，マシンはホストコンピュータ上で同期化されなければなりません。

動作モード（YS 840DI モードあるいはコンピュータ通信モード）がマシン上で切換った場合，このことは RPC コール R_MACHINE_H() を使用してホストコンピュータにレポートされなければなりません。

3.2 マシンステータス

マシンのステータスは次のいずれかになります。

- コールドリスタート　　: MMC 上でのリスタート後
- 非アクティブ　　: 加工なし
- アクティブ　　: 加工進行中
- 故障　　: 加工中断
- ユニット停止

3.3 ドッキング位置／ストレージロケーションステータス

トランスポートシステム上でトランスポートジョブを開始するために，ホストコンピュータはドッキング位置のステータスを認識しなければなりません。

次のステータスが可能です。

- 利用可能
- トランスポート制御系についてディスエーブル
- 故障

3.4 ワークキャリアステータス

ホストコンピュータがトランスポートジョブを決定するには，ワークキャリアステータスも同様に必要となります。ワークキャリアのステータスは次のいずれかになります。

- 加工されない - プログラム割当てなし：
- 加工用の NC プログラムがまだ割当てられていません。つまり，まだ加工できません
- 加工されない - プログラムは割当てられている：
- 進行中
- 終了
- エラーを伴って終了
- バッファリング用：

ワークキャリアはマシン上でバッファされるだけです。加工は行われません。

4 ツールデータ

4.1 ツールデータ

ツールのデータは必ずしもすべて必要とされるとは限らないので，3 つのバージョンが利用できます。

1. 第 1 バージョンには完全な範囲のツールデータが含まれます。他の 2 つのバージョンにはツールデータのサブセットしか含まれません。
2. ツールごとのデータ範囲の構成可能性は，完全なエリアにしか及びません。つまり，各バージョンに（レジストリで）提供されるリストが，このバージョンで転送されるエリアを指定します。

ツールデータ用のファイルフォーマットは，YS 840DI のデータバックアップフォーマット（/BA/ によると，パンチテープ／ ASCII フォーマット；たとえば，_N_TOx_TOA または _N_TOx_INI 中の）に対応します。データコンテンツおよびレイアウトの完全な説明については，本書パート 2 を参照してください。

ツールデータエリア

ツールデータは NCK 上の種々のエリアに保存されます。

ツールデータエリアは次のように示されます。

表 4.1 ツールデータエリア

エリア名	ツール名
一般ツールデータ	$TC_TPx[y]
ユーザー関連ツールデータ	$TC_TPCx[y]
切削エッジデータ	$TC_DPx[y,z]
ユーザー関連切削エッジデータ	$TC_DPCx[y,z]
ツール監視データ	$TC_MOPx[y,z]
ユーザー関連ツール監視データ	$TC_MOPCx[y,z]

x　システム変数の特異な名称を生成するのに使用される，エリアごとの通し番号です。

y　T 番号です。

z　切削エッジ番号です。

ファイルの構造

表 4.2

	説明
$TC_TP1[1]= 2	Duplo 番号
$TC_TP2[1]= "4711"	ID 番号
$TC_TP3[1]=1	ハーフロケーションにおける左側の容量
$TC_TP4[1]=1	ハーフロケーションにおける右側の容量
$TC_TP5[1]=1	ハーフロケーションにおける上側の容量
$TC_TP6[1]=1	ハーフロケーションにおける下側の容量
$TC_TP7[1]=2	マガジンロケーションタイプ
$TC_TP8[1]=131	ステータス
$TC_TP9[1]=0	ツール監視のタイプ
$TC_TP10[1]=2	置換え方法
$TC_TP11[1]=0	ツール情報
$TC_DP1[1,1]=0	切削エッジパラメータ 1
$TC_DP2[1,1]=0	切削エッジパラメータ 2
$TC_DP3[1,1]=0	切削エッジパラメータ 3
$TC_DP ...	...
$TC_DP24[1,1]=0	切削エッジパラメータ 24
$TC_DP25[1,1]=0	切削エッジパラメータ 25

5 ホストコンピュータと WINDOWS MMC との間の通信

5.1 リモートプロシージャコールの概要

ホストコンピュータと YS 840DI 工作機械上の MMC との間の通信は，少量のデータを転送する場合，リモートプロシージャコール (RPC) を使用して行われます。RPC を介した通信の間に，通信パートナは，そのコールに含まれるパラメータ（データ）付きのプロシージャ名で示されたファンクションを実行するようリクエストされます。

（注）以下で使用される RPC コールのインタフェース定義言語 IDL（テクニカルプログラム定義）については，付録を参照してください。

本セクションで説明されるプロシージャの名称は以下のシステムに基づいて構成されます。

5.1.1 プロシージャ名の構造

プロシージャ名には次の 3 つの構成要素が含まれます。

1. 指令識別子（最初の文字）
2. データ／ファンクション識別子
3. レシーバ識別子（最後の文字）

指令識別子

指令識別子は，プロシージャ名の最初の文字に現れます。

C	指令コール	(Command)
R	データリクエストの受信	(Receive)
T	データリクエストの送信	(Transmit)

例：T_MACHINE_M ().

データ／ファンクション識別子

この識別子は，リクエストあるいは送信されるデータのタイプ，あるいはそのデータが渡されるファンクションを示します。

例：T_MACHINE_M (), R_NC4WPC_M().

レシーバ識別子

レシーバ識別子は，ファンクションを実行するユニットのアドレスを示します。

H	ホストコンピュータがレシーバです	(Host)
M	マシンがレシーバです	(Machine)

5.1.2 一般パラメータ

ホスト

ホストコンピュータの名称（最高 16 文字)。複数のマシンが複数のホストコンピュータにネットワーキングされる場合，ホストは，データを交換するホストコンピュータ用の独自の識別子になります。

マシン

マシンの名称（最大 16 文字)。ネットワーク上に存在するすべてのマシンについて独自の識別子が利用できなければなりません。

OrderNum

オーダ番号：この番号はオプションであり，RPC リクエストおよびその回答が独自に割当てられなければならない場合に使用できます。

（注）パラメータのタイプが文字列の場合，文字列は '\0' で限定されなければなりません。最大文字列長は，個々のパラメータによって指定されます。

5.2 通信シーケンス

前提条件

ホストコンピュータと 1 つまたは複数のマシンとの間でエラーのない通信を実行するには，ホストコンピュータが RPC を処理する対象となる通信パートナをホストコンピュータが知っている必要があります。これらのマシン（クライアント）のマシン名がホストコンピュータ上に保存されなければなりません。

パラメータ

パラメータとは，RPC 内で転送されるデータのことです。RPC ごとに転送されるテーブルブロックは，ホストコンピュータおよびマシンに対するパラメータを説明しています。

確認応答

ローカルプロシージャの場合と同様に，RPC の戻り値は確認応答または否定応答を示します。リクエストが非同期に処理された場合，この確認応答はリクエストの受信しか確認応答することができません。処理後に，あるいは処理中にエラーが発生した場合は，適切な RPC メッセージを使用して通信パートナに通知しなければなりません。エラーメッセージは，アラームサーバと関連して，MMC 上に表示されます。コールが正しく実行された場合は，戻り値 = 0 となります。エラーが生じた場合の戻り値のリストについては付録を参照してください。

（注）RPC を開始する SINCOM のコンポーネントは，RPC が処理されるのを待機し，その間，それ以上の処理を実行することができないので，ホストコンピュータソフトウェアはできるだけ早くコールされたファンクションを返さなければなりません。ホストコンピュータ上にコールされたファンクション内では，RPC に含まれるデータがバッファにコピーされ，ファンクションが直ちに返されるべきです。ホストコンピュータ上での実際の処理は，その後に実行されるべきです。

5.3 マシンステータスデータ

5.3.1 マシンステータスデータの送信

コールされたファンクション

R_MACHINE_H (ホスト,
マシン,
OrderNum,
MachineMode,
MachineStatus,
NCProgram,
ClampCubeSide,
DockPos,
DockPosStatus,
WPC,
WPCStatus,
Resint1,
Resint2,
Resbyte)

転送方向： MMC → ホストコンピュータ

意味

マシンデータをホストコンピュータに送信する。

データ

表 5.1 マシンステータスデータ転送用のパラメータ

パラメータ	説明	フォーマット
ホスト	ホストコンピュータ（ホスト）の名称	16 バイト（文字列）
マシン	マシン名	16 バイト（文字列）
OrderNum	オーダ番号	4 バイト（long.int）
MachineMode	動作モード ・YS 840DI モード： 1: 自動 2: MDA 4: JOG 8: TEACH IN ・コンピュータリンクモード： 100: ホストモード（無人） 200: ホストモード（有人） 300: 手動モード 400: 特殊モード	4 バイト（long.int）
MachineStatus	マシンステータス 0: コールドリスタート 1: 非アクティブ 2: アクティブ 3: 故障 4: コンポーネント停止	4 バイト（long.int）
NCProgram	現在の NC プログラム *	128 バイト（文字列）
ClampCubeSide	クランプキューブ上のサイド **	4 バイト（long.int）
DockPos[3]	ドッキング位置番号 ドッキング位置番号は，インタフェース DB のドッキング位置リスト中のインデックスに対応する。 ドッキング位置番号 = 0 は " 割当てられていない " を意味する (1.1「一般説明」を参照)。	3 x 4 バイト（long.int）
DockPosStatus[3]	ドッキング位置ステータス 0: 利用可能 1: トランスポート制御系についてディスエーブル 2: 故障	3 x 4 バイト（long.int）
WPC[3]	ワークキャリア名	3 x 6 バイト（文字列）
WPCStatus[3]	ワークキャリアステータス 1: 加工されない，プログラムが割当てられていない 2: 加工されない，プログラムは割当てられている 4: プログラム選択の準備 8: プログラム選択終了 16: 進行中 32: 終了 64: エラーを伴って終了 128: バッファリング用	3 x 4 バイト（long.int）
Resint1***	予備 1	4 バイト（long.int）
Resint2***	予備 2	4 バイト（long.int）

パラメータ	説明	フォーマット
Resbyte	予備 3	8 バイト（文字列）

* 現在実行中の NC プログラムの NC プログラム名

** 現在加工中の（クランプキューブ上の）サイド

*** Resint 1 および 2 は，PLC の DB インタフェースに表示されます。PLC によって値が DB インタフェースに入力された場合，その値はホストコンピュータに転送されます。これらの値はコンピュータリンクに影響を与えません。ホストコンピュータに転送されるだけです。

使用に関する注記

- MMC は，マシン上でステータスが切換るごとにこの RPC を開始しなければなりません。コンピュータ通信ソフトウェアが現在のデータを決定し，RPC を開始します。
- DB インタフェースで特定のビットをセットすることにより，PLC がプロシージャをトリガします。
- 指令 T_MACHINE_M （マシンステータスデータのリクエスト，下記を参照）を使用して，ホストコンピュータがプロシージャを開始することもできます。

（注）3 つ以上のドッキング位置が説明される場合，5.12 「マシン割当てデータ」に基づいて，個別の説明ファイルが転送されなければなりません。

（注）・変数中の両方のノード (MachineMode) の動作モード（YS 840DI およびホストコンピュータ）をレポートするために，その量を値として転送することができます（たとえば，201：ホストコンピュータモード（有人）= 200 および YS 840DI 自動 = 1）。

・コンピュータ通信ソフトウェアは動作モードを確認しません。

5.3.2 マシンステータスデータのリクエスト

ファンクションコール

T_MACHINE_M (ホスト,
マシン,
OrderNum)

転送方向： ホストコンピュータ → MMC

意味

マシンステータスデータをリクエストする

データ

表 5.2 マシンステータスデータのリクエスト

パラメータ	説明	フォーマット
ホスト	ホストコンピュータ（ホスト）の名称	16 バイト（文字列）
マシン	マシン名	16 バイト（文字列）
OrderNum	オーダ番号	4 バイト（long.int）

使用に関する注記

ホストコンピュータは，たとえば同期中に，マシンステータスデータをリクエストするのにこのコールを使用できます。次に，MMC が指令 R_MACHINE_H() を使用してホストコンピュータにそのデータを送信します。

例

```
T_MACHINE_M("FLR1","BAZ3",0);
```

5.4 生産ダイアログ

説明

ワークキャリアがマシンに到着すると，PLC による開始時に，MMC はマシンステータスデータをホストコンピュータに送信します。

ホストコンピュータはそのデータからどのワークキャリアがマシンに到着したのかを認識することができます。次に，ホストコンピュータはこのワークキャリアに割当てられたプログラムを転送します。

ワークキャリアがクランプキューブを有する場合，1 つの NC プログラムがキューブの各サイドに割当てられます。コンピュータリンクソフトウェアはこれらのプログラム割当てを保存します。各プログラム割当ては，ワークキャリア，サイドおよび NC プログラムから構成されます。どの場合でも，次の NC プログラムが転送され，選択されます。NC プログラムはその後 PLC からスタートさせることができます（ホストコンピュータモード（有人）および（無人）で）。マシンは，NC スタートと，後に NC エンドをマシンステータスデータでレポートします。

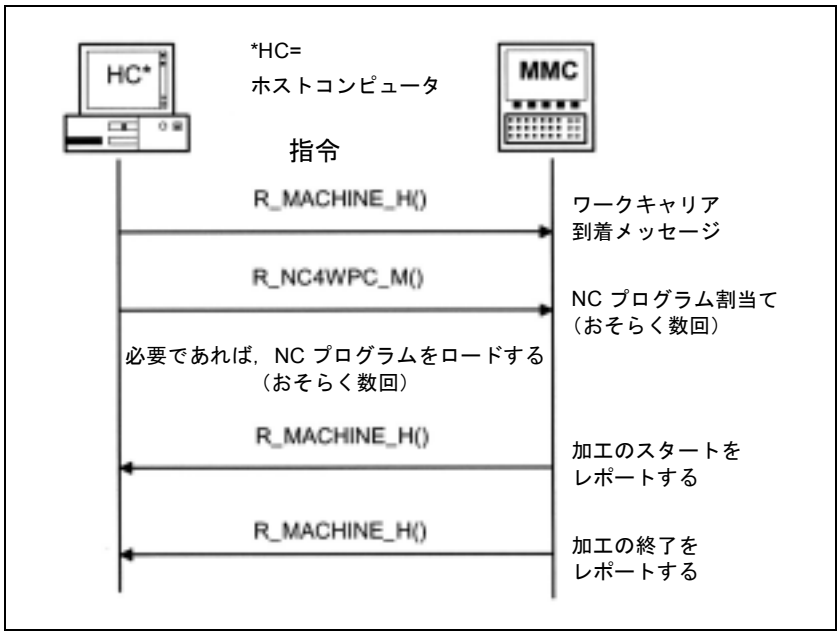


図 5.1 生産ダイアログ，通常の実行，エラーなし

このプロシージャの間にエラーが生じた場合，メッセージが出力されます（5.5.1「MMC からホストコンピュータへのメッセージ」を参照）。

5.4.1 プログラム割当て

ファンクションコール

R_NC4WPC_M (ホスト,
マシン,
OrderNum,
WPC,
NCProg,
日時,
NCPLength,
ClampCubeSide,
TpFlag,
NCExtern,
Resint1,
Resint2,
Resbyte)

転送方向： ホストコンピューター→ MMC

意味

どのプログラムを起動すべきかをマシンに指示する

データ

表 5.3 プログラム割当て用のパラメータ

パラメータ	説明	フォーマット
ホスト	ホストコンピュータ（ホスト）の名称	16 バイト（文字列）
マシン	マシン名	16 バイト（文字列）
OrderNum	オーダ番号	4 バイト（long.int）
WPC	ワークキャリア名	6 バイト（文字列）
NCProg	NC プログラム NC プログラムの例： "\mpf.dir\cylinderhead.mpf"	128 バイト（文字列）
日時	NC プログラムの最終変更日時 （ユニックス時間）	4 バイト（long.int）
NCPLength	バイトで表した NC プログラムのサイズ	4 バイト（long.int）
ClampCubeSide	クランプキューブ上のサイド	4 バイト（long.int）
TpFlag	トランスポートフラグ = 0: フォローアップオペレーションなし = 1: フォローアップオペレーション = 9: バッファリング用	4 バイト（long.int）

パラメータ	説明	フォーマット
NCExtern	外部ソースからの実行（オプション） 0: NCK 上で NC プログラムを実行する 1: 外部ソースからプログラムを実行する	4 バイト（long.int）
Resint1	予備 1	4 バイト（long.int）
Resint2	予備 2	4 バイト（long.int）
Resbyte	予備 3	8 バイト（文字列）

使用に関する注記

- クランプキューブの各サイドが個別の NC プログラムによって加工される場合，ワークキャリアについてこの RPC は数回実行されることがあります。ホストコンピュータがプログラム割当てを SINCOM にレポートした順に，サイドが加工されます。
- 最後の 1 つを除いてワークキャリアのすべてのプログラムが割当てられると，トランスポートフラグ "1 = フォローアップオペレーション " がセットされなければなりません。フォローアップオペレーションフラグがサイドについてセットされた場合，ワークキャリアは NC プログラムの終了時に加工ステーションに留まります。最後のサイドについてフラグがセットされていないという事実は，それ以上の加工が必要でないということと，ワークキャリアを加工ステーションから移動させることができるということを示しています。
- ワークキャリアがバッファリングだけの目的でマシン上に保存された場合，トランスポートフラグ "9 = バッファリング用 " をセットすることによって，このことをレポートすることができます。この場合，NC プログラムは指定されません。

例

R_NC4WPC_M ("FLR1", "BAZ3", 0, "WPC05", "\mpf.dir\Kw15.mpf",
862826400, 3210, 1, 0, 0, 0, 0, "\0");

5.5 メッセージ

5.5.1 MMC からホストコンピュータへのメッセージ

コールされたファンクション

R_REPORT_H (ホスト,
マシン,
OrderNum,
タイプ,
番号,
時刻,
フラグ ,
Resint1,
Resint2,
Resbyte)

転送方向： MMC → ホストコンピュータ

意味

ホストコンピュータにメッセージを送信する

データ

表 5.4 メッセージ用のパラメータ

パラメータ	説明	フォーマット
ホスト	ホストコンピュータ（ホスト）の名称	16 バイト（文字列）
マシン	マシン名	16 バイト（文字列）
OrderNum	オーダ番号	4 バイト（long.int）
タイプ	メッセージタイプ 1: アラーム 2: オペレータ中断 3: 操作メッセージ 4: コンピュータリンクソフトウェアからのエラーメッセージ 5: R_DATA_M (), R_DDEDATA_M () および R_VAR_M () 付きの確認応答	4 バイト（long.int）
番号 [10]	メッセージ番号配列 ・最高 10 アラーム用の配列。 ・必要でない配列要素は 0 で初期化されるべきである。 ・他のメッセージタイプでは，番号 [0] しか割当てられない。	10 x 4 バイト（long.int）
時刻 [10]	タイムスタンプ配列 ・最高 10 エントリ用の配列。 ・必要でない配列要素は 0 で初期化されるべできである。	10 x 4 バイト（long.int）
フラグ [10]	着信／発信フラグ，最高 10 エントリの配列 C: 着信メッセージ，マシン停止なし S: 着信メッセージ，マシン停止 L: 全メッセージ発信 • 着信メッセージでは次のことが区別されなければならない：マシン停止はい／いいえ	10 x 1 バイト

パラメータ	説明	フォーマット
Resint1	予備 1	4 バイト（long.int）
Resint2	予備 2	4 バイト（long.int）
Resbyte	予備 3	8 バイト（文字列）

R_DATA_M ()，R_DDEDATA_M () および R_VAR_H () の場合，このコール中に同期処理を実行することはできません。したがって，SINCOM は処理後に確認応答をホストコンピュータに送信し，処理が終了したことをホストコンピュータに通知しなければなりません。R_DATA_M のサブファンクション番号は，R_DATA_M に対する確認応答付きの " エラー番号 " として使用されます。これは確認応答を割当てるのに使用されます。

エラー番号 "1000" は，R_DDEDATA_M を伴って返され，エラー番号 " 0" は，R_VAR_M を伴って返されます。

（注）インデックスが同じ番号，時刻およびフラグはグループを構成します。

使用に関する注記

RPC R_REPORT_H() は，個別のメッセージまたは最高 10 個のアラームメッセージから成るグループをホストコンピュータに送信するのに使用されます。

特殊状況：最後のアラームがマシン上で発信すれば，このステータスは R_REPORT_H() と次のパラメータによってレポートされます。

- タイプ = 1
- 番号 [0] = 0
- フラグ [0] = L

がホストコンピュータに送信されます。

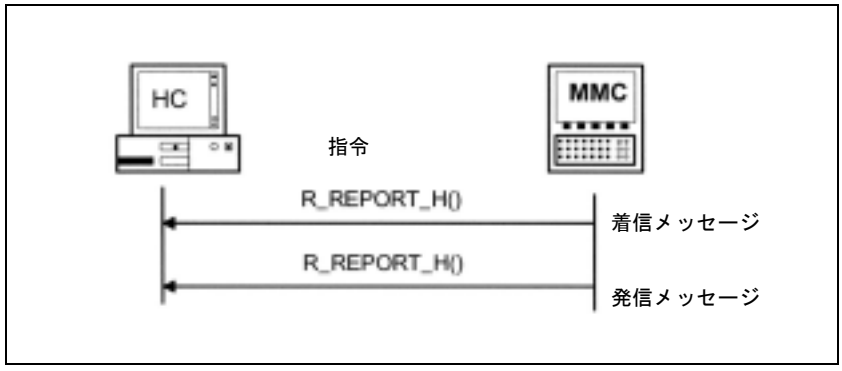


図 5.2 アラーム／オペレータ中断／操作メッセージ用のプロシージャ

マシン停止を伴うアラームまたはオペレータ中断の終了後もマシン上で加工が継続する場合，マシンステータスと共に，RPC R_MACHINE_H () を使用して（トランスポートシステムの場合は，R_TPS_H () を使用して）このことをホストコンピュータにレポートしなければなりません。

メッセージタイプ 4 用のエラーメッセージのリストについては，付録を参照してください。

5.5.2 ホストコンピュータから MMC へのメッセージ

ファンクションコール

R_REPORT_M (ホスト,
マシン,
OrderNum,
タイプ,
番号,
Resint1,
Resint2,
Resbyte)

転送方向： ホストコンピュータ → MMC

意味

ホストコンピュータのエラーメッセージが MMC 上のコンピュータリンクソフトウェアに送信されます。

データ

表 5.5

パラメータ	説明	フォーマット
ホスト	ホストコンピュータ（ホスト）の名称	16 バイト（文字列）
マシン	マシン名	16 バイト（文字列）
OrderNum	オーダ番号	4 バイト（long.int）
タイプ	メッセージタイプ 4: MMC に対するホストコンピュータのエラーメッセージ	4 バイト（long.int）
番号	エラー番号	4 バイト（long.int）
Resint1	予備 1	4 バイト（long.int）
Resint2	予備 2	4 バイト（long.int）
Resbyte	予備 3	8 バイト（文字列）

使用に関する注記

MMC 上では，エラーメッセージを

- ログファイルに記録できます
- アラームサーバによって表示できます
- SINCOM によって評価できます

例

R_REPORT_M ("FLR1", "BAZ3", 0, 4, -13, 0, 0, "\0");

5.6 ユーザーメッセージの交換

5.6.1 MMC へのメッセージ（V1.0 には実装されていないオプション）

ファンクションコール

R_MESSAGE_M(ホスト,
マシン,
OrderNum,
メッセージ,
Resint1,
Resint2,
Resbyte)

転送方向 ホストコンピュータ → MMC

意味

MMC の操作パネル上にメッセージを表示する

データ

表 5.6

パラメータ	説明	フォーマット
ホスト	ホストコンピュータ（ホスト）の名称	16 バイト（文字列）
マシン	マシン名	16 バイト（文字列）
OrderNum	オーダ番号	4 バイト（long.int）
メッセージ	メッセージテキスト	128 バイト（文字列）
Resint1	予備 1	4 バイト（long.int）
Resint2	予備 2	4 バイト（long.int）
Resbyte	予備 3	8 バイト（文字列）

使用に関する注記

ホストコンピュータは，次のパートプログラムの処理を準備するために，たとえば，準備されるツールまたは材料についてのメッセージをマシンオペレータに送信します。

例

R_MESSAGE_M("FLR1", "BAZ3", 0, "Hello Machine", 0, 0, "\0");

5.6.2 ホストコンピュータへのメッセージ（V1.0 には実装されていないオプション）

コールされたファンクション

R_MESSAGE_H(ホスト,
マシン,
OrderNum,
メッセージ,
Resint1,
Resint2,
Resbyte)

転送方向 MMC → ホストコンピュータ

意味

ホストコンピュータ上にメッセージを表示する

データ

表 5.7

パラメータ	説明	フォーマット
ホスト	ホストコンピュータ（ホスト）の名称	16 バイト（文字列）
マシン	マシン名	16 バイト（文字列）
OrderNum	オーダ番号	4 バイト（long.int）
メッセージ	メッセージテキスト	128 バイト（文字列）
Resint1	予備 1	4 バイト（long.int）
Resint2	予備 2	4 バイト（long.int）
Resbyte	予備 3	8 バイト（文字列）

使用に関する注記

マシンオペレータは，プログラム実行の準備が実行されたことをホストコンピュータに通知します。

5.7 データダイアログ

説明

NC プログラムやツールデータなどのより大きなデータはファイルフォーマットで転送されます。FTP プロトコル（ファイル転送プロトコル）を使用すると，Windows MMC は FTP クライアントとしてしか機能できないので，ファイル転送は常に MMC 上のコンピュータ通信ソフトウェアによって実行されなければなりません。ホストコンピュータは，RPC コール R_DATA_M を使用して，ファイルが処理できる状態であることを MMC 上のコンピュータ通信ソフトウェアに通知します。次に，コンピュータ通信ソフトウェアはそのファイルを読出し，処理します。反対方向では，MMC 上のコンピュータ通信ソフトウェアがそのファイルを転送し，RPC コール R_DATA_H を使用して，そのファイルがホストコンピュータ上で処理できる状態であることをホストコンピュータに通知します。

プログラム転送

NC プログラムはファイルフォーマットで転送されます。転送後，NC プログラムは NC データ管理システムに保存されなければなりません。

ファイルコンテンツの識別

R_DATA_M() および R_DATA_H() コール中のデータには，データのタイプと正確な転送に必要とされる条件とを識別するサブファンクション番号が含まれます。

ツールデータが入ったファイルはツールごとに読取られ，指定されたサブファンクション番号に基づいて処理されなければなりません。ファイルは処理後に処理ファンクションによって削除されなければなりません。

コンピュータリンクがスタートアップされると，受信された古いデータはすべて削除されなければなりません。

次のサブファンクション番号は同じです。

- リクエスト，
- 転送，および
- 削除

表 5.8 サブファンクション番号：SFct

サブファンクション番号	ファンクション	注釈
1	NC プログラム	Name1 = NC プログラム Name2 = パスが HC 上にあるファイル名
10	現行の NC プログラムのリスト	Name1 = ファイルパス Name2 = リストファイルの名称
20	すべてのツールのツールステータスデータ ツールデータの完全なセット	Name1 = 空白 Name2 = パスが HC 上にあるファイル名
21	1 つのツールのツールステータスデータ 種類 1: ツールデータの完全なセット	Name1 = ID 番号，Duplo 番号 Name2 = パスが HC 上にあるファイル名

22	1 つのツールのツールステータスデータ 種類 2: ツールデータの簡略セット	Name1 = ID 番号，Duplo 番号 Name2 = パスが HC 上にあるファイル名
23	1 つのツールのツールステータスデータ 種類 3 ツールデータの簡略セット	Name1 = ID 番号，Duplo 番号 Name2 = パスが HC 上にあるファイル名
24	アダプタ番号付きのツールのツールデータ ツールデータの完全なセット	Name1 = アダプタ番号 Name2 = パスが HC 上にあるファイル名
26	ツールのオプション／手動ローディング ツールデータの完全なセット	Name1 = ID 番号，Duplo 番号 Name2 = パスが HC 上にあるファイル名
27	ツールのオプション／手動アンローディング	Name1 = ID 番号 Duplo 番号 Name2 = パスが HC 上にあるファイル名
28	ツールをツールパレットからロードする	Name1 = ツールパレット番号 Name2 = ツールステータスデータが入ったファイルの名称
29	ツールをツールパレットにアンロードする	Name1 = ツールパレット番号 Name2 = ツールステータスデータが入ったファイルの名称
50	マシン割当てデータ	Name1 = 空白 Name2 = パスが HC 上にあるファイル名
90	任意のファイルを転送する これ以上加工しない - オプション	Name1 = MMC 上のファイル名 Name2 = パスが HC 上にあるファイル

5.7.1 ファイルとしてのデータのリクエスト（MMC からのリクエスト）

ファンクションコール

T_DATA_M (ホスト,
マシン,
OrderNum,
SFct,
Name1,
Name2)

転送方向： ホストコンピュータ → MMC

意味

ファイルとしてのデータを MMC からリクエストする。

データ

表 5.9

パラメータ	説明	フォーマット
ホスト	ホストコンピュータ（ホスト）の名称	16 バイト（文字列）
マシン	マシン名	16 バイト（文字列）
OrderNum	オーダ番号	4 バイト（long.int）
SFkt	サブファンクション番号	4 バイト（long.int）
Name1	名称，表 5-8 サブファンクション番号：SFct を参照	128 バイト（文字列）
Name2	追加名称，表 5-8 サブファンクション番号：SFct を参照	128 バイト（文字列）

例

T_DATA_M ("FLR1", "BAZ3", 0, 1, "\mpf.dir\ Kw15.mpf", "\0");
T_DATA_M ("FLR1", "BAZ3", 0, 10, "\mpf.dir", "\0");
T_DATA_M ("FLR1", "BAZ3", 0, 20, "\0", "\0");
T_DATA_M ("FLR1", "BAZ3", 0, 21, "Drill10mm,0002", "\0");
T_DATA_M ("FLR1", "BAZ3", 0, 22, "Drill10mm,0002", "\0");
T_DATA_M ("FLR1", "BAZ3", 0, 23, "Drill10mm,0002", "\0");
T_DATA_M ("FLR1", "BAZ3", 0, 50, "\0", "\0");
T_DATA_M ("FLR1", "BAZ3", 0, 90, "c:\mmc2\sincom.log", "\0") ;

5.7.2 ファイルとしてのデータのリクエスト（ホストコンピュータからのリクエスト）

コールされたファンクション

T_DATA_H (ホスト ,
マシン,
OrderNum,
SFct,
Name1,
Name2)

転送方向： MMC → ホストコンピュータ

意味

ファイルとしてのデータをホストコンピュータからリクエストする。

データ

表 5.10 データリクエスト用のパラメータ

パラメータ	説明	フォーマット
ホスト	ホストコンピュータ（ホスト）の名称	16 バイト（文字列）
マシン	マシン名	16 バイト（文字列）
OrderNum	オーダ番号	4 バイト（long.int）
SFkt	サブファンクション番号	4 バイト（long.int）
Name1	名称，表 5-8 サブファンクション番号：SFct を参照	128 バイト（文字列）
Name2	追加名称，表 5-8 サブファンクション番号：SFct を参照	128 バイト（文字列）

5.8 転送されたファイルの受取り

5.8.1 マシンに対するオーダ：データの受取り

ファンクションコール

R_DATA_M (ホスト ,
マシン,
OrderNum,
SFct,
Name1,
Name2,
日時,
LastFile)

転送方向： ホストコンピュータ → MMC

意味

ホストコンピュータは，指定されたファイルをホストコンピュータから読出し，処理するよう SINCOM にリクエストします（たとえば，データ管理システムにファイルを入れるために）。

データ

表 5.11 受取りリクエスト用のパラメータ

パラメータ	説明	フォーマット
ホスト	ホストコンピュータ（ホスト）の名称	16 バイト（文字列）
Machine	マシン名	16 バイト（文字列）
OrderNum	オーダ番号	4 バイト（long.int）
SFkt	サブファンクション番号	4 バイト（long.int）
Name1	名称，表 5-8 サブファンクション番号：SFct を参照	128 バイト（文字列）
Name2	追加名称，表 5-8 サブファンクション番号：SFct を参照	128 バイト（文字列）
日時	最終変更日時 （ユニックス時間，NC プログラムファイル用）	4 バイト（long.int）
LastFile	ワークの最後のファイル 0: フォローすべきファイルがまだ他にある 1: ワークの最後のファイルあるいは単一ファイル	4 バイト（long.int）

例

R_DATA_M ("FLR1", "BAZ3", 0, 1, "\mpf.dir\ Kw15.mpf",
"f:\ncpro\NCKW0815.txt", 862826400, 1);
R_DATA_M ("FLR1", "BAZ3", 0, 10, "Main programs", "f:\tmp\NCList.txt", 0, 1);
R_DATA_M ("FLR1", "BAZ3", 0, 26, "Drill10mm,0002", "f:\tmp\wzfile.txt");
R_DATA_M ("FLR1", "BAZ3", 0, 27, "TP003", " f:\tmp\tp003.txt ");
R_DATA_M ("FLR1", "BAZ3", 0, 28, "TP003", " f:\tmp\tp003.txt ");
R_DATA_M ("FLR1", "BAZ3", 0, 1001, "c:\mmc2\oemdata.txt",
"c:\tmp\oemdata.txt");

ファイルは R_DATA_M 中にホストコンピュータから同期して読み出されますが，処理を RPC 中で同期して実行することはできません。したがって，RPC の戻り値はファイル転送が成功したかどうかを示すことしかできません。ファイルの処理後，SINCOM は，タイプ = 5 およびエラー番号 = R_DATA_M からのサブファンクション番号の R_REPORT_H を確認応答として送信します。

5.8.2 ホストコンピュータに対するオーダ：データの受取り

コールされたファンクション

R_DATA_H (ホスト,
マシン,
OrderNum,
SFct,
Name1,
Name2,
日時 ,
LastFile)

転送方向： MMC → ホストコンピュータ

意味

すでに転送されたファイルは，ホストコンピュータ上のデータ管理システムに保存されるべきです。

データ

表 5.12 受取りリクエスト用のパラメータ

パラメータ	説明	フォーマット
ホスト	ホストコンピュータ（ホスト）の名称	16 バイト（文字列）
マシン	マシン名	16 バイト（文字列）
OrderNum	オーダ番号	4 バイト（long.int）
SFkt	サブファンクション番号	4 バイト（long.int）
Name1	名称，表 5-8 サブファンクション番号：SFct を参照	128 バイト（文字列）
Name2	追加名称，表 5-8 サブファンクション番号：SFct を参照	128 バイト（文字列）
日時	最終変更日時 （ユニックス時間，NC プログラムファイル用）	4 バイト（long.int）
LastFile	ワークの最後のファイル 0: フォローすべきファイルがまだ他にある 1: ワークの最後のファイルあるいは単一ファイル	4 バイト（long.int）

5.9 データの削除

5.9.1 MMC 上でのデータの削除

ファンクションコール

C_DELETE_M (ホスト,
マシン,
OrderNum,
SFct,
Name1,
Name2)

転送方向： ホストコンピュータ → MMC

意味

以前のファイル転送のデータを削除する

データ

表 5.13 削除リクエスト用のパラメータ

パラメータ	説明	フォーマット
ホスト	ホストコンピュータ（ホスト）の名称	16 バイト（文字列）
マシン	マシン名	16 バイト（文字列）
OrderNum	オーダ番号	4 バイト（long.int）
SFkt	サブファンクション番号	4 バイト（long.int）
Name1	削除されるファイルの名称	128 バイト（文字列）
Name2	追加名称	128 バイト（文字列）

（注）現在，データ管理中のファイルにしかアクセスできません。
SFct = 1
例：Name1 ="\mpf.dir\cylinderhead.mpf"

例

C_DELETE_M ("FLR1", "BAZ3", 0, 1, "\mpf.dir\ Kw15.mpf", "\0");

5.10 NC プログラム

次に説明するプログラム処理は，5.7 「データダイアログ」で説明されたファンクションの特殊な用途です。これらのファンクションについて知っていることが前提とされます。

5.10.1 NC プログラムのリクエスト（ホストコンピュータからのリクエスト）

1. ファンクションコール

T_DATA_M()

SFct = 1

Name1 = データ管理中のプログラム名，例：

\mpf.dir\carrier4711.mpf

転送方向 ホストコンピュータ → MMC

意味

ホストコンピュータが特定のプログラムを MMC からリクエストします。

2. ファイル転送

リクエストされた NC プログラムの入ったファイルが転送されます。

3. コールされたファンクション

R_DATA_H()

SFct = 1

Name1 = データ管理システム中のプログラム名

Name2 = ホストコンピュータ上のファイルのパスを含む名称

日時 = 最終変更日時

転送方向 MMC → ホストコンピュータ

意味

MMC が特定のプログラムをホストコンピュータにリクエストします。

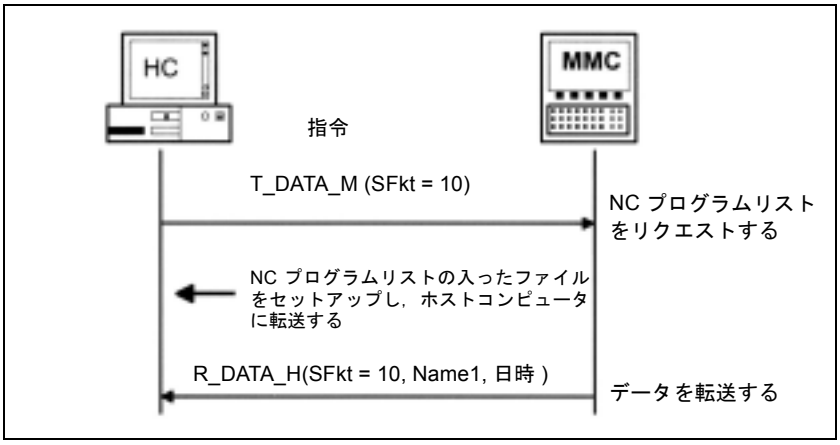


図 5.3 NC プログラムのリクエスト（ホストコンピュータからのリクエスト）

5.10.2 NC プログラムのリクエスト（MMC からのリクエスト）

1. コールされたファンクション

T_DATA_H ()

SFct = 1

Name1 = データ管理システム中のプログラム名

転送方向 MMC → ホストコンピュータ

意味

MMC が特定の NC プログラムをホストコンピュータにリクエストします。

2. ファンクションコール

R_DATA_M ()

SFct = 1

Name1 = データ管理システム中のプログラム名

Name2 = ホストコンピュータ上のファイルのパスを含む名称

日時 = 最終変更日時

転送方向 ホストコンピュータ → MMC

意味

ホストコンピュータは MMC のために特定の NC プログラムを準備します。

3. ファイル転送

SINCOM はリクエストされた NC プログラムのファイルを MMC とデータ管理に転送します。

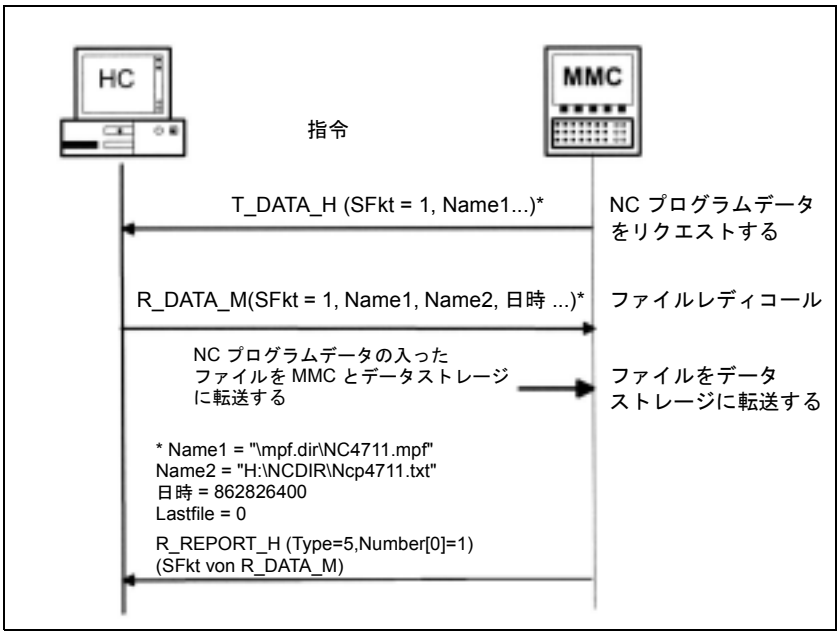


図 5.4 NC プログラムのリクエスト（MMC からのリクエスト）

（注）NC プログラムのリクエストおよび転送は個別にしか実行できません。

ワークキャリア指定は，'\0' で終了しなければならず，その長さは '\0' を含めて最大 6 バイトにしなければなりません。

NC プログラムには常にデータ管理パスが含まれていなければなりません。

例：NCProg = "\mpf.dir\cylinderhead.mpf"

MMC 中のデータ管理はバージョンを識別しないので，最終変更日時およびファイルサイズを表示させることができます。NC プログラムが MMC のデータ管理に含まれているが，サイズあるいは変更日時が異なる場合，プログラムを起動させる前に，SINCOM はそのファイルをホストコンピュータにリクエストしなければなりません。

5.10.3 NC プログラムの転送

ファンクションコール

R_DATA_M ()

SFct = 1

Name1 = データ管理システム中のプログラム名

Name2 = ホストコンピュータ上のファイルのパスを含む名称

日時 = 最終変更日時

転送方向 ホストコンピュータ → MMC

意味

ホストコンピュータが MMC のために特定のプログラムを準備します。

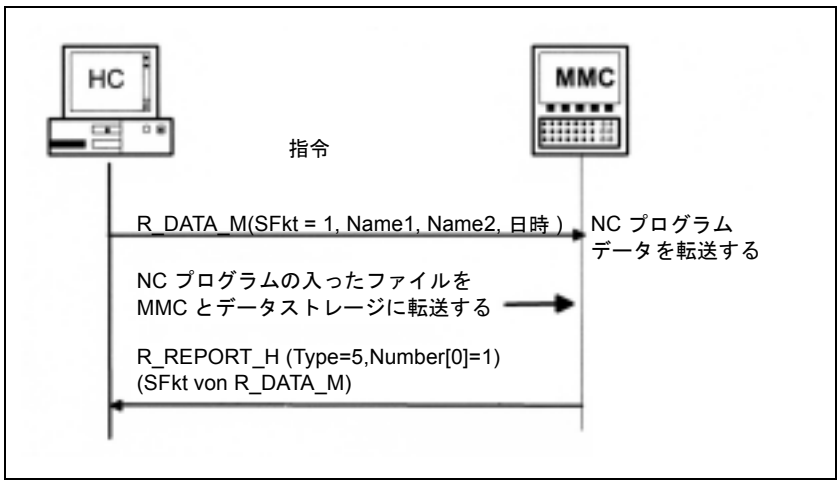


図 5.5 NC プログラムの転送（ホストコンピュータからの転送）

コールされたファンクション

R_DATA_H()

SFct = 1

Name1 = データ管理システム中のプログラム名

Name2 = ホストコンピュータ上のファイルのパスを含む名称

日時 = 最終変更日時

転送方向 MMC → ホストコンピュータ

意味

MMC がホストコンピュータのために特定のプログラムを準備します。

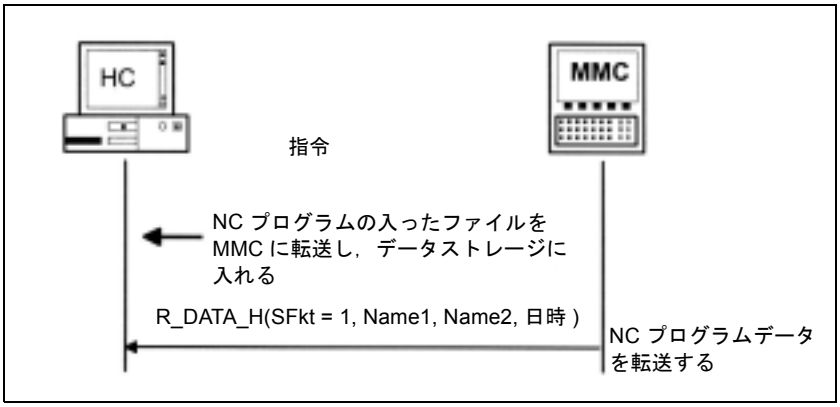


図 5.6 NC プログラムの転送（MMC からの転送）

5.10.4 マシン上での NC プログラムの削除

ファンクションコール

C_DELETE_M (SFkt = 1, Name1)

転送方向 ホストコンピュータ → MMC

意味

ホストコンピュータは，Name1 によって指定されたプログラムを削除するよう MMC に指示します。

データ

渡されるデータについては C_DELETE_M() の説明の箇所ですでに説明済みです。次のパラメータが割当てられなければなりません。

SFct = 1, Name1 = "\mpf.dir\cylinderhead.mpf" または
"\spf.dir\4711.spf"

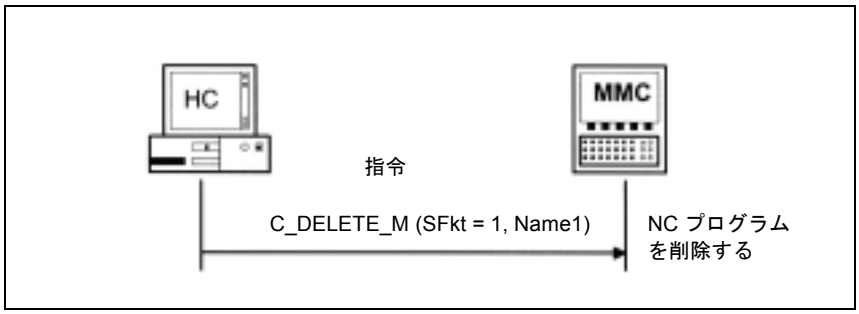


図 5.7 NC プログラムの削除（ホストコンピュータから開始された削除）

5.10.5 現行の NC プログラムのリストのリクエスト（ホストコンピュータからのリクエスト）

1. ファンクションコール

T_DATA_M() with

SFkt = 10

Name 1 = データ管理中のパス（例，"\mpf.dir"）

転送方向　　ホストコンピュータ → MMC

意味

ホストコンピュータは，現行のプログラムのリストを転送するようマシンにリクエストします。

2. ファイル転送

NC プログラムリストの入ったファイルがホストコンピュータに転送されます。

3. コールされたファンクション

R_DATA_H()

SFkt = 10

Name1 = データ管理システム中のパス

Name2 = NC プログラムリストを有するファイル名

転送方向　　MMC → ホストコンピュータ

意味

マシンからホストコンピュータへ転送：現行の NC プログラムのリスト。

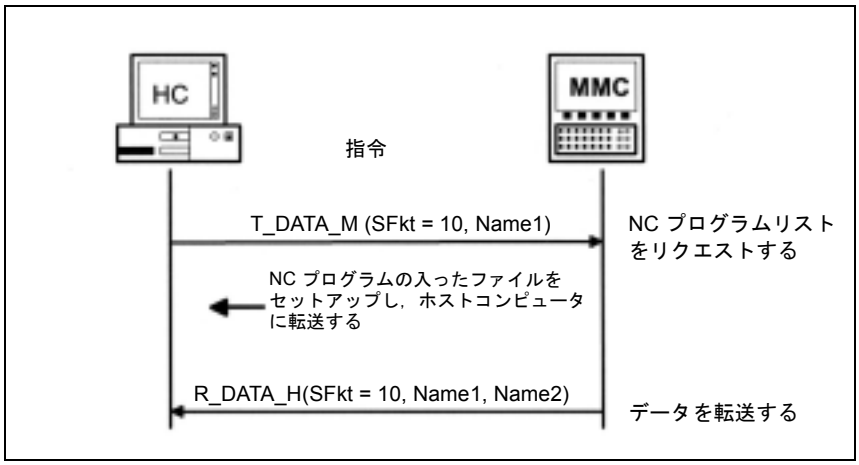


図 5.8 現行の NC プログラムのリストのリクエスト
（ホストコンピュータからのリクエスト）

5.10.6 現行の NC プログラムのリストのリクエスト （MMC からのリクエスト）

1. コールされたファンクション

T_DATA_H ()

SFct = 10

Name 1 = データ管理中のパス（例，"\mpf.dir"）

転送方向　　ホストコンピュータ → MMC

意味

マシンからホストコンピュータへ転送：
現行の NC プログラムのリストの転送。

2. ファンクションコール

R_DATA_M ()

SFct = 10

Name1 = データ管理中のパス

Name2 = NC プログラムリストを有するファイル名

転送方向　　ホストコンピュータ → MMC

意味

提供された NC プログラムリストを受取るようマシンにリクエストします。

3. ファイル転送

SINCOM は NC プログラムリストの入ったファイルを MMC に転送します。

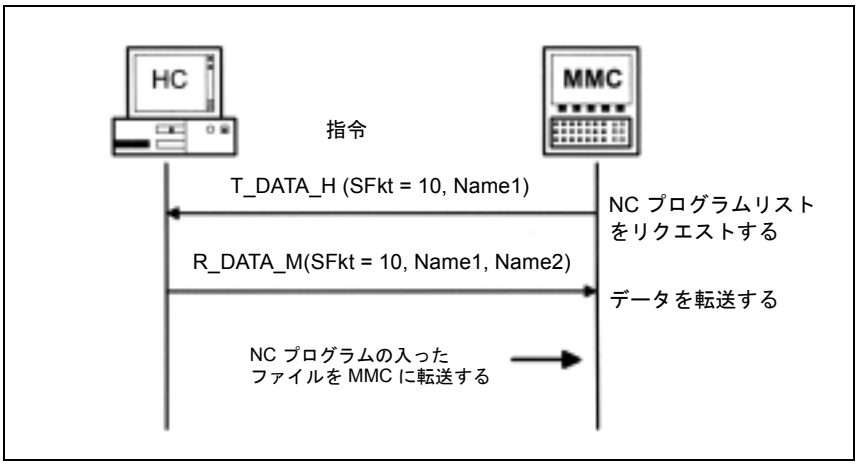


図 5.9 現行の NC プログラムのリストのリクエスト（MMC からのリクエスト）

5.10.7 NC プログラムリストの転送

ファンクションコール

R_DATA_M()

SFkt = 10

Name1 = データ管理システム中のパス

Name2 = NC プログラムリストを有するファイル名

転送方向　　ホストコンピュータ → MMC

意味

ホストコンピュータからマシンへ転送：現行のプログラムのリスト

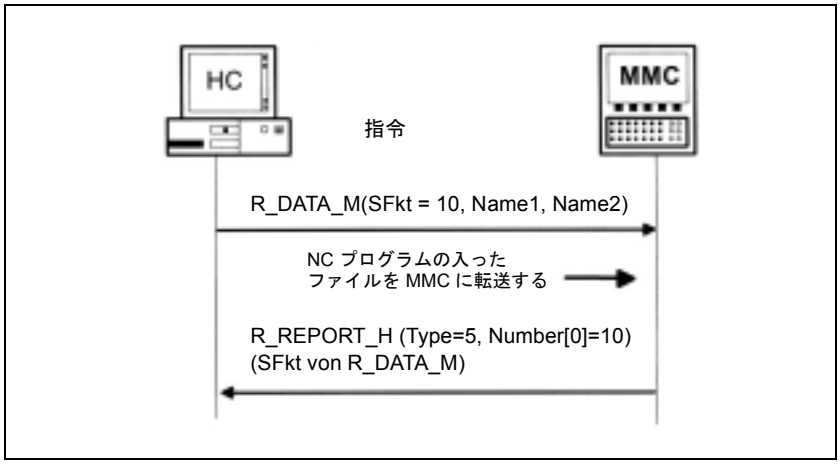


図 5.10

コールされたファンクション

R_DATA_H()

SFct = 10

Name1 = データ管理システム中のパス

Name2 = NC プログラムリストを有するファイル名

転送方向　　MMC → ホストコンピュータ

意味

マシンからホストコンピュータへ転送：現行のプログラムのリスト

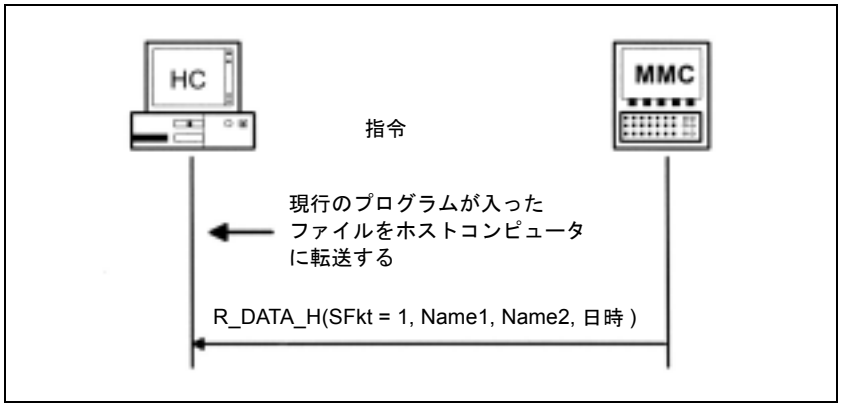


図 5.11

NC プログラムリストの入ったファイルには，ファイルのリストと，このデータ管理パスに含まれるサブディレクトリが含まれます。これらのサブディレクトリのコンテンツはリストに表示されないので，必要に応じて，個別のリクエストで決定されなければなりません。

R_DATA_H で返されたファイルの構造

\mpf.dir

Cylinderhead.MPF,FM,5320,876403708

Crankshaft.MPF,FN,8300,862826400

行

ファイルの最初の行は，Name1 について記述されたディレクトリとそのコンテンツを示しています。以降の行ではそれぞれファイルまたはサブディレクトリ名と追加情報をコンマで区切って示しています。

列

最初の列は，NC プログラム名あるいはサブディレクトリの名称を示しています。

2 番目の列には 2 つの文字が含まれます。

- 最初の文字は，それがファイル (F) であるのか，サブディレクトリ (D) であるのかを示します。
- 2 番目の文字は，そのファイルが MMC 中にあるのか，NCK 中にあるのかを示します。

例：

"Fx" - ファイル

"Dx" - ディレクトリ

"xM" - MMC 上

"xN" - NCK 中あるいは NCK および MMC 中

3 番目の列はファイルサイズをバイトで表示します。

4 番目の列は，1970 年 1 月 1 日からの経過時間を秒で表したファイルの日時（UNIX 時間）を 10 進数で表します。

シリンダヘッド 876403715 の時刻は，1997 年 10 月 9 日 15:28:35 を意味します。クランクシャフト 862826400 の場合は，1997 年 5 月 5 日 12:00:00 を意味します。

5.11 ツールダイアログ

ツールデータは常にファイルフォーマットで転送されます。

ファイル構造については 4.1「ツールデータ」ですでに説明済みです。

5.11.1 完全なツールマガジン割当てのスキャン

表 5.14

転送方向	指令	意味
ホストコンピュータ → MMC	T_DATA_M (SFkt = 20)	すべてのツールデータをリクエストする
ホストコンピュータ ← MMC		すべてのツールステータスデータが入ったファイルをホストコンピュータに転送する
ホストコンピュータ ← MMC	R_DATA_H (SFkt = 20, Name1 = 空白, Name2= すべてのツールステータスデータが入ったファイル)	ファイルの到着をホストコンピュータにレポートする

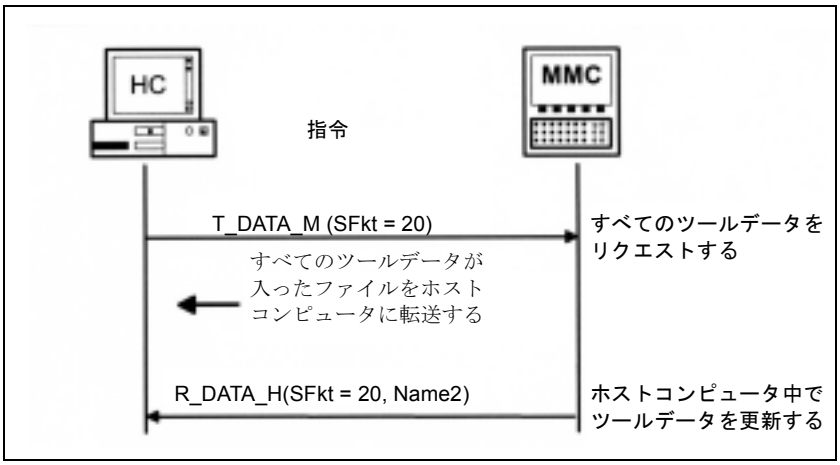


図 5.12 全ツールのスキャン（ホストコンピュータから開始されたスキャン）

5.11.2 ツールアダプタ番号付きのツールデータ（オプション）

表 5.15

転送方向	指令	意味
ホストコンピュータ ← MMC	T_DATA_H (SFct = 24, Name1= アダプタ番号)	アダプタ番号付きのツールデータをリクエストする
ホストコンピュータ → MMC	R_DATA_M (SFct = 24, Name1= アダプタ番号, Name2= ツールステータスデータを有するファイル名)	アダプタ番号付きのツールデータを転送する

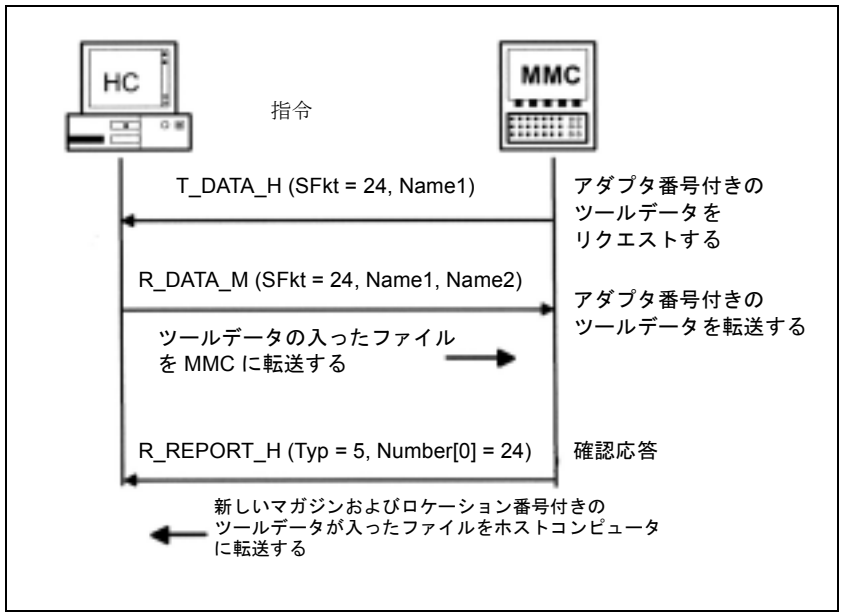


図 5.13

5.11.3 オプション／手動ローディング

表 5.16

転送方向	指令	意味
ホストコンピュータ ← MMC	T_DATA_H (SFkt = 26, Name1= ID 番号，Duplo 番号)	ツールデータをリクエストする
ホストコンピュータ → MMC	R_DATA_M (SFkt = 26, Name1= ID 番号，Duplo 番号，Name2= ツールステータスデータを有するファイル名)	ファイルを読み出すよう MMC にリクエストする。MMC 上でツールステータスデータの入ったファイルを読み出す。ツールをロードする。
ホストコンピュータ ← MMC	R_DATA_H (SFkt = 21, Name1= ID 番号，Duplo 番号 Name2= ツールステータスデータを有するファイル名)	新しいマガジンおよびロケーション番号付きのツールステータスデータが入ったファイルをホストコンピュータへ（これはロード確認応答である）

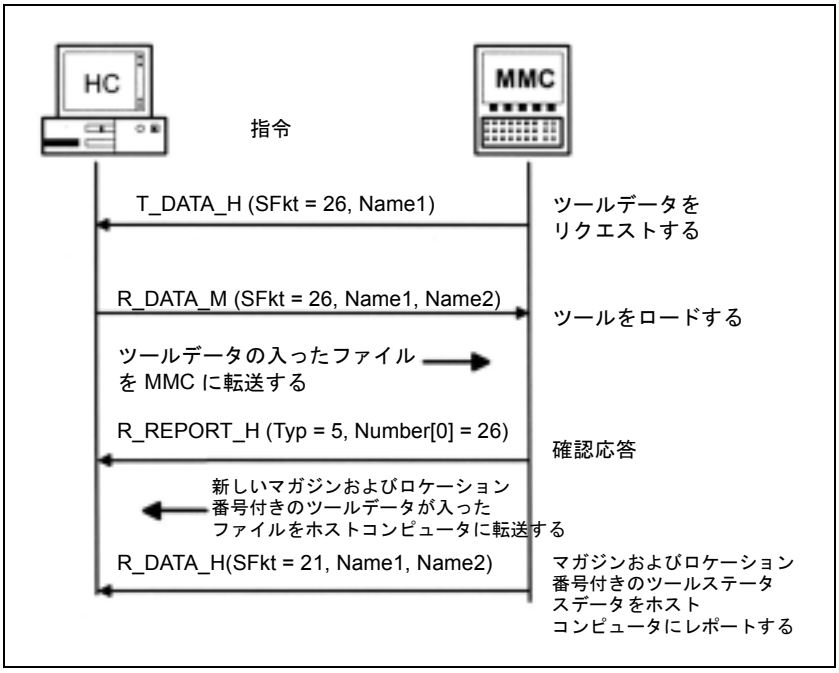


図 5.14 ツールのロード（MMC から開始されたロード）

5.11.4 オプション／手動アンローディング

表 5.17

転送方向	指令	意味
ホストコンピュータ ← MMC	R_DATA_H (SFkt = 27, Name1= ID 番号，Duplo 番号 Name2= ツールステータスデータを有するファイル名)	新しいマガジンおよびロケーション番号付きのツールステータスデータが入ったファイルをホストコンピュータへ （アンロード確認応答）

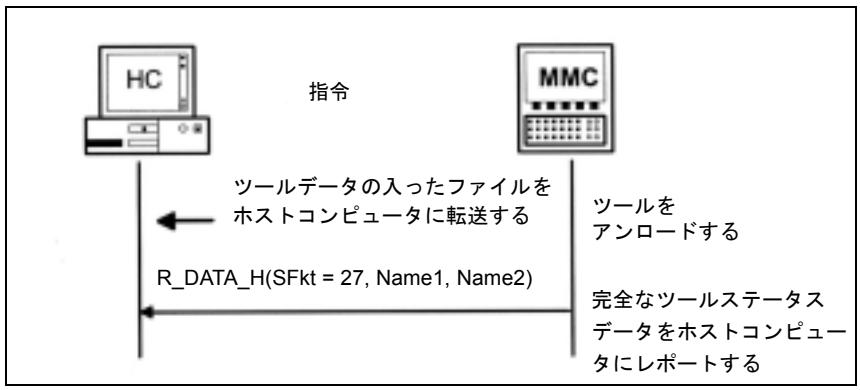


図 5.15 ツールのアンロード（MMC から開始されたアンロード）

5.11.5 ツールのレポート

表 5.18

転送方向	指令	意味
ホストコンピュータ ← MMC	R_DATA_H (SFkt = 21, Name1= ID 番号，Duplo 番号 Name2= ツールステータスデータを有するファイル名)	ツールステータスデータの入ったファイルをホストコンピュータに転送する （ツールのレポート）

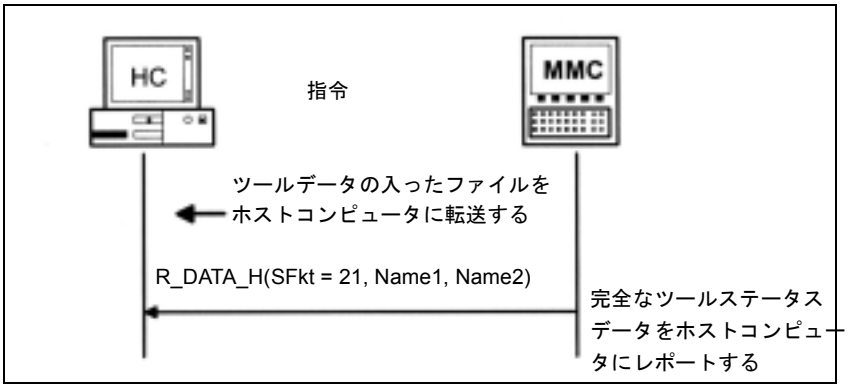


図 5.16 ツールのレポート（MMC からのレポート）

5.11.6 ツールパレット／カセットのロード （V1.0 には実装されていないオプション）

表 5.19 ツールパレットがマシンに到着する

転送方向	指令	意味
ホストコンピュータ→ MMC	R_DATA_M (SFkt = 27, Name1= ツールパレット番号, Name2= ツールステータスデータを有するファイル名)	MMC 上でツールステータスデータの入ったファイルを読み出す
ホストコンピュータ ← MMC	R_DATA_H (SFkt = 20, Name1= ID 番号，Duplo 番号 Name2= ツールステータスデータを有するファイル名)	新しいマガジンおよびロケーション番号付きのツールステータスデータが入ったファイルをホストコンピュータへ

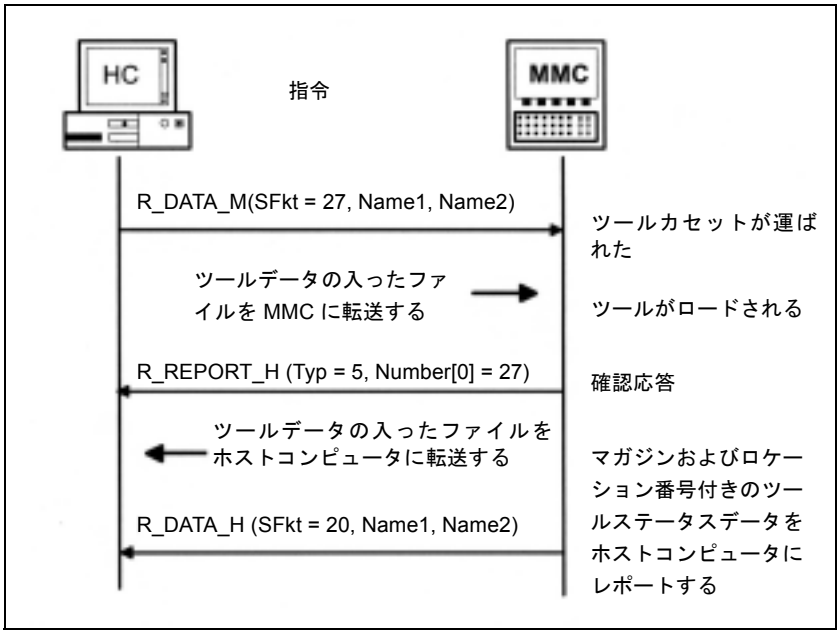


図 5.17 ツールパレット／カセットのロード

5.11.7 ツールパレット／カセットのアンロード （V1.0 には実装されていないオプション）

表 5.20 ツールパレットがマシンに到着する

転送方向	指令	意味
ホストコンピュータ → MMC	R_DATA_M (SFkt = 28, Name1= ツールパレット番号, Name2= ツールステータスデータを有するファイル名)	MMC 上でツールステータスデータの入ったファイルを読み出すツールがアンロードされる
ホストコンピュータ ← MMC	R_DATA_H (SFkt = 20, Name1= ID 番号，Duplo 番号) Name2= ツールステータスデータを有するファイル名)	新しいマガジンおよびロケーション番号付きのツールステータスデータが入ったファイルをホストコンピュータへ

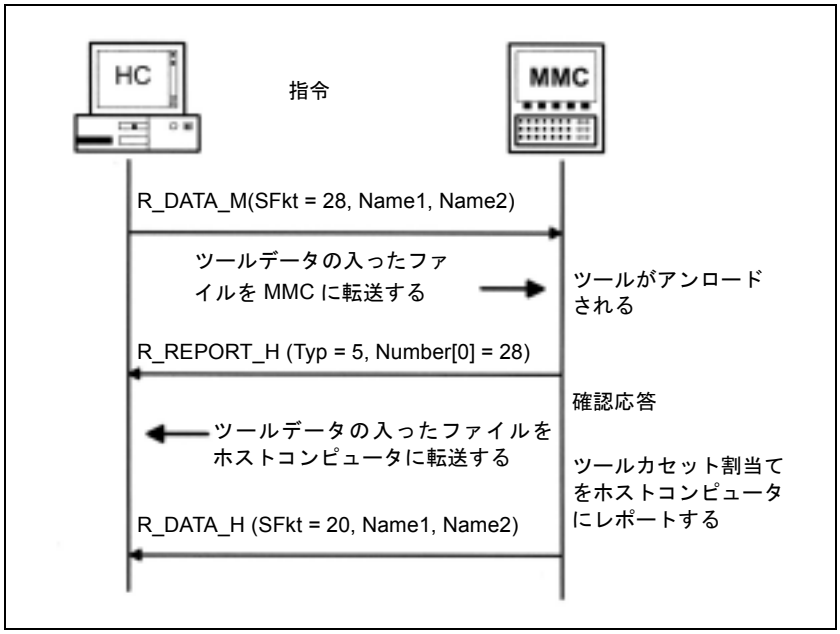


図 5.18 ツールパレット／カセットのアンロード

5.12 マシン割当てデータ

ドッキング位置／ストレージロケーションが 3 つ以上あるマシンの場合，たとえば，ストレージロケーションが複数あるトランスポートシステムが使用される場合，すべてのドッキング位置／ストレージロケーションのデータは ASCII ファイルで転送されます。このファイルがホストコンピュータに転送されると，RPC コール R_DATA_H (SFkt = 50) が開始されます（5.8.2 「ホストコンピュータに対するオーダ：データの受取り」を参照)。

各ドッキング位置／ストレージロケーションについて，

- ドッキング位置番号
- ワークキャリア番号，および
- ワークキャリアステータス

が転送されます。

これらのデータフィールドは，次のようにコンマで区切ってファイルに保存されます。

ASCII ファイルの構造

DockPos 1, DockPosStatus, WPC, WPCStatus < ラインフィード >

DockPos 2, DockPosStatus, WPC, WPCStatus < ラインフィード >

...

DockPos n, DockPosStatus, WPC, WPCStatus <EOF>

表 5.21 ファイルパラメータの説明

パラメータ	説明	フォーマット
Dockpos	ドッキング位置番号	ASCII
DockPosStatus	ドッキング位置ステータス 0: 利用可能 1: トランスポート制御系についてディスエーブル 2: 故障	ASCII
WPC	ワークキャリア名	ASCII
WPCStatus	ワークキャリアステータス 1: 加工されない，プログラムが割当てられていない 2: 加工されない，プログラムは割当てられている 16: 進行中 32: 終了 64: エラーを伴って終了 128: バッファリング用	ASCII

ホストコンピュータは，T_DATA_M(SFct = 50) を使用して，これらのデータをリクエストすることができます。

5.13 MODE 選択

ファンクションコール

C_MODE_M (ホスト ,
マシン,
OrderNum,
モード)

転送方向 : **ホストコンピュータ → MMC**

意味

特定の動作モードを起動させるようマシンに指示する。

データ

表 5.22 モード選択パラメータ

パラメータ	説明	フォーマット
ホスト	ホストコンピュータ（ホスト）の名称	16 バイト（文字列）
マシン	マシン名	16 バイト（文字列）
OrderNum	オーダ番号	4 バイト（長い整数）
モード	モード 1: 特殊モードオン 2: 特殊モードオフ 3: ユニットのスイッチオフ 4: ユニットのスイッチオン	4 バイト（長い整数）

例

C_MODE_M ("FLR1", "BAZ3", 0, 3);

5.13.1 特殊モード

説明

特殊モードの場合，自動的な材料の流れは，ワークキャリアがマシンに運ばれないということを意味します。ワークキャリアの到着を示すメッセージが送られても，ホストコンピュータはプログラム割当てを実行しません。ワークキャリアを運ぶのに，手動トランスポートアクションを使用することができます。ユーザーは手動で NC プログラムを選択およびスタートさせなければなりません。

自動的な材料の流れによって運ばれたワークキャリアは，特殊モードでも継続して自動的に運ばれます。手動トランスポートアクションによって運ばれたワークキャリアは，再度手動で運ばれなければなりません。

マシン用の特殊モードは，ホストコンピュータ上でいつでも起動させることができます。進行中の加工オペレーションは通常の方法で終了します。

特殊モードが起動されると直ちに加工されるワークキャリアを手動で運ぶことができます。マシンが停止する必要はありません。

- モード = 1 特殊モードの起動
- モード = 2 特殊モードの停止

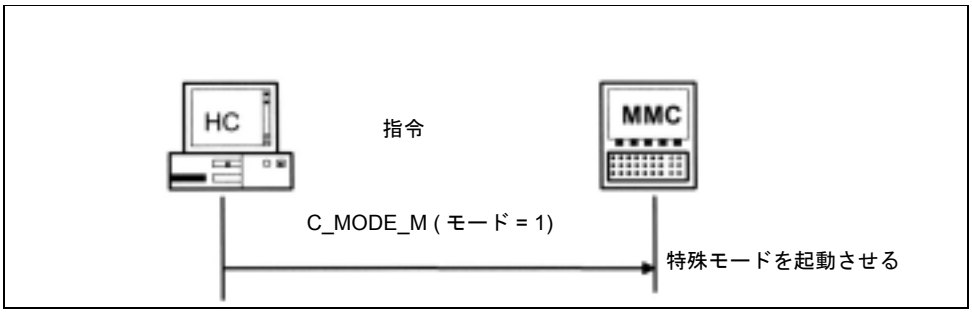


図 5.19 特殊モードの起動（ホストコンピュータから開始された特殊モードの起動）

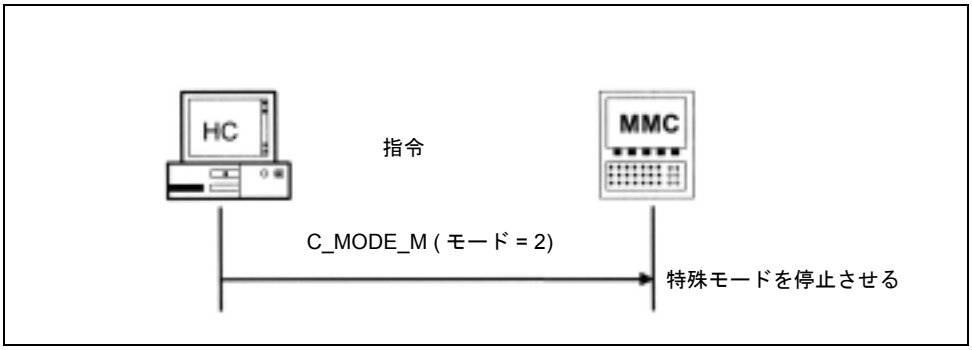


図 5.20 特殊モードの停止（ホストコンピュータから開始された特殊モードの停止）

5.13.2 コンポーネントの起動／停止

ホストコンピュータからドライブまたは他のプラントコンポーネントを停止させるには，作業終了時にリクエストが必要となります。起動リクエストも同様に作業開始時に必要となります。

停止リクエストが特定の時刻に行われても，最後のワークキャリアが加工された後に行われても，コンピュータリンクには関係がありません。コンピュータリンクは DB インタフェースを介して PLC にリクエストを送るだけです。PLC は，加工が進行中でないことを確認してからコンポーネントを停止しなければなりません。

コンポーネントの停止後，R_MACHINE () を使用して，この状態をホストコンピュータにレポートしなければなりません。同じことが起動後にも適用されます。

- モード = 3 コンポーネントの停止
- モード = 4 コンポーネントの起動

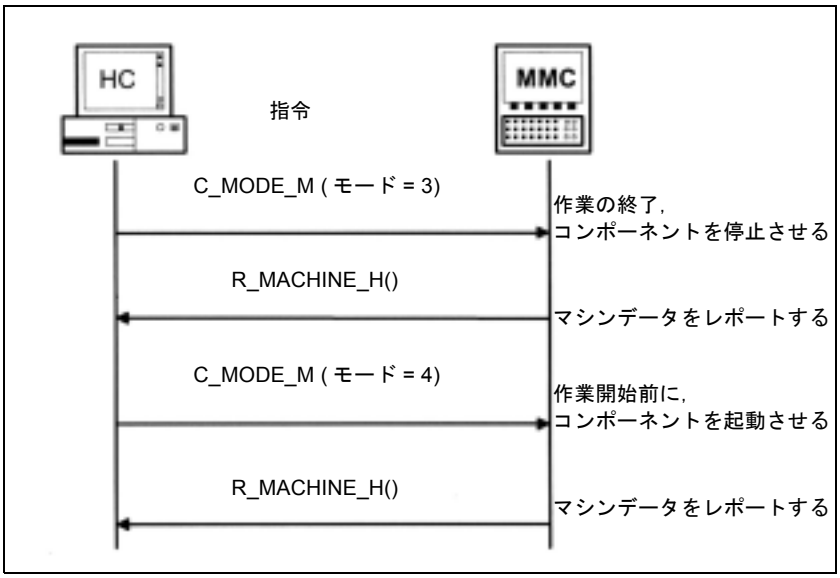


図 5.21 作業の終了（ホストコンピュータから開始された作業の終了）

5.14 同期

同期とは，ホストコンピュータ上のシステムイメージが実際の状況と一致するように，現在のデータをホストコンピュータに送信することを意味します。コンピュータまたはマシンがリスタートされた場合に，あるいは接続が中断された後に，同期が必要となります。

同期中，マシンは新しい加工オペレーションをスタートさせてはなりません。現在進行中のオペレーションは同期プロシージャの影響を受けません。

ホストコンピュータは，T_MACHINE_M () を使用して，マシンステータスデータをマシンにリクエストし，T_DATA_M (SFkt = 50) を使用して，マシン割当てデータをリクエストします。次に，ホストコンピュータは，まだ仕上がっていないワークキャリアの全サイドについてプログラム割当て R_NC4WPC_M () を転送します。

ホストコンピュータがツール準備／計算に対して責任がある場合，接続の（非常に長い）中断後に，すべてのツールがスキャンされるべきです。スキャンは同期後に実行されるべきで，これによりホストコンピュータが最新のツールデータにアクセスできることが保証されます。ツールスキャンは自動的には実行されません。T_DATA_M (SFct =20) を使用して，ホストコンピュータがスキャンをリクエストしなければなりません。

5.14.1 同期スタート／終了

ファンクションコール

C_SYNCH_M (ホスト ,
マシン,
OrderNum,
SynchFlag)

転送方向： **ホストコンピュータ → MMC**

意味

ホストコンピュータが同期プロセスを開始または終了します。

データ

表 5.23 同期パラメータの説明

パラメータ	説明	フォーマット
ホスト	ホストコンピュータ（ホスト）の名称	16 バイト（文字列）
マシン	マシン名	16 バイト（文字列）
OrderNum	オーダ番号	4 バイト（long.int）
SynchFlag	スタート／終了フラグ 1: スタート 0: 終了	4 バイト（long.int）

使用に関する注記

次のセクションでは，同期プロセス中に生じる相互作用について説明します。

例

C_SYNCH_M ("FLR1", "BAZ3", 0, 1);

5.14.2 同期プロシージャ

ホストコンピュータ → MMC	C_SYNCH_M ()	SynchFlag =1 // スタート
ホストコンピュータ → MMC	T_MACHINE_M ()	
ホストコンピュータ ← MMC	R_MACHINE_H ()	

ストレージロケーションが 3 つ以上あるマシンあるいはトランスポートシステムの場合，以下のプロシージャが続きます。

ホストコンピュータ→ MMC	T_DATA_M ()	SFkt = 50
ホストコンピュータ← MMC	マシン割当てファイルを転送する	
ホストコンピュータ← MMC	R_DATA_H () SFkt = 50, Name 1 = ファイル名	

まだ仕上がっていないワークキャリアすべてについて（一回ごと）

ホストコンピュータ → MMC	プログラム割当て	R_NC4WPC_M ()
ホストコンピュータ → MMC	C_SYNCH_M ()	SynchFlag = 0 // 終了

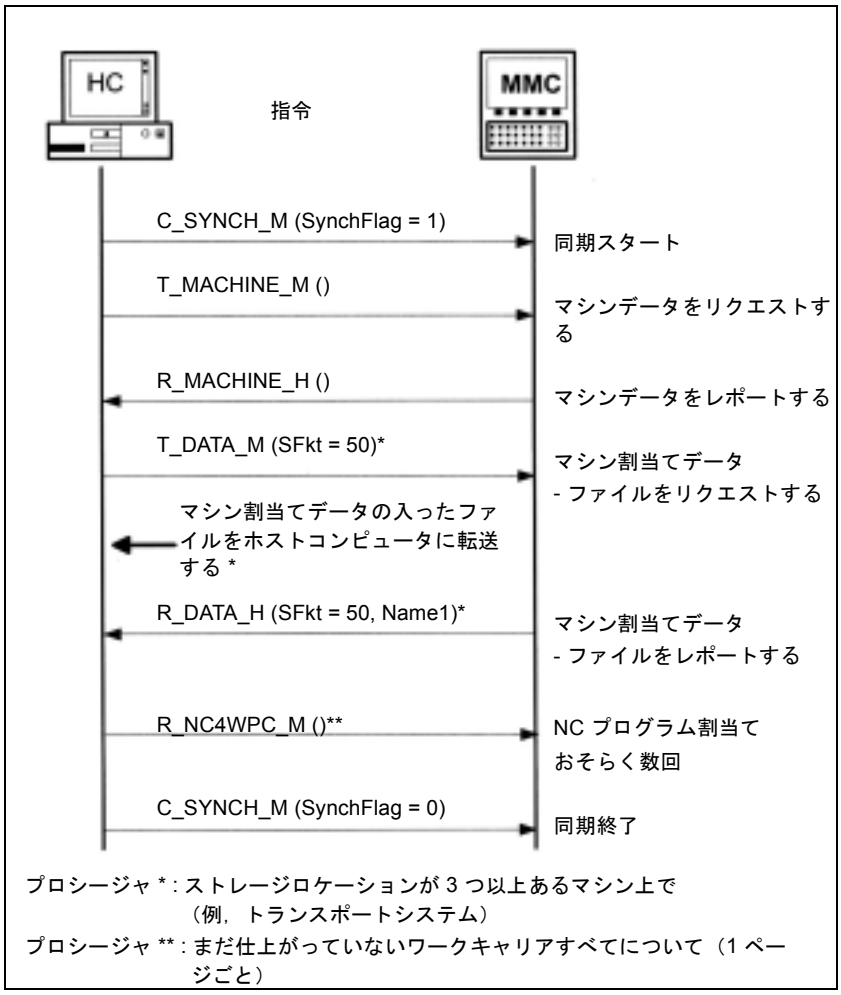


図 5.22　同期（ホストコンピュータから開始された同期）

6 OEM アプリケーション用のデータ通信

コンピュータリンクを通して，MMC 上の任意の OEM アプリケーションとホストコンピュータとの間でデータを送受信するには，MMC 上のコンピュータリンクソフトウェアが他のアプリケーションと通信できなければなりません。

MMC 上のプログラム間の通信は動的データ交換 (DDE) を介して行われるので，コンピュータリンクと OEM アプリケーションとの間の通信にも DDE が使用されます。

ファンクションおよびインタフェースは，両方向へのデータ転送について定義されます。

6.1 OEM アプリケーションへのデータ転送

ファンクションコール

R_DDEDATA_M (　ホスト,
マシン,
OrderNum,
アプリケーション,
トピック,
アイテム,
データ)

転送方向：　**ホストコンピュータ → MMC**

意味

この RPC コールは，DDE を介して，指定された OEM アプリケーションに対するユーザーデータの転送を開始します。必要なパラメータはすべて R_DDEDATA_M で渡されます。

この RPC 中に DDE 接続を確立することは不可能なので，OEM アプリケーションへのデータ転送は，R_DDEDATA_M が返された後にしか実行できません。

1 R_DDEDATA_M () につき 1 つの DDE 接続がセットアップされます。ユーザーデータをコピーする前に，SINCOM は，'pipe'（パイプ）記号（'|' ASCII コード 124）が後に付くホスト名を OEM アプリケーション用のセンダー情報として挿入します。この完全な文字列は，DDE-POKE で OEM アプリケーションに転送され，その後，DDE 接続がクローズされます。

データが転送されると，SINCOM は，タイプ = 5 およびエラー番号 = 1000 である R_REPORT_H を R_DDEDATA_M のセンダーに対する確認応答として送信します。

（注）
2 進データではなく，ASCII データだけをユーザーデータとして転送することが許可されています。この文字列は 2 進の 0 で終わります。

データ

表 6.1 OEM アプリケーションに対するデータ転送用のパラメータ説明

パラメータ	説明	フォーマット
ホスト	ホストコンピュータ（ホスト）の名称	16 バイト（文字列）
マシン	マシン名	16 バイト（文字列）
OrderNum	オーダ番号	4 バイト（long.int）
アプリケーション	アプリケーション名	64 バイト（文字列）
トピック	DDE トピック	64 バイト（文字列）
アイテム	DDE アイテム	64 バイト（文字列）
データ	DDE ユーザーデータ	32 KB （文字列）

例

R_DDEDATA_M ("FLR1", "BAZ3", 0, "OEMAPP", "TOPIC1", "ITEM1", "Hello OEM Application");

6.2 OEM アプリケーションからホストコンピュータへのデータ転送

コールされたファンクション

R_DDEDATA_H(ホスト,
マシン,
OrderNum,
データ)

転送方向： **ホストコンピュータ ← MMC**

意味

この RPC は，コンピュータリンクが対応する DDE コールを受信すると，ホストコンピュータに送信されます。機能性とパラメータについて以下で説明します。

データ

表 6.2 ホストコンピュータに対するデータ転送用のパラメータ説明

パラメータ	説明	フォーマット
ホスト	ホストコンピュータ（ホスト）の名称	16 バイト（文字列）
マシン	マシン名	16 バイト（文字列）
OrderNum	オーダ番号	4 バイト（long.int）
データ	DDE ユーザーデータ	32 KB （文字列）

6.2.1 OEM アプリケーションとコンピュータリンクソフトウェアとの間の DDE

DDE を介して MMC 上のコンピュータリンクプログラムにデータを転送するために，OEM アプリケーションをイネーブルする場合，コンピュータリンクプログラムは DDE サーバの役割を果たします。

OEM アプリケーションは，次の DDE パラメータを使用して DDE 接続をセットアップし，POKE を使用してそのデータを渡さなければなりません。

- アプリケーション = Sincom,
- トピック = OEM,

- アイテム = SendData。

DDE コールで渡されたデータストリングは，データが転送されるホストの実際のユーザーデータよりも前に，ホスト名を指定しなければなりません。'pipe'（パイプ）記号（'|' 文字コード 124）は，セパレータとして使用されなければなりません（ホスト名｜ユーザーデータ）。

上記の R_DDEDATA_H () コールでデータが受信されると，指定されたホストにユーザーデータストリングが渡されます。ホスト名はコンピュータリンクに知られていなければなりません（5 章を参照）。

6.2.2 OEM アプリケーションからホストコンピュータへのファイル転送

OEM アプリケーションからコンピュータリンクを通ってホストコンピュータへファイルを転送するのに，DDE コールが使用されます。このコールは，すでに利用できる RPC ファンクション R_DATA_H を使用します。アプリケーション名とトピックは，上記の DDE コールの場合と同じです。ファイルはアイテムとして指定されるべきです。

- アプリケーション = Sincom,
- トピック = OEM,
- アイテム = PutFile。

データストリングには，R_DATA_H コールに必要とされるパラメータが含まれていなければなりません。'pipe'（パイプ）記号（'|' 文字コード 124）は，セパレータとして使用されなければなりません。

データストリングの構造：Host|SFkt|Name1|Name2|Date|LastFile。

SFkt は 1000 以上の数字でなければなりません。SFkt は，ホストコンピュータ上でのファイル割当てに使用されます。

- Name1 は MMC 上のファイル名です（ドライブとパスの付いた）
- Name2 はホストコンピュータ上のファイル名です（同様にドライブとパスの付いた）

日時はオプションです。日時には，ユニックス時間フォーマットで（秒単位で）表したファイルの日時／時刻が含まれます。

LastFile はオプションです。5 章を参照してください。

6.2.3 ホストコンピュータから OEM アプリケーションへのファイル転送

ホストコンピュータから MMC へファイルを転送するのに, R_DATA_M (SFkt = 90) が使用されます。SFkt 90 の場合，ファイルは単に MMC 上に読み出されるだけです。それ以上の処理は行われません。ファイルが受信されたことを OEM アプリケーションに通知するメッセージは，R_DDEDATA_M () を使用して，ホストコンピュータから送信できます。

OEM アプリケーション用のファイル転送が FTP を使用して行われる場合，このファイル転送はコンピュータリンクソフトウェアによってのみ処理されなければなりません。

MMC がネットワークを通ってホストコンピュータのドライブに直接アクセスできる場合，OEM アプリケーションはホストコンピュータからファイルを読出し，ホストコンピュータ自身にファイルをコピーすることができます。

7 構成可能なデータ転送／変数サービス

7.1 説明

SINCOM と PLC および NCK との間でデータを読書きするために，SINCOM は変数サービスを使用することができます。この読書きの制御は，NC-DDE サーバの変数サービスに基づいています。

ホストコンピュータは，T_VAR_M () を使用して，データをリクエストすることができます。次にそのデータは SINCOM によって取出され，R_VAR_H () でホストコンピュータに渡されます。ホストコンピュータも同様に R_VAR_M () を使用して，データを渡すことができます。そのデータは次に SINCOM によって書込まれます。

読取りジョブを記述する変数セットを Ini ファイルに保存することができます。SINCOM は，これらの変数が変化すれば自動的にホストコンピュータにレポートすることができます。

このようなファンクションを使用すると，PLC および NCK はプロジェクト別のデータを読書きできます。SINCOM を使用したデータの読書きは，OEM MMC パッケージの変数サービスに基づいています。OEM の説明の 8 章は付録に含まれていて，MMC と NCK/PLC との間のインタフェースについて説明しています。

コンピュータリンク (RK) 変数サービス用の Ini

RK 変数サービス用の INI ファイルの構造：

```
; 変数セットを識別する
[MeasuredValues]

; 個々の変数用のアイテムをリンクさせる
VAR01=/Plc/DataBlock/Word[c90,10]
VAR02=/Channel/GeometricAxis/actToolBasePos[3]
VAR03= ....

; アクセスモード
; 0= アクセスは行われません
; 1= HotLink が変数セット中のすべての変数に対して確立されます
; 2= HotLink が変数 VAR01 に対してセットアップされます
;    VAR01 が変化すると，他のすべての変数が LinkRequest で読取られます。
; 3= ハンドシェイク。変数 VAR01 が後に 0 でクリアされることを除いて，2 と同じ。
    0 でクリアされた後，この変数は一定の遅延時間後にしか再セットできません。あまりに早く再セットされると，ステータスの変化が検出できなくなる可能性があります。

モード =1
```

（注）$ 変数，たとえば，$TC_TP1 （T 番号）を介したデータへのアクセスは，このサービスでは実行されません。

転送されるデータを**セット**中の**変数**として構成することができます。

次の仕様が適用されます。

- 全セットで合計 50 の変数を定義できます。
- 1 セットあたりの変数の最大数は 10 です。
- 最高 10 のセットを定義できます。

これらの制限が守られていれば，どのような組み合わせでも使用できます。

定義は C:\ADD_ON ディレクトリ中の SCVARSET.INI file に保存されなければなりません。各変数セットは，Ini ファイル中にセクションを有し，各括弧に入れたセクション名，アクセスモード，ホスト名（オプション）および変数定義から構成されます。

ホスト名が指定されない場合，この RPC について構成されたすべてのホストに対して R_VAR_H が送信されます。

アクセスモード

0　ホットリンクは確立されません；ホストコンピュータからリクエストがあり次第 (T_VAR_M)，セットの変数が読取られるだけです。

1*　ホットリンクがセット中の各変数についてセットアップされます

2*　ホットリンクはセットの第 1 変数についてしかセットアップされません

3*　2 と同様に，ホットリンクはセットの第 1 変数についてしかセットアップされません。

- ホットリンクの最大数は全セットで 10 に制限されています。

ホットリンク後，すべての変数が読取られると，第 1 変数は 0 でクリアされます（ハンドシェイク）。変数 1 は，この目的で配列を定義することはできません。個々の変数を定義できるだけです。

* ホットリンクを使用すると，変数セットのすべての変数が読取られ，R_VAR_H で，ホストコンピュータに送信されます。

SCVARSET.INI ファイルのフォーマット：

[Set01]

モード =0

ホスト =FLR1

Var01=/Channel/Parameter/R[1]

Var02=/Channel/Parameter/R[5]

[Set02]

モード =3

ホスト =FLR2

Var01=/Plc/Datablock/Byte[c50,1]

Var02=/Plc/Datablock/Byte[c50,2]

Var03=/Plc/Datablock/Byte[c50,3,20]("!l%d,")

Var04=/Plc/Datablock/Byte[c50,4]

Var05=/Plc/Datablock/Byte[c50,5,20]("!l%d,")

Var06=/Plc/Datablock/Byte[c50,6]

Var07=/Plc/Datablock/Byte[c50,7,20]("!l%d,")

Var08=/Plc/Datablock/Byte[c50,8]

Var09=/Plc/Datablock/Byte[c50,9,20]("!l%d,")

Var10=/Plc/Datablock/Byte[c50,10]

[Set03]

モード=1

Var01=/Plc/Datablock/Byte[c51,0,10]("!l%d,")

Var02=/Plc/Datablock/Byte[c51,30,50]("!l%d,")

（注）$ 変数，たとえば，$TC_TP1 （T 番号）を介したデータへのアクセスはこのサービスでは実行されません。

（注）MMC ソフトウェアには，DDETEST.EXE というテストツールが含まれています。このツールを使用すると，NCDDE サーバ実行中に，変数アクセスをテストすることができます。この方法で，SCVARSET.INI で使用する予定の全変数をテストするべきです。
SCTEST セットアッププログラムには，ヘルプファイル Btss_gr.hlp （および英語用の Btss_uk.hlp）が含まれています。このヘルプファイルには変数アクセスに関連する情報が含まれます。

データ

表 7.1

パラメータ	説明	フォーマット
ホスト	ホストコンピュータ（ホスト）の名称	16 バイト（文字列）
マシン	マシン名	16 バイト（文字列）
OrderNum	オーダ番号	4 バイト（long.int）
VarMode	変数モード	4 バイト（long.int）
VarSet	変数セットの名称	16 バイト（文字列）
VarDescr *	データ説明	1024 バイト（文字列）
VarData *	ユーザーデータ	10 KB （文字列）

* データ説明およびユーザーデータ中の変数は ’pipe’（パイプ）記号（’|’ ASCII コード 124）で区切らなければなりません。

7.2 データの転送

7.2.1 マシンへの変数データの転送

ファンクションコール

R_VAR_M (ホスト,
マシン,
OrderNum,
VarMode
VarSetVarDescr
VarData)

転送方向： **ホストコンピュータ → MMC**

意味

ホストコンピュータから SINCOM へデータを転送します。次に，そのデータは PLC または NCK に書込まれます。

M コール R_VAR_M は，VarSet で，SCVARSET.INI からの変数セットの名称を指定するか，あるいは，VarDescr で，変数説明を指定することができます。

データが転送されると，SINCOM は，タイプ = 5 およびエラー番号 = 0 の R_REPORT_H を R_VAR_M のセンダーに対する確認応答として送信します。

データ

表 7.2

パラメータ	説明	フォーマット
ホスト	ホストコンピュータ（ホスト）の名称	16 バイト（文字列）
マシン	マシン名	16 バイト（文字列）
OrderNum	オーダ番号	4 バイト（long.int）
VarMode	変数モード	4 バイト（long.int）
VarSet	変数セットの名称	16 バイト（文字列）
VarDescr *	データ説明	1024 バイト（文字列）
VarData *	ユーザーデータ	10 KB （文字列）

データ説明およびユーザーデータ中の変数は，'pipe'（パイプ）記号（'|' ASCII コード 124）で区切らなければなりません。

例 1

SET03 が SCVARSET.INI に定義されます。

[Set03]

モード =1

ホスト =FLR2

Var01=/Plc/Datablock/Byte[c51,0,10]("!l%d,")

Var01=/Plc/Datablock/Byte[c51,30,50]("!l%d,")

R_VAR_M ("FLR1", "BAZ3", 0, 0, "Set03", "\0", "33|50");

例 2

R_VAR_M ("FLR1", "BAZ3", 0, 0, "\0",
"/Plc/DataBlock/Word[c50,20]|/Plc/DataBlock/Word[c60,30]|",
"33|50");

7.2.2 ホストコンピュータへの変数データの転送

コールされたファンクション

R_VAR_H (ホスト,
マシン,
OrderNum,
VarMode
VarSet
VarDescr
VarData)

転送方向： **ホストコンピュータ ← MMC**

意味

SINCOM は，PLC または NCK から読取ったデータをホストコンピュータに転送します。

データ

表 7.3

パラメータ	説明	フォーマット
ホスト	ホストコンピュータ（ホスト）の名称	16 バイト（文字列）
マシン	マシン名	16 バイト（文字列）
OrderNum	オーダ番号	4 バイト（long.int）
VarMode	変数モード	4 バイト（long.int）
VarSet	変数セットの名称	16 バイト（文字列）
VarDescr *	データ説明	1024 バイト（文字列）
VarData *	ユーザーデータ	10 KB （文字列）

* データ説明およびユーザーデータ中の変数は，'pipe'（パイプ）記号（'|' ASCII コード 124）で区切らなければなりません。

SCVARSET.INI で定義される変数セットがレポートされます。

例：

R_VAR_H ("FLR1", "BAZ3", 0, 0, "Set02", "\0", "33|50");

7.3 データのリクエスト

7.3.1 変数データのリクエスト（マシンに対するリクエスト）

ファンクションコール

T_VAR_M (ホスト,
マシン,
OrderNum,
VarMode
VarSet
VarDescr
VarData)

転送方向： **ホストコンピュータ → MMC**

意味

ホストコンピュータは，PLC または NCK からデータを読取るよう SINCOM にリクエストします。次に，そのデータが R_VAR_H () で返されます。VarSet は，SCVARSET.INI で定義された変数セットしか指定できません。VarDescr は評価されません。

データ

表 7.4

パラメータ	説明	フォーマット
ホスト	ホストコンピュータ（ホスト）の名称	16 バイト（文字列）
マシン	マシン名	16 バイト（文字列）
OrderNum	オーダ番号	4 バイト（long.int）
VarMode	変数モード	4 バイト（long.int）
VarSet	変数セットの名称	16 バイト（文字列）
VarDescr *	データ説明	1024 バイト（文字列）
VarData *	ユーザーデータ	10 KB （文字列）

* データ説明およびユーザーデータ中の変数は，’pipe’（パイプ）記号（’|’ ASCII コード 124）で区切らなければなりません。

例

_VAR_M ("FLR1", "BAZ3", 0, 0, "Set02", "\0", "\0");

7.3.2 変数データのリクエスト（ホストコンピュータに対するリクエスト）

コールされたファンクション

T_VAR_H (ホスト,
マシン,
OrderNum,
VarMode
VarSet
VarDescr
VarData)

転送方向： **ホストコンピュータ ← MMC**

意味

SINCOM はホストコンピュータにデータをリクエストします。データは R_VAR_M () で SINCOM に返されます。

（注）
SINCOM はこの RPC を現在使用していません。

8 ホストコンピュータとトランスポートシステム (TPS) との間の通信

8.1 TPS ／マシンインタフェース

トランスポートシステム (TPS) は，ホストコンピュータからトランスポートジョブを受信すると，そのジョブが（論理的および物理的に）実行可能かどうかをチェックしなければなりません。そのためには，トランスポートシステムは移動先が利用できること，およびディスエーブルされていないことを確認する必要があります。

ドッキング位置

TPS と工作機械との間の転送ポイントのことをドッキング位置と呼びます。

ドッキング位置には次の 3 つのタイプがあります。

1. 入力ドッキング位置（搬送のみ可能）
2. 出力ドッキング位置（回収のみ可能）
3. 入力／出力ドッキング位置（搬送と回収の両方が可能）

8.2 TPS ステータスデータ

コールされたファンクション

R_TPS_H（ ホスト,
マシン,
OrderNum,
モード,
MachineStatus,
TpoStatus,
DockPos,
DockPosStatus,
WPC,
Resint1,
Resint2,
Resbyte）

転送方向：ホストコンピュータ ← MMC

意味

TPS ステータスデータをホストコンピュータに送信する。

データ

表 8.1 トランスポートシステムステータスデータのパラメータ

パラメータ	説明	フォーマット
ホスト	ホストコンピュータ（ホスト）の名称	16 バイト（文字列）
マシン	TPS 名	16 バイト（文字列）
OrderNum	オーダ番号	4 バイト（long.int）
モード	動作モード ・YS 840DI モード： 1: 自動 2: MDA 4: JOG ・コンピュータ通信モード： 100: ホストコンピュータモード 300: 手動モード	4 バイト（long.int）
MachineStatus	マシンステータス 0: コールドリスタート 1: 非アクティブ 2: アクティブ 3: 故障 4: 作業の終了（ドライブ停止）	4 バイト（long.int）
TpOStatus	トランスポートジョブステータス 0: トランスポートジョブデータなし 1: 新しいトランスポートジョブ（スタートされない） 2: 進行中 4: WPC が伝達媒体上にある 8: ジョブ完了 16: エラー，ジョブ実行不可能 32: エラー，別の移動先がアプローチされる (DockPos)	4 バイト（long.int）
DockPos[2]	ドッキング位置番号 ドッキング位置番号は，1 から始まる，インタフェース DB のドッキング位置リスト中のインデックスに対応する。ドッキング位置番号 = 0 は " 割当てられていない " を意味する。	4 バイト（long.int）
DockPosStatus[2]	ドッキング位置ステータス 0: 利用可能 1: トランスポート制御系についてディスエーブル 2: 故障	4 バイト（long.int）
WPC[2]	ワークキャリア名	6 バイト（文字列）
Resint1 **	予備 1	4 バイト（long.int）
Resint2 **	予備 2	4 バイト（long.int）
Resbyte	予備 3	8 バイト（文字列）

* WPC は，2 次元文字配列と定義されます (char [2][6]) が，2 つのワークキャリア名のそれぞれは，'\0' で終わる文字列として指定されなければなりません。

** Resint1 および Resint2 は PLC の DB インタフェースに表示されます。PLC によって値が DB インタフェースに入力された場合，その値はホストコンピュータに転送されます。これらの値はコンピュータリンクに影響を与えません。ホストコンピュータに渡されるだけです。

使用に関する注記

ステータス変化およびモード選択のたびに，またトランスポートオペレーションの開始時および終了時に，TPS はこの RPC をセットします。

コンピュータリンクソフトウェアが MMC 上でスタートされると，この RPC はスタートアップメッセージとして，構成されたホストコンピュータのそれぞれに送られます。

8.3 TPS ステータスデータのリクエスト

ファンクションコール

T_TPS_M (ホスト,
マシン,
OrderNum)

転送方向：ホストコンピュータ → MMC

意味

MMC にトランスポートシステムステータスデータをリクエストする。

データ

表 8.2 TPS ステータスデータリクエストのパラメータ

パラメータ	説明	フォーマット
ホスト	ホストコンピュータ（ホスト）の名称	16 バイト（文字列）
マシン	TPS 名	16 バイト（文字列）
OrderNum	オーダ番号	4 バイト（long.int）

使用に関する注記

ホストコンピュータはこのコールを使用して，たとえば，同期中に，TPS ステータスデータをリクエストすることができます。TPS は R_TPS_H() を使用して，データを返します。

例

```
T_TPS_M ("FLR1", "BAZ3", 0);
```

8.4 トランスポートジョブ

ファンクションコール

C_TPORDER_M (ホスト,
マシン,
OrderNum,
SDockPos,
DDockPos,
WPC,
WPCType,
BufferFlag,
優先順位,
ChainNum,
伝達媒体,
Resint1,
Resint2,
Resbyte)

転送方向：ホストコンピュータ → MMC

意味

トランスポートジョブをマシンに送る

データ

表 8.3 トランスポートジョブのパラメータ

パラメータ	説明	フォーマット
ホスト	ホストコンピュータ（ホスト）の名称	16 バイト（文字列）
マシン	マシン名	16 バイト（文字列）
OrderNum	オーダ番号	4 バイト（long.int）
SdockPos	移動元ドッキング位置番号 ドッキング位置番号は，1 から始まる，インタフェースブロックのドッキング位置リスト中のインデックスに対応する。	4 バイト（long.int）
DdockPos	移動先ドッキング位置番号 ドッキング位置番号は，1 から始まる，インタフェースブロックのドッキング位置リスト中のインデックスに対応する。	4 バイト（long.int）
WPC	ワークキャリア名	6 バイト（文字列）
WPCTyp	ワークキャリアタイプ	4 バイト（long.int）
BufferFlag	バッファリングだけに使用される移動先	4 バイト（long.int）
優先順位	トランスポート優先順位	4 バイト（long.int）
ChainNum	ジョブチェーン番号	4 バイト（long.int）
伝達媒体	トランスポート伝達媒体番号	4 バイト（long.int）

パラメータ	説明	フォーマット
Resint1	予備 1	4 バイト（long.int）
Resint2	予備 2	4 バイト（long.int）
Resbyte	予備 3	8 バイト（文字列）

例

C_TPORDER_M ("FLR1", "BAZ3", 0, 3, 4, "WPC05", 0, 0, 0, 1, 0, 0, "\0");

移動元および移動先ドッキング位置

移動元および移動先ドッキング位置には，インタフェース DB のドッキング位置リスト中の対応するドッキング位置データレコードを表すインデックスが含まれます。インデックスは 1 から始まります。

ワークキャリア

妥当性をチェックするために，トランスポートシステムはワークキャリアの名称を使用することができます。

ワークキャリアタイプは，ワークキャリアのタイプまたはサイズを含めることのできるトランスポートシステム用の追加情報です。

パラメータ BufferFlag = 1 を使用すると，ホストコンピュータは，ワークキャリアがバッファリングだけの目的でマシンに運ばれ，そこ（補助バッファロケーション）で加工されることはないということをトランスポートシステムに通知することができます。必要に応じて，トランスポート制御系 (TPS) はこの情報をマシンに渡さなければなりません。他のすべての搬送については，BufferFlag = 0 がセットされなければなりません。

トランスポート優先順位

トランスポート優先順位は，複数のジョブがトランスポートシステムに転送される場合の追加情報です。ジョブが処理される順番を制御するのに優先順位を使用できます。

ジョブチェーニング

ジョブチェーニングは追加情報です。ストレージロケーションが 2 つあるトランスポート伝達媒体とドッキング位置が 1 つしかないマシンの場合，チェーン番号を使用して 2 つのトランスポートジョブを論理的に関連付けることができます。

たとえば，1 番目のトランスポートジョブには，加工されていないワークキャリアをマシンに運ぶことが含まれています。仕上がったワークキャリアがまだマシン上にあり，1 番目のトランスポートジョブと同じジョブチェーニング番号を持つ 2 番目のジョブには，加工されたワークキャリアをマシンから回収し，バッファまたはクランピングロケーションに運ぶことが含まれています。この場合，新しいパレットが取出されてマシンに運ばれ，加工されたワークキャリアはそのキャリッジの空いたロケーションに入れられます。その後初めて，加工されていないワークキャリアをマシンに返すことによって，1 番目のジョブを完了することができます。その後，トランスポートジョブ 2 の移動先まで加工されたワークキャリアを運び，このジョブも同様に完了することができます。

トランスポートキャリッジ

トランスポートキャリッジは補足情報です。トランスポート伝達媒体が複数あるトランスポートシステムの場合，ホストコンピュータはこのパラメータをセットして，どの伝達媒体を搬送に使用するかを決めることができます。

8.4.1 トランスポートシーケンス

表 8.4

転送方向	指令	注釈
ホストコンピュータ → MMC	C_TPORDER_M ()	
ホストコンピュータ ← MMC	R_TPS_H ()（オプション）	TPS アクティブ，空のストレージロケーション，および伝達媒体上の WPC をレポートする
ホストコンピュータ ← MMC	R_TPS_H ()	TPS 非アクティブ，WPC 上の新しいストレージロケーションおよび空の伝達媒体をレポートする

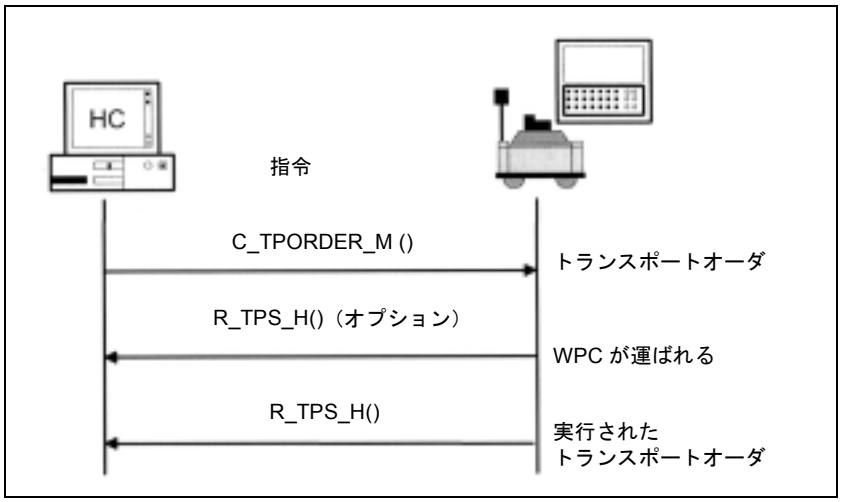


図 8.1 トランスポートダイアログ，通常の状況，エラーなし

8.4.2 トランスポートジョブ中のエラー

エラーが生じた場合，TPS はエラーメッセージをホストコンピュータに出力します。トランスポートジョブ中に生じる可能性のあるエラーは次のとおりです。

SINCOM で使用されるエラー番号：

表 8.5 トランスポートシステム中のエラー番号

番号	ファンクション
-200	トランスポートジョブをインタフェース DB に書込むことができない
-700	トランスポートジョブエラーがトランスポートシステムによってレポートされる。

8.5 トランスポートシステム (TPS) の同期

TPS の同期用のプロシージャは，マシンの場合と同じです。ホストコンピュータが TPS からのステータスおよび割当てデータをリクエストします。

同期中，トランスポートシステムはステータスが変化してもレポートしてはなりません。

表 8.6

転送方向	指令	注釈
ホストコンピュータ → MMC	C_SYNCH_M () SynchFlag = スタート	
ホストコンピュータ → MMC	T_TPS_M ()	

同期中にトランスポートジョブが操作可能であっても，このジョブの終了は同期後にしかレポートされません。

表 8.7

転送方向	指令	注釈
ホストコンピュータ ← MMC	R_TPS_H ()	
ホストコンピュータ → MMC	T_DATA_M () SFkt = 50	
ホストコンピュータ ← MMC		マシン割当てファイルをホストコンピュータに転送する
ホストコンピュータ ← MMC	R_DATA_H () SFkt = 50, Name 1 = ファイル名	
ホストコンピュータ → MMC	C_SYNCH_M () SynchFlag = 終了	

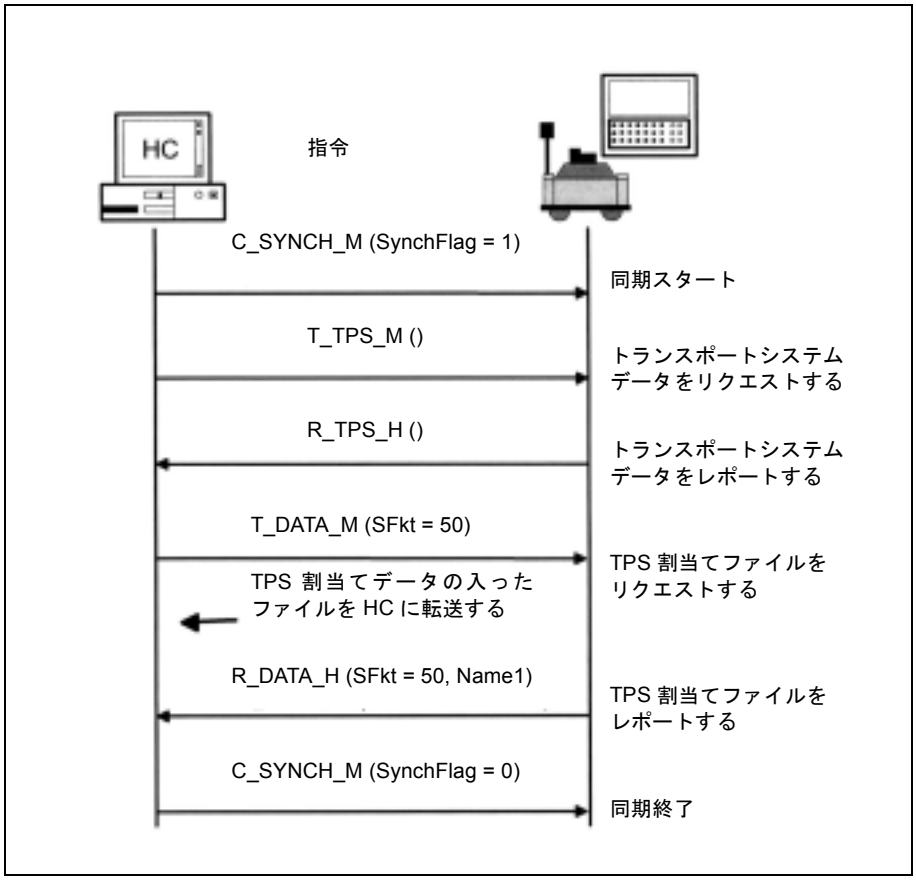


図 8.2 TPS 同期（ホストコンピュータから開始された TPS 同期）

9 RPC コールの要約

9.1 ホストコンピュータから YS 840DI へのファンクションコール

表 9.1

コール	説明
T_MACHINE_M ()	" サーバ " からのイニシエーションをクライアントに送信する
T_TPS_M ()	" サーバ " からのイニシエーションをクライアントに送信する
R_NC4WPC_M ()	クライアントに対するイニシエーションをリストに入れる
R_REPORT_M()	メッセージをマシンに転送する
C_DELETE_M ()	データ管理システム中のデータを削除する
C_MODE_M ()	インタフェース中のビットをセットする
C_SYNCH_M ()	インタフェース中のビットをセット／リセットする
C_TPORDER_M ()	クライアントに対するイニシエーションをリストに入れる
T_DATA_M ()	
R_DATA_M ()	
T_VAR_M ()	オプション
R_VAR_M ()	オプション
R_DDEDATA_M ()	
R_MESSAGE_M ()	V 1.0 以外

9.2 YS 840DI からホストコンピュータへのファンクションコール

表 9.2

コール	説明
R_MACHINE_H ()	マシンステータスデータをホストコンピュータに提供する
R_TPS_H ()	トランスポートシステムステータスデータをホストコンピュータに提供する
R_REPORT_H ()	メッセージをホストコンピュータに送信する
T_DATA_H ()	
R_DATA_H ()	
T_VAR_H ()	オプション
R_VAR_H ()	オプション
R_DDEDATA_H ()	
R_MESSAGE_H ()	V 1.0 以外

（注）

すべてのサブファンクション番号 (SFkt) を両方向で利用できるとは限りません。

表 9.3 サブファンクション番号 SFkt

サブファンクション番号	ファンクション	指令	注釈
1	NC プログラム	T_DATA_H () T_DATA_M () R_DATA_H () R_DATA_M ()	Name1 = NC プログラム Name2 = ホストコンピュータ上にパスがあるファイル名
10	現行の NC プログラムのリスト	T_DATA_M () R_DATA_H ()	
20	すべてのツールのツールステータスデータ 種類 1，ツールデータの完全なセット	T_DATA_M () R_DATA_H ()	Name1 = 空白 Name2 = パス付きのファイル名
21	1 つのツールのツールステータスデータ 種類 2，ツールデータの簡略セット	T_DATA_M () R_DATA_H ()	Name1 = ID 番号，Duplo 番号 Name2 = パス付きのファイル名
22	1 つのツールのツールステータスデータ 種類 3，ツールデータの簡略セット	T_DATA_M () R_DATA_H ()	Name1 = ID 番号，Duplo 番号 Name2 = パス付きのファイル名
23	すべてのツールのツールステータスデータ 種類 3，ツールデータの簡略セット	T_DATA_M () R_DATA_H ()	Name1 = ID 番号，Duplo 番号 Name2 = パス付きのファイル名
24	アダプタ番号付きのツールのツールデータ ツールデータの完全なセット	T_DATA_H R_DATA_M	
26	ツールのオプション／手動ローディング ツールデータの完全なセット，ツールデータ 1	T_DATA_H () R_DATA_M ()	
27	ツールのオプション／手動アンローディング	R_DATA_H	
28	ツールをツールパレットからロードする	T_DATA_H () R_DATA_M ()	
29	ツールをツールパレットにアンロードする	T_DATA_H () R_DATA_M ()	
50	マシン割当てデータ	T_DATA_M () R_DATA_H ()	
90	任意のファイル	T_DATA_M () R_DATA_H ()	

10 SINCOM-OCX

10.1 はじめに

SINCOM-OCX とは？

SINCOM-OCX 開発パッケージは，SINCOM Computer Link（SINCOM コンピュータリンク）用のアドオン製品です。

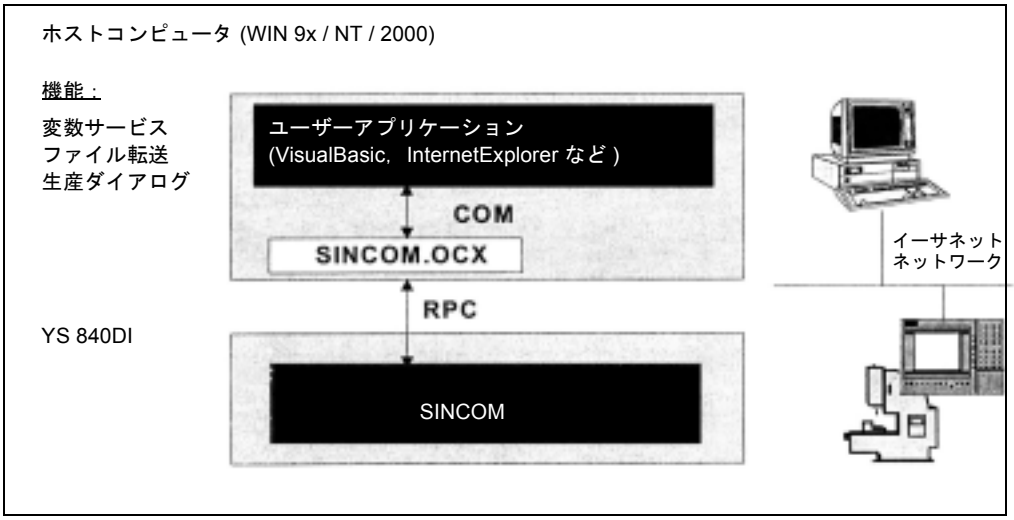
SINCOM Computer Link は，YS 840DI 制御装置とより上位のホストコンピュータとの間で通信を可能にするインタフェースを備えています。ホストコンピュータと制御装置との間の通信は，RPC (Remote Procedure Call) を介して行われます。RPC はプラットフォーム独立型なので，SINCOM インタフェースは MS Windows, UNIX などのどの OS でも使用できます。

アプリケーション

通常は，RPC のユーザーは C/C++ プログラミング言語を使用する必要がありますが、SINCOM-OCX を使用すると，C/C++ でプログラムすることなく普通の Windows 開発システムのすべてにアクセスできます。32 ビット ActiveX コンポーネントをリンクすることのできる開発システムはすべてサポートされています。たとえば、MS Visual Basic 4.0 (32 ビット) / 5.0 / 6.0，MS Visual J++ 6.0，Internet Explorer 4.0 / 5.0，WinDev などの開発システムがサポートされています。

動作原理

SINCOM.OCX を使用すると，SINCOM からの RPC インタフェースが COMcalls (Component Object Model) の中に組込まれます。SINCOM.OCX を使用しても RPC の機能には影響ありません。RPC の機能については，本資料の 5 章と 6 章で説明されています。

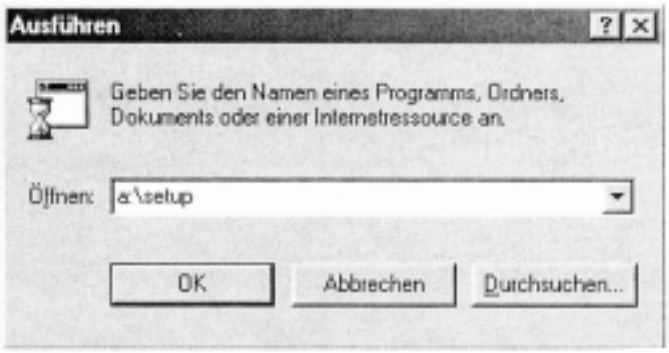


オペレーティングシステム

SINCOM.OCX は，TCP/IP ネットワークがインストールされている WIN 9x/NT/ 2000 コンピュータで使用できます。通信相手としては、SINCOM が入っている 1 つあるいは複数の YS 840DI 制御装置が必要です。

10.2 SINCOM-OCX 開発パッケージのインストール

SINCOM-OCX 開発パッケージをインストールするには，1 枚目のインストール用フロッピディスクの Setup.exe を実行します。

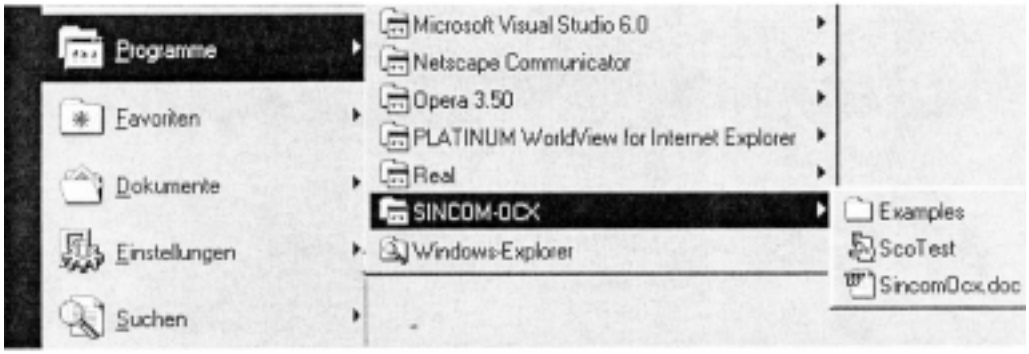


インストール中に，ターゲットディレクトリおよびスタートメニューのプログラムフォルダ名を入力するように促されます。

Doc	本マニュアルが入っているディレクトリ
Bin	SINCOM.OCX および ScoTest.exe アプリケーションが入っているディレクトリ
ScoTest	ScoTest アプリケーションのソースコードが入っているディレクトリ
Examples	SINCOM.OCX の使用例が入っているディレクトリ

また，Microsoft Visual Basic 6.0 (SP3) ランタイムシステムが <Windows>\System ディレクトリにインストールされています。

以下のようにして，スタートメニューからインストールされたこのファイルを見つけます：



テストおよび例

インストール後，ScoTest アプリケーションの SINCOM-OCX を使用して SINCOM インタフェースをテストできます。ScoTest アプリケーションを使用するために満足しなければならない要件については本章の 10.4 「ScoTest テストアプリケーション (SINCOM OCX test)」を参照してください。SINCOM とのリンクテストに成功したら，本章の 10.5 「SINCOM-OCX の使用例」に示されている例を使用することができます。

10.3 SINCOM.OCX コンポーネントの説明

SINCOM.OCX は 32 ビット ActiveX/COM コンポーネントとして実行されるので，そのようなコンポーネントを統合することのできる 32 ビット Windows 開発システムのすべてで使用できます。

SINCOM-OCX コンポーネントの別のインスタンスを，マシン制御装置への各リンクごとにユーザーアプリケーション中に定義しなければなりません。

10.3.1 インストール

SINCOM-OCX 開発パッケージをインストールすると，SINCOM.OCX ファイルが <Windows>\System ディレクトリに保存され，Windows レジストリに登録されます。

別のコンピュータにもインストールする場合

SINCOM.OCX コンポーネントを別のコンピュータでも使用する場合は，その使用するコンピュータに SINCOM-OCX 開発パッケージをインストールする必要があります。インストールするには InstallShield のようなインストールプログラムを使用するか，あるいは次のように手動で行います。

1. Sincom-ocx\bin ディレクトリの sincom.ocx ファイルをインストールするコンピュータの <Windows>\System ディレクトリにコピーします。
2. 次のコマンドを使用して，SINCOM-OCX コンポーネントを Windows レジストリに登録します。

   Regsvr32 <Windows>\System\sincom.ocx

SINCOM 制御装置への TCP/IP リンクも存在していなければなりません。

10.3.2 SINCOM-OCX コンポーネントの属性

SINCOM-OCX コンポーネントは，リンクを構成するのに使用される次の属性を備えています。

属性	説明	例
MachineID (マシン ID)	SINCOM 構成内のマシン名。マシン名は自由に選択できます。この情報は識別目的で RPC の通信相手へ送られます。	M1
MachineIP (マシン IP)	マシン制御装置の IP ネットワークアドレス。この情報は Windows ネットワークインスタレーションから得られます。スタティック IP 割当てが用いられます。この時点で，コンピュータのネットワーク名を SINCOM-OCX で使用することができます。	195.2.208.233
MachinePort (マシンポート)	TCP/IP 通信時にコンピュータ内のアプリケーションにアクセスするための追加情報。1000 ～ 64000 の範囲で自由に選択できます。マシン制御装置のポートナンバには 3011 をお勧めします。SINCOM 構成では，この情報をマシン終点と呼びます。	3011
MachineTimeout (マシンタイムアウト)	この値は、RPC を SINCOM へ送るときの時間応答を調整するために使用します。制御装置が起動していなかったなどで、RPC を SINCOM へ送ることができなかった場合の，RPC コールがエラーで中止されるまでの時間をこのタイムアウト値によって定義します。この属性には，0 ～ 9 までの値が使用できます。この値はタイム値ではなく，Microsoft RPC システムで定義された相対値です。 0 - 最小タイムアウト 5 - デフォルトタイムアウト 9 - 最大タイムアウト デフォルトタイムアウト値の (5) を使用することをお勧めします。	5
HostID (ホスト ID)	SINCOM 構成のホストコンピュータ名。ホストコンピュータ名は自由に選択できます。この情報は識別目的で RPC の通信相手へ送られます。	H1
HostPort (ホストポート)	TCP/IP 通信時にコンピュータ内のアプリケーションをアクセスするための追加情報。1000 ～ 64000 の範囲で選択できます。ホストコンピュータのポートナンバには 3010 をお勧めします。SINCOM 構成では，この情報をホスト終点と呼びます。	3010
HostEnabled (ホストイネーブル)	この属性は SINCOM-OCX コンポーネントが RPC を受取る用意ができているかどうかを示します。 True（**真**）- SINCOM-OCX は RPC を受取る用意ができている。 False（**偽**）- SINCOM-OCX は RPC を受取る用意ができていない。 ユーザーアプリケーションでこの属性を**真**に設定することによって，RPC を受取る用意ができた状態になります。また，少なくとも 1 回は RPC が SINCOM へ正常に転送された場合は，RPC を受取るための用意ができた設定になります。	True

ホスト ID，ホストポート，ホストイネーブルの属性は，アプリケーション（EXE)の SINCOM-OCX のすべてのインスタンスに応用されます。１つのインスタンスで行われた変更は，すべてのインスタンスにも同じ変更が行われます。

10.3.3 RPC を SINCOM へ送るメソッド

SINCOM.OCX を使用すると，SINCOM からの RPC インタフェースが COMcalls の中に組込まれます。SINCOM への RPC 送信は，SINCOM-OCX コンポーネントと同じ名前のメソッドを有するコールによってトリガされます。

例えば,

Ret = T_MACHINE_M (Host, Machine, OrderNum)

メソッド T_MACHINE_M が，OCX のインスタンスの１つに作成されます。

Ret = Machine1.T_MACHINE_M (OrderNum)

これらすべてのメソッドにおいて，1 番目と 2 番目のパラメータ（ホストおよびマシン）は省略可能です。これらのパラメータは，属性（当該インスタンスのホスト ID およびマシン ID）から導き出されます。

メソッドによって与えられた戻り値（Return values）については，本章の 10.3.6「エラー処理」を参照してください。

10.3.4 受取り用意の起動

RPC がマシンに少なくとも 1 回は正しく送られた場合，あるいは属性のホストイネーブルが真に設定されている場合は，SINCOM-OCX は SINCOM から RPC を受取る用意ができています。

10.3.5 SINCOM からの RPC の受取り

SINCOM からの RPC は，SINCOM-OCX コンポーネントのインスタンスのイベントとしてアプリケーションへ送られます。例えば，SINCOM からの RPC

T_DATA_H (Host, Machine, OrderNum, SFct, Name1, Name2)

は，以下に示すイベントとしてアプリケーションへ送られます。

TxDATAxH (OrderNum, SFct, Name1, Name2).

開発システムによっては，例えば Visual Basic では，イベント名に "_" 文字（下線）を認めていないので，RPC 名に含まれる "_" 文字は "x" に置換えられます。

10.3.6 エラー処理

SINCOM-OCX のメソッドで与えられるエラー番号は，次の 2 つのカテゴリに分類されます :

- Microsoft RPC システムからのエラーメッセージ
- SINCOM からのエラーメッセージ

Microsoft RPC システムからのエラーメッセージ

RPC を SINCOM へ送ることができない場合，あるいは SINCOM-OCX が RPC サーバを起動させることができない場合に，このエラーが生じます。このカテゴリのエラー番号は，1700 から 1938 の範囲に入ります（RPC ステータスコード)。

実際に生じるこの種のエラーについては，「**代表的なエラー状況**」の表で説明しています。詳細については，下記の Microsoft オンラインマニュアルで説明されています。

http://msdn.microsoft.com/library/

Platform SDK -> Networking and Directory services -> Remote Procedure Calls(RPC)

SINCOM からのエラーメッセージ

RPC が SINCOM へ正しく転送されているにもかかわらず，RPC の内容あるいは SINCOM の現在の状態が原因で正しく処理できなかった場合に，このエラーが生じます。このカテゴリのエラー番号は負の数になります。詳細については，本資料の「付録」の A「エラー番号」を参照してください。

次のようなエラーが起こり得ます :

代表的なエラー状況

ここでは，よく起こるエラー状況を示します :

エラー状況	RPC メソッドの戻り値
マシン IP 属性のエラー値（Wrong value）	1722 (RPC_S_SERVER_UNAVAILABLE)
マシンポート属性のエラー値	1722 (RPC_S_SERVER_UNAVAILABLE)
マシン ID 属性のエラー値	-100 (ERR_WRONG_MACHINE)
ホスト ID 属性のエラー値	-110 (ERR_WRONG_HOST)
ホストポート属性のエラー値	0，同時に SINCOM からの返答はなく，制御装置の SINCOMERR.LOG に 1 項目（an entry）が生成されます。
SINCOM-OCX コンポーネントが受取りの用意を起動できない。	1720 (RPC_S_CANT_CREATE_ENDPOINT) 同じコンピュータ内の別のアプリケーションがホストポートの属性に定義されたポートをすでに使用している場合に，通常このエラーは起こります。例えば，SINCOM-OCX を使用しているアプリケーションを起動させようとした時に，同じホストポート値を使用している ScoTest がすでに実行中の場合です。
TCP/IP リンクの中断	1722 あるいは 1726 (約 20 秒後) リンクを復旧させた後に，RPC コールは再び 0 を返します。

10.3.7 テスト接続の制限

VisualBasic の開発環境内で，VisualBasic がブレークポイント（break point）で中断された場合，イベントは OCX にトリガされません。

10.4 ScoTest テストアプリケーション（SINCOM OCX テスト）

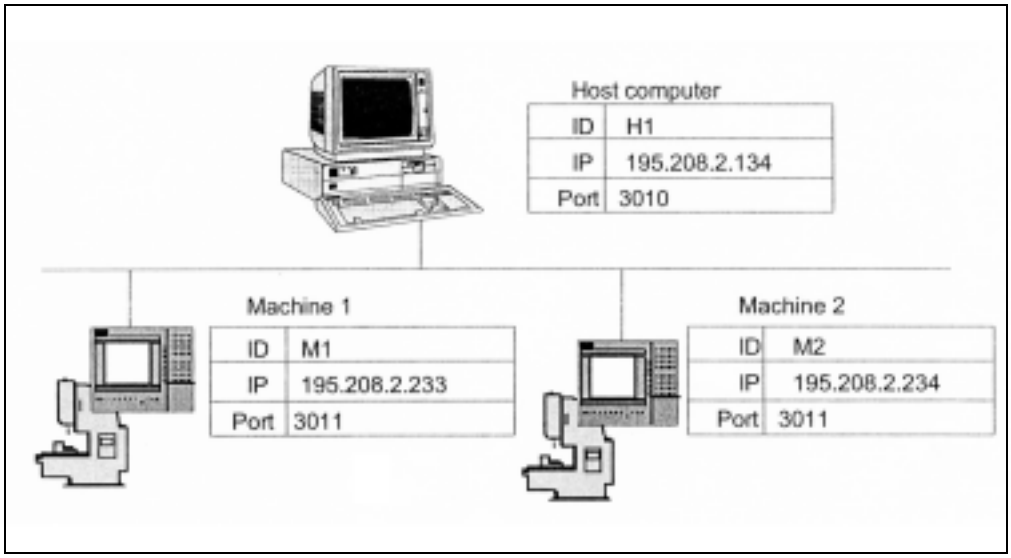
ScoTest アプリケーションによって，対話型のダイアログで SINCOM インタフェースの各 RPC を送受信できます。それぞれの制御装置をリンクすることができます。

このアプリケーションは，Visual Basic 6.0 で書かれています。また，このアプリケーションのソースコードは，Sincom-ocx\ScoTest ディレクトリへ送られます。詳しくは，10.4.4「ScoTest アプリケーションのソースコード」を参照してください。
ScoTest を使用するためには次の条件を満たす必要があります：

1. SINCOM パッケージがインストールされている YS 840DI が少なくとも 1 台は使用できなければなりません。
2. ホストコンピュータ（Windows パソコン）に，TCP/IP プロトコルがインストールされていなければなりません。
3. 制御装置とホストコンピュータ間にネットワークリンクが確立されていなければなりません。

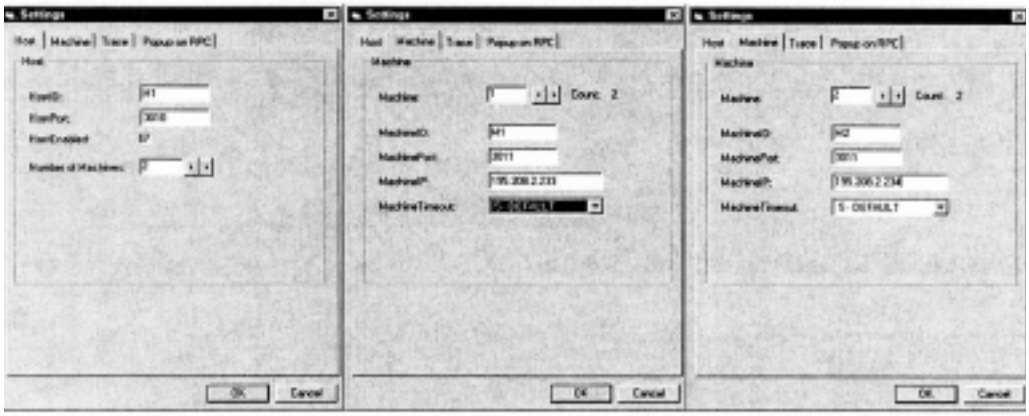
10.4.1 構成

ネットワーク構築の構成例：



ID	通信相手の名前は自由に選択できます。この情報は識別目的で RPC の通信相手に送られます。
IP	ホストコンピュータあるいは制御装置の IP ネットワークアドレス。この情報は Windows ネットワークインスタレーションから得られます。スタティック IP 割当てが用いられます。この時点で，コンピュータのネットワーク名を SINCOM-OCX で使用することができます。
ポート	TCP/IP 通信時にコンピュータ内のアプリケーションにアクセスするための追加情報。1000 ～ 64000 の範囲でポートナンバを自由に選択できます。ホストコンピュータのポートナンバには 3010，マシン制御装置には 3011 をお勧めします。SINCOM 構成では，この情報を終点と呼びます。

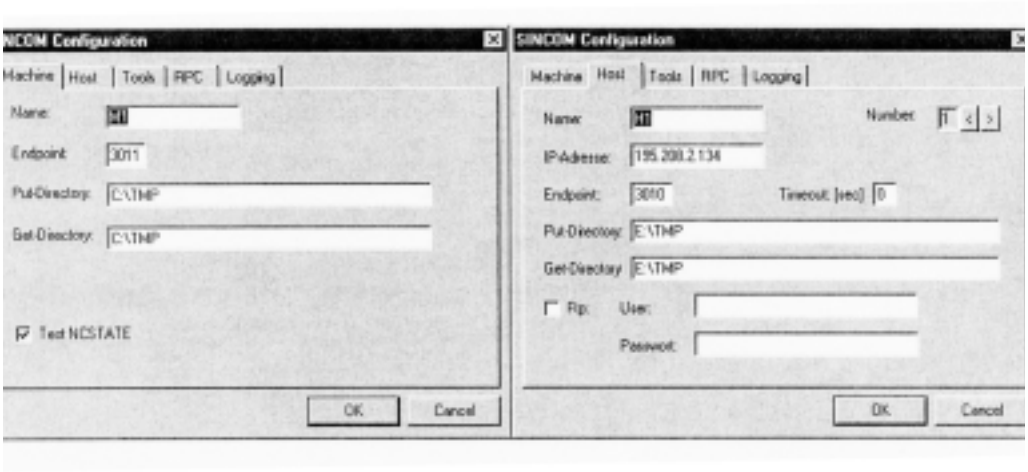
次の項目は，ScoTest の環境設定のネットワーク構築の画面です。環境設定は，メニューの [Settings] から呼出すことができます。



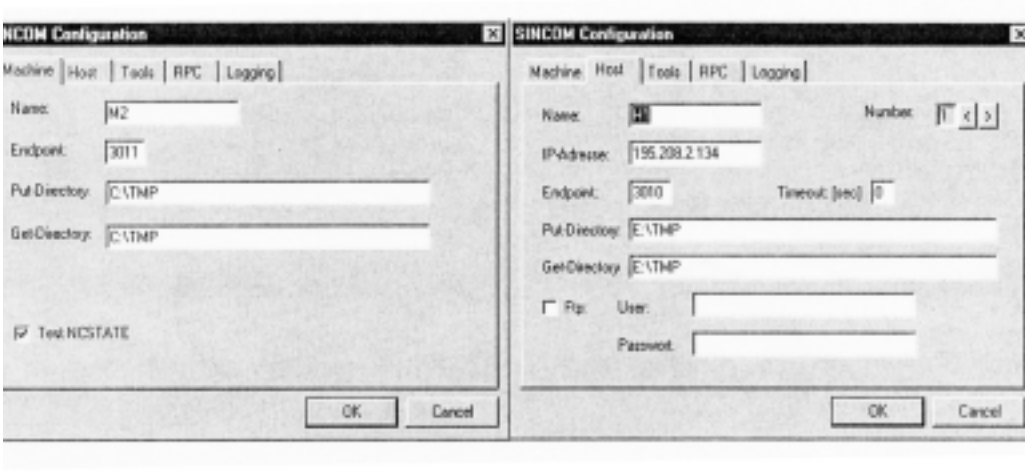
ホストイネーブルおよびマシンタイムアウトの入力フィールドには，それぞれの名前に関係する SINCOM-OCX インタフェースの属性が示されます。

制御装置間をリンクするには，アプリケーション scconfig を使用します。ネットワーク構築のために，次のように入力してください。scconfig の詳細については，前の章を参照してください。

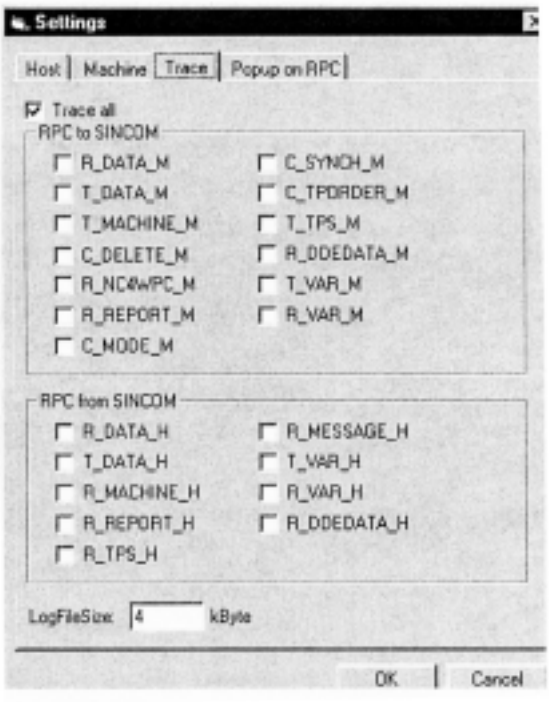
マシン 1



マシン 2

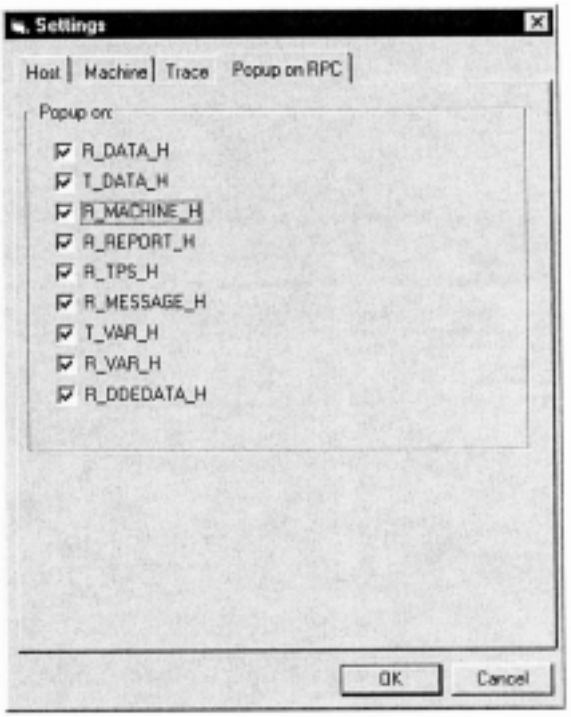


[Trace] タブを表示させると，トレースする RPC を選択することができます。トレースは画面に表示され，ScoTest.log ログファイルにログインされます。そのログファイルの最大サイズを決めることもできます。



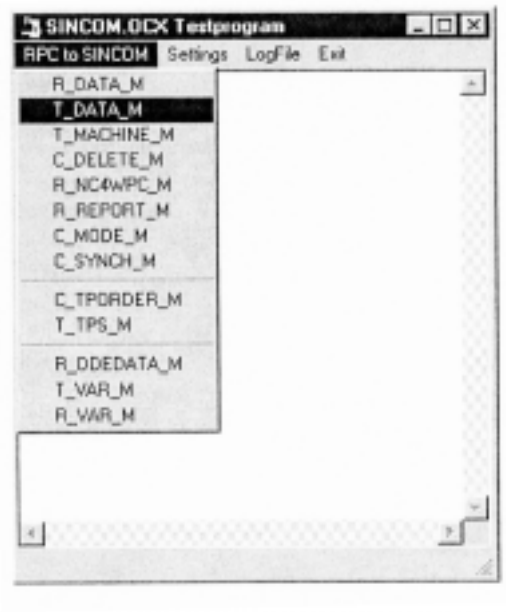
[Popup on RPC] タブを表示させると，受取った RPC に対する応答を定義することができます。

10.4.1 構成

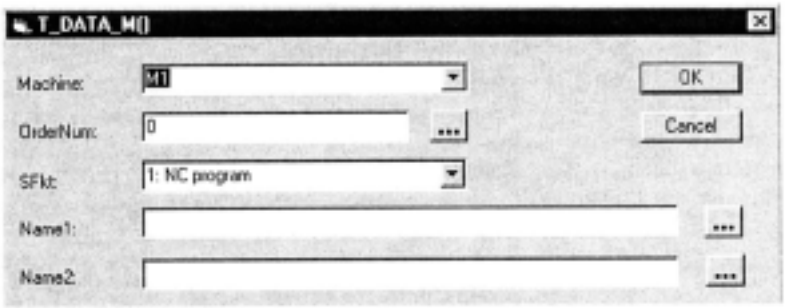

Settings
Host
Machine
Trace
Popup on RPC
Popup on:
R_DATA_H
T_DATA_H
R_MACHINE_H
R_REPORT_H
R_TPS_H
R_MESSAGE_H
T_VAR_H
R_VAR_H
R_DDEDATA_H
OK
Cancel

10.4.2 RPC を SINCOM へ送信

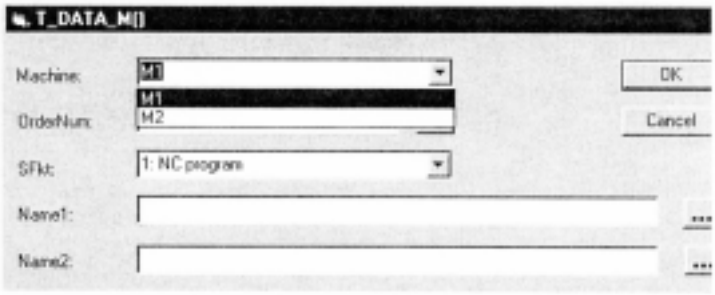
ScoTest アプリケーションの対話型形式によって，定義されたすべての RPC を SINCOM へ送ることができます。［RPC to SINCOM］メニューから対話型形式を呼出すことができます。



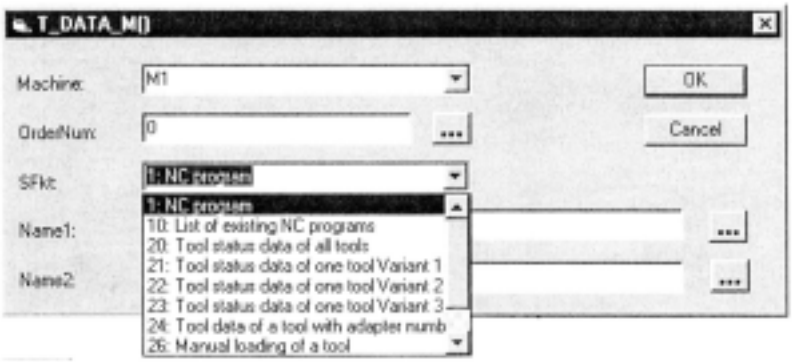
次に RPC 送信の例として T_DATA_M を示します。



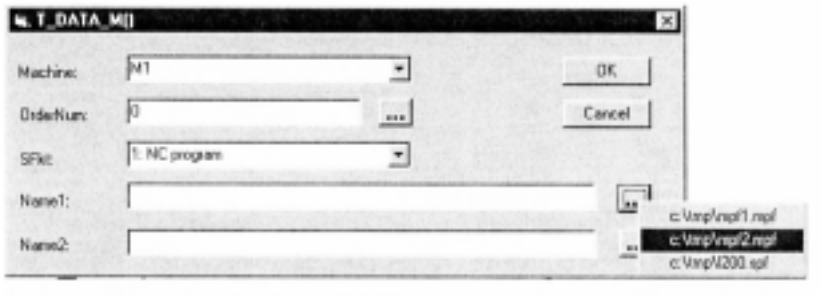
入力フィールドには，本資料に示された各 RPC パラメータに対応する名前が記されています。これらのパラメータは，本マニュアルで説明されています。マシンパラメータでは，どの制御装置へ RPC を送信するか定義することができます。この入力フィールドは選択ができるようになっており，構成で定義された通信相手を示します。



本資料に既に定義された値を持つパラメータ（例えば SFct) がすべてラジオボタン形式で示されています。



その他の入力フィールドでは，「 ... 」ボタンをクリックして以前に使用した値を選択することができます。



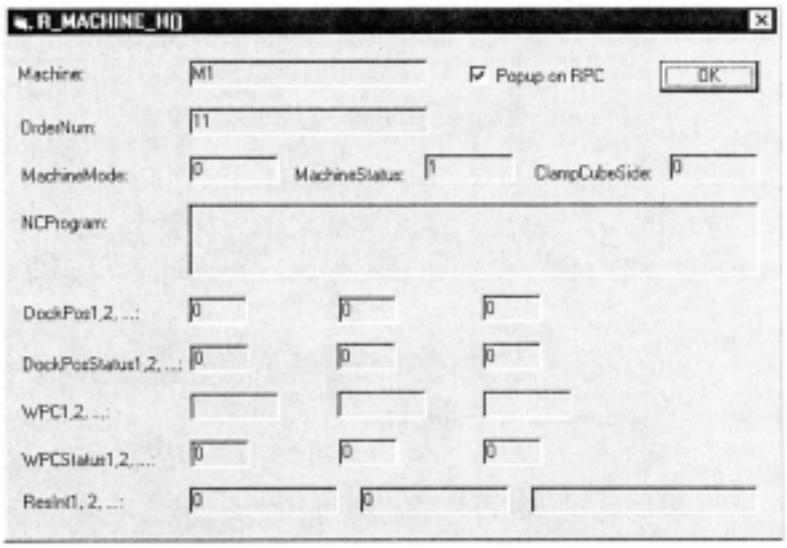
［OK］ボタンをクリックすると RPC を送信できます。送信された全ての RPC がトレースされます。そのトレースは，ScoTest アプリケーションのメインウィンドウの画面に出力され，ScoTest.log ログファイルの ScoTest.log にログインされます。送信された各 RPC のために，戻り値もトレースされ，ステータスバーに表示されます (Ret=0)。戻り値の詳細については，10.3.6「エラー処理」を参照してください。

SINCOM との基本リンクをテストするには，T_MACHINE_M の RPC を送ります。SINCOM は，R_MACHINE_H の RPC で応答します。

10.4.3 SINCOM からの RPC を受信

オプションの**ホストイネーブル**をオンにして通信相手を構成すると，ScoTest アプリケーションは，SINCOM からの RPC を受取る用意ができます。ホストイネーブルをオフにして構成されていると，少なくとも 1 つでも RPC が SINCOM へ正しく転送されたことがある場合のみ，SINCOM から RPC が送られます。

受信した RPC はトレースされます。そのトレースは，ScoTest アプリケーションのメインウィンドウの画面に出力され，ログファイル ScoTest.log にログインされます。各 RPC の全パラメータを表示した対話型様式も現れます。



出力フィールドには，本資料に示された各 RPC に対応するパラメータ名が入ります。これらのパラメータの詳細については，本マニュアルで説明しています。

受信した RPC の対話型形式の画面を表示するには [Popup on RPC] ラジオボタンにチェックを入れ，非表示にするにはチェックを外します。または，設定の［Popup on RPC］タブでも行えます。

10.4.4 ScoTest アプリケーションのソースコード

このアプリケーションのソースコードは，セットアップルーチンによって Sincom-

ocx\ScoTest ディレクトリに格納されています。

このアプリケーションは，Microsoft VisualBasic 6.0 開発システムで書かれています。

このアプリケーションは次のモジュールで構成されています：

ScoTest.vbp	VisualBasic プロジェクトファイル（project file）
ScoTest.frm ScoTest.frx	アプリケーションのメインウィンドウ
ScoConfig.frm ScoConfig.frx	環境設定（Configuration form）
History.frm	入力値の選択
Logen.bas	トレース機能（Tracing functions）
Util.bas	補助機能
R_DATA_H.frm ...	各 RPC に対応する入力および画面形式

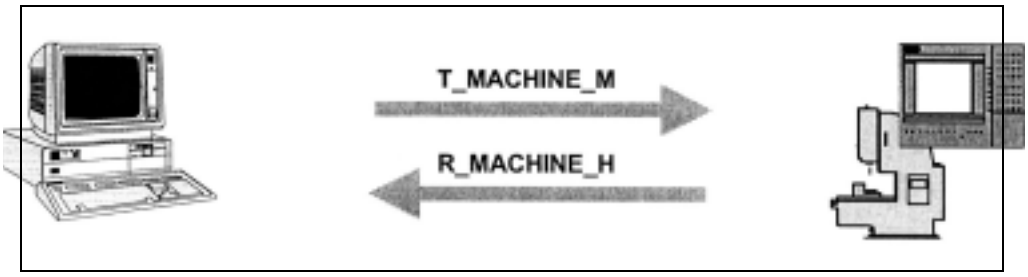
10.5 SINCOM-OCX の使用例

本章で示される例については，10.4.1「構成」で表示されているネットワーク構成を参照してください。お客様のネットワーク構成を例と対応させるには，それぞれの例に応じたソースコードの IP アドレスに変更しなければなりません。
SINCOM がマシン制御装置にインストールされ，ネットワークリンクが確立されていることを前提とします。

10.5.1 例 1 - マシン状況の問合せ (Visual Basic)

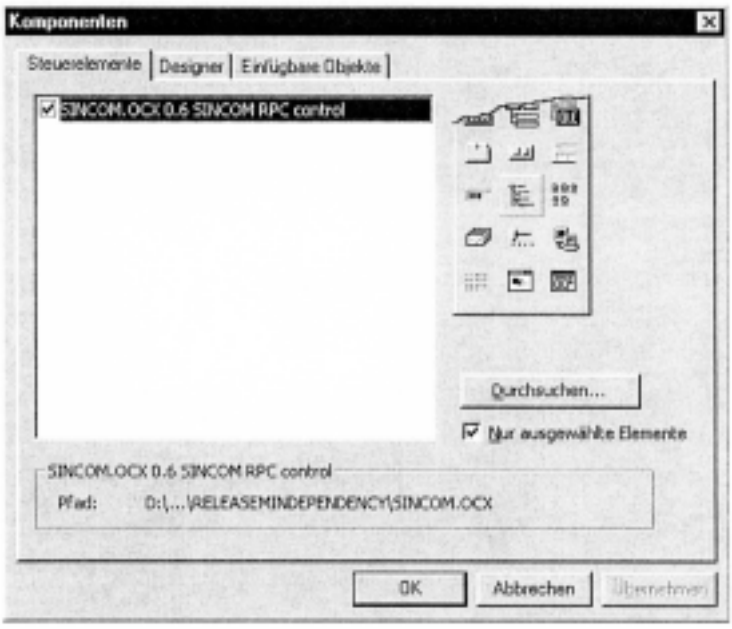
この例では，Visual Basic アプリケーションは T_MACHINE_M() の RPC をマシンへ送り，その応答として SINCOM から R_MACHINE_H() の RPC を受取ります。

RPC の使用法については，本資料の 5.3.1「マシンステータスデータの送信」および 5.3.2「マシンステータスデータのリクエスト」を参照してください。アプリケーションを作成するための Visual Basic 開発環境に必要なステップがすべて示されています。



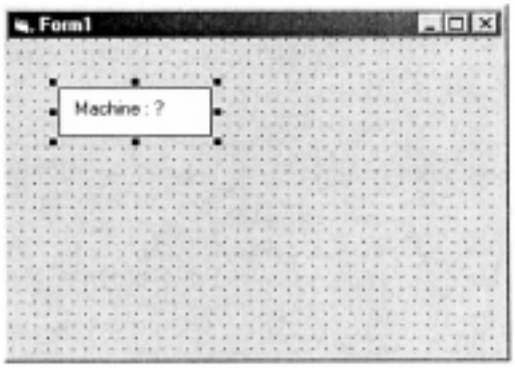
SINCOM-OCX コンポーネントを Visual Basic 6.0 に組込む

SINCOM-OCX コンポーネントを Visual Basic で使用するためには，次の画面でのコンポーネントの設定が必要です。この画面は，メニューの [Project] -> [Components] で表示されます。



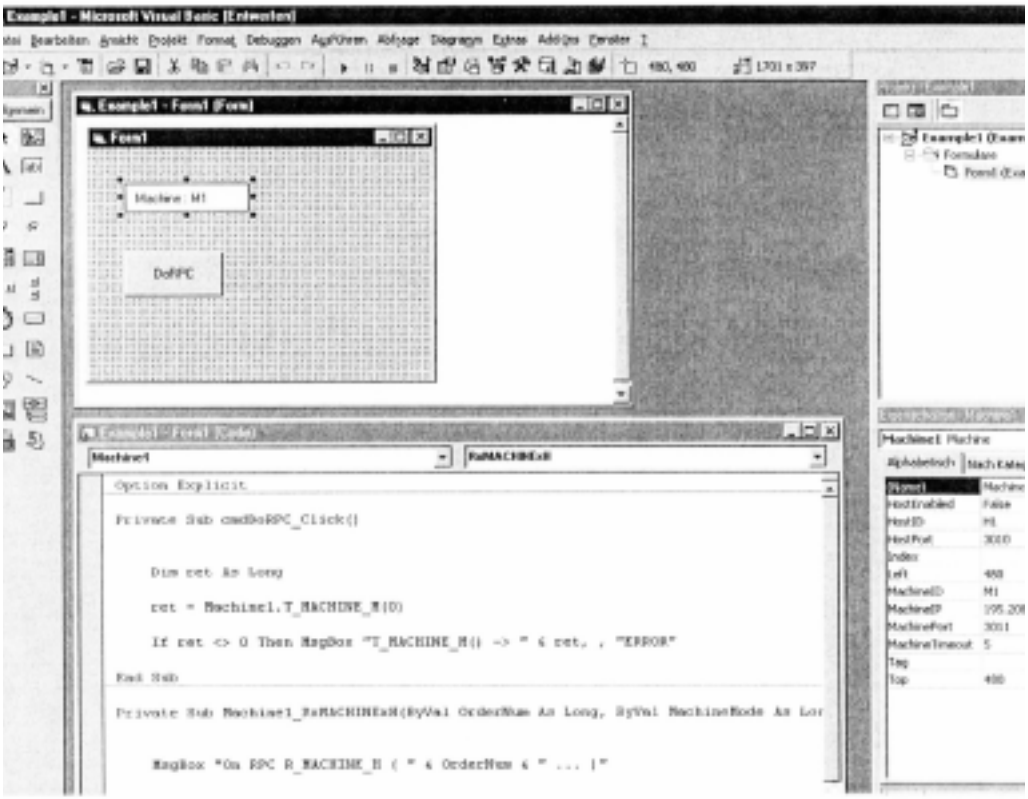
その後に SINCOM-OCX アイコン（黄色のマシン）が，ツールボックスウィンドウに現れます。

SINCOM-OCX アイコンをフォームにドラッグアンドドロップします。そうするとコンポーネントが開発環境に表示されます。しかし，完全なアプリケーションのランタイム中には表示されません。



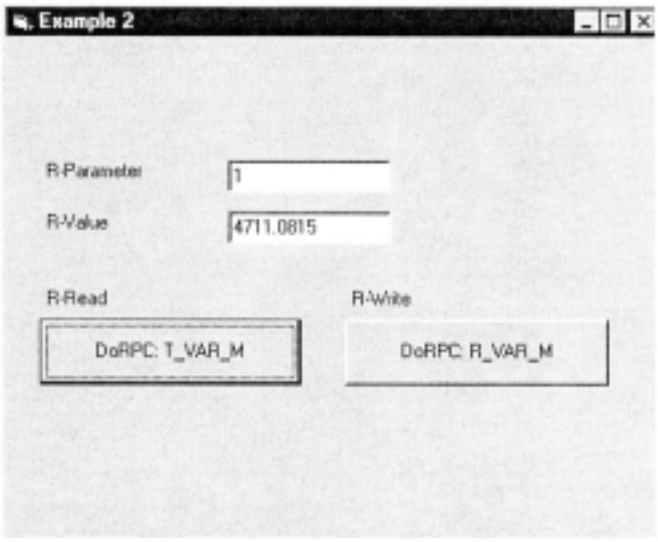
プロパティウィンドウでは，属性を格納することができます。マシン制御装置の IP アドレスには，お客様のネットワーク構成に応じた IP アドレスを使用してください。

マシン ID の属性を変更すると，フォーム上の OCX の表示も変更されます。[DoRPC] ボタンをクリックすると，T_MACHINE_M の RPC の送信がトリガされます。SINCOM アプリケーションは，R_MACHINE_H の RPC で応答します。例のアプリケーションに対する応答が，メッセージボックスに表示されます。

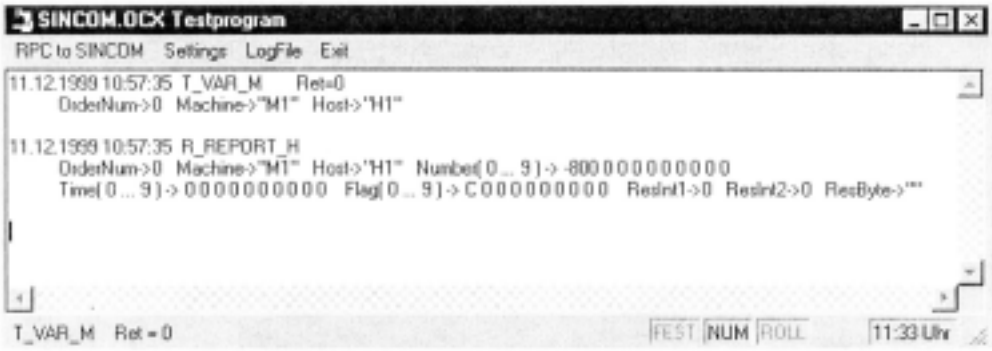


10.5.2 例 2 - R パラメータの読込みおよび書込み（Visual Basic）

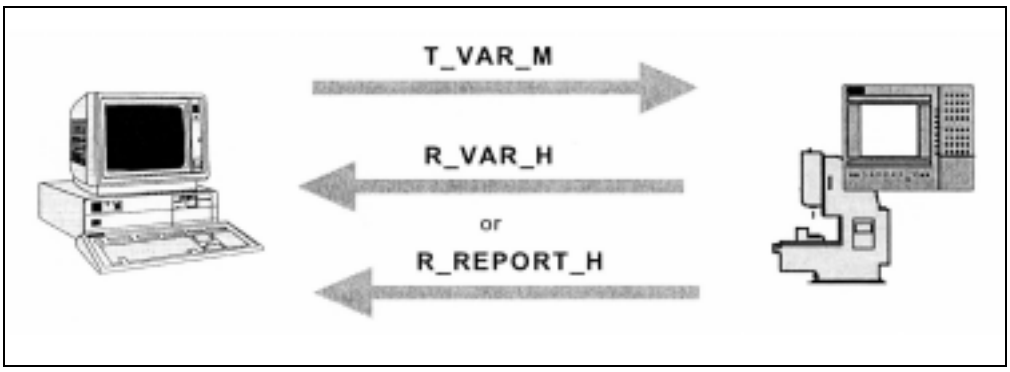
この例では，R パラメータの例を使用して SINCOM 変数サービスを説明します。変数サービスの詳細については，7 章「設定可能なデータ転送 / 変数サービス」を参照してください。



SINCOM の変数サービスをオンにするには，変数セットが 1 つでも制御装置の c:\add_on\scvarset.ini ファイルに定義されている必要があります。このファイルに変更が加えられた場合，制御装置のコールドリスタート後にその変更は有効になります。そうでない場合は，SINCOM は R_REPRT_H() の RPC で応答しエラー -800 を返します。

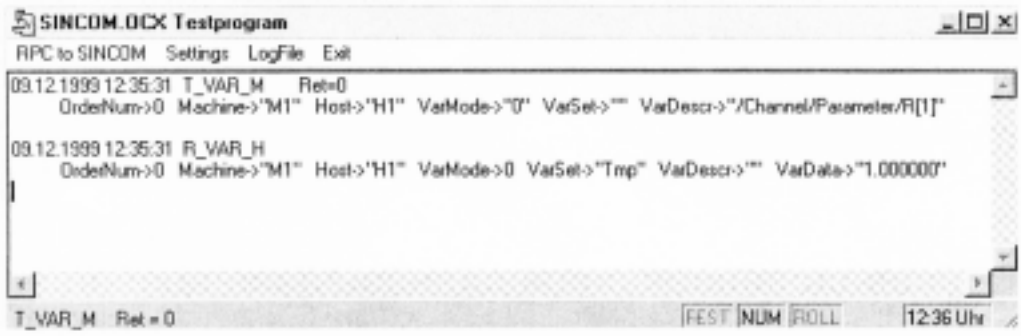


R パラメータ読込みのフローチャート

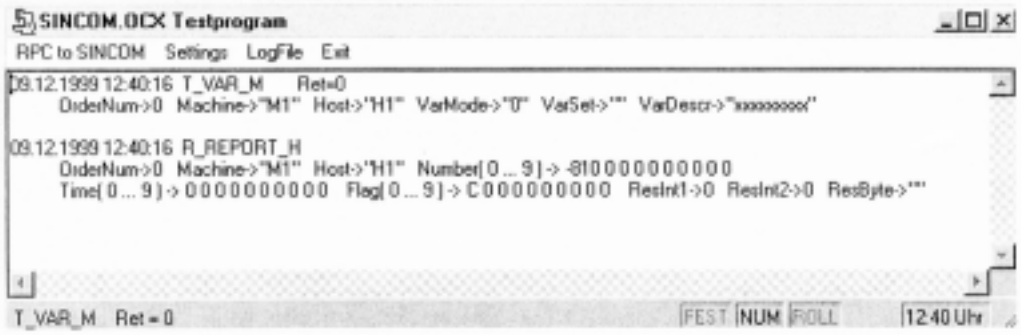


VisualBasic アプリケーションは，T_VAR_M() の RPC を使用して R パラメータ値を要求します。

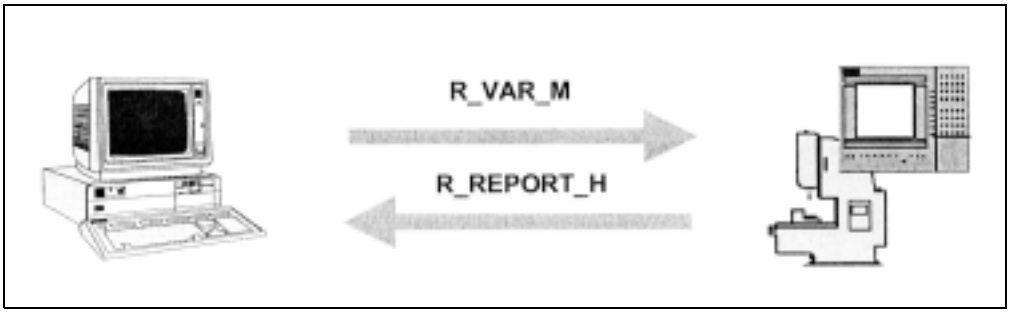
SINCOM は，R_VAR_H() の RPC を使用して現在の R パラメータ値を返します。



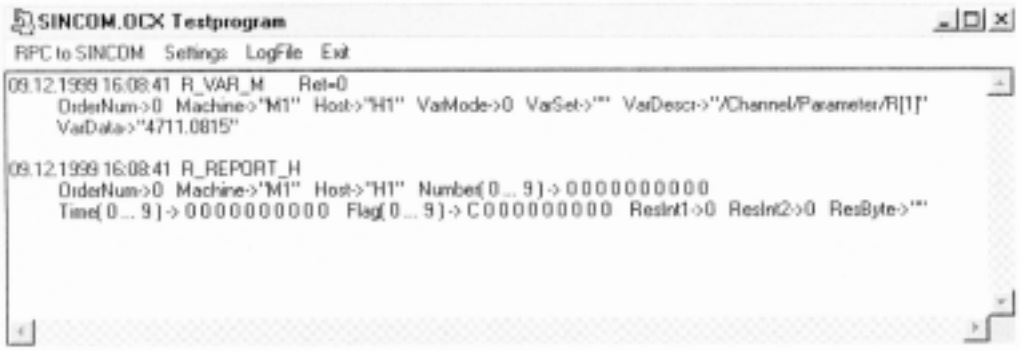
エラーの場合には，SINCOM は R_REPORT_M() の RPC を返します。



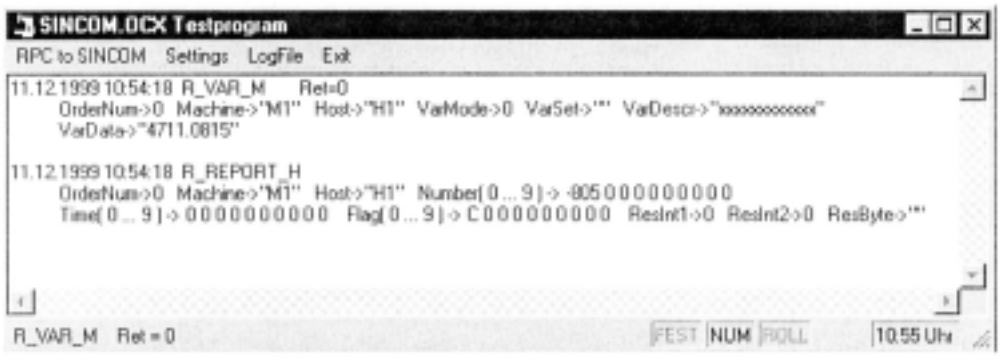
R パラメータ書込みのフローチャート



VisualBasic アプリケーションは，R_VAR_M() の RPC を使用して SINCOM へ R パラメータ値を渡します。SINCOM は，R_REPORT_H() の RPC を使用して書込み操作を通知します。



エラーの場合には，SINCOM から R_REPORT_H() の RPC が送られます。ただし，Number(0) パラメータには，関連したエラーコードが含まれます。

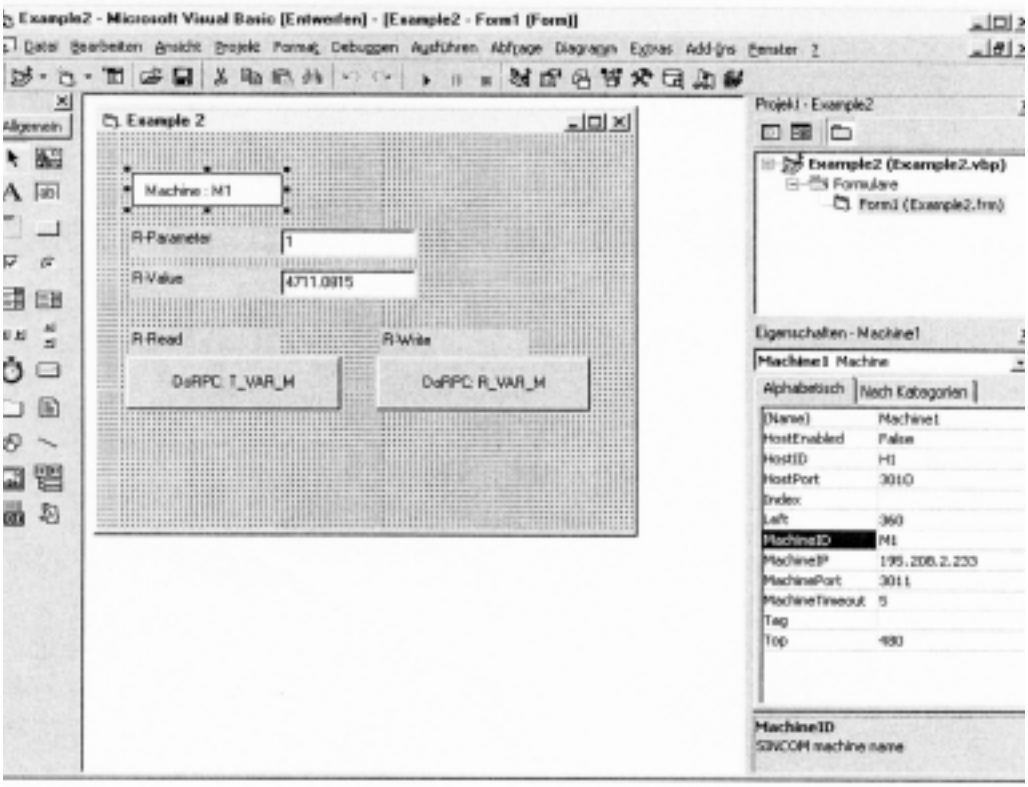


Visual Basic ソース コード

SINCOM-OCX の組込みは完了しおいてください。詳細については，10.5.1「例 1 - マシン状況の問合せ (Visual Basic)」の「SINCOM-OCX コンポーネントを Visual Basic 6.0 に組込む」を参照してください。

R パラメータの入力フィールドには，読込みおよび書込みの R パラメータ数を定義することができます。**R 値**の入力フィールドには，現在読込まれている R パラメータ値，あるいは書込まれて新たに入力された値が表示されます。

どちらかのボタンをクリックして，T_VAR_M() あるいは R_VAR_M() の RPC を送ります。



Date: \Examples\Example2\Example2.frm

```
Option Explicit
Private Sub cmdT_VAR_M_Click()
    ' read R parameter
    Dim ret         As Long
    Dim RParam      As Long         ' R parameter number
    Dim RItem       As String       ' item for access
    RParam = Val(txtRParam.Text)
    RItem = "/Channel/Parameter/R[" & RParam & "]"
    ret = Machine1.T_VAR_M(0, 0, "", RItem)
    If ret <> 0 Then MsgBox "T_VAR_M() -> " & ret, , "ERROR"
End Sub
Private Sub cmdR_VAR_M_Click()
' write R parameter
    Dim ret         As Long
    Dim RParam      As Long         ' R parameter number
    Dim RItem       As String       ' item for access
    Dim RValue      As String
    RParam = Val(txtRParam.Text)
    RItem = "/Channel/Parameter/R[" & RParam & "]"
    RValue = txtRValue.Text
    ret = Machine1.R_VAR_M(0, 0, "", RItem, RValue)
    If ret <> 0 Then MsgBox "R_VAR_M() -> " & ret, , "ERROR"
End Sub
Private Sub Machine1_RxVARxH(ByVal OrderNum As Long,_
    ByVal VarMode As Long, ByVal VarSet As String,_
    ByVal VarDescr As String, ByVal VarData As String)
  ' show R parameter in the form
  txtRValue.Text = VarData
End Sub
Private Sub Machine1_RxREPORTxH(ByVal OrderNum As Long,_
    ByVal Typ As Long, ByVal Number As Variant,_
    ByVal Time As Variant, ByVal Flag As Variant,_
    ByVal ResInt1 As Long, ByVal ResInt2 As Long,_
    ByVal ResByte As String)
  If Number(0) <> 0 Then
    MsgBox "On RPC R_REPEOR_H ( ... Number(0)->" & Number(0)&")"
  End If
End Sub.
```

10.5.3 例 3 - R パラメータ読込みの起動（Internet Explorer）

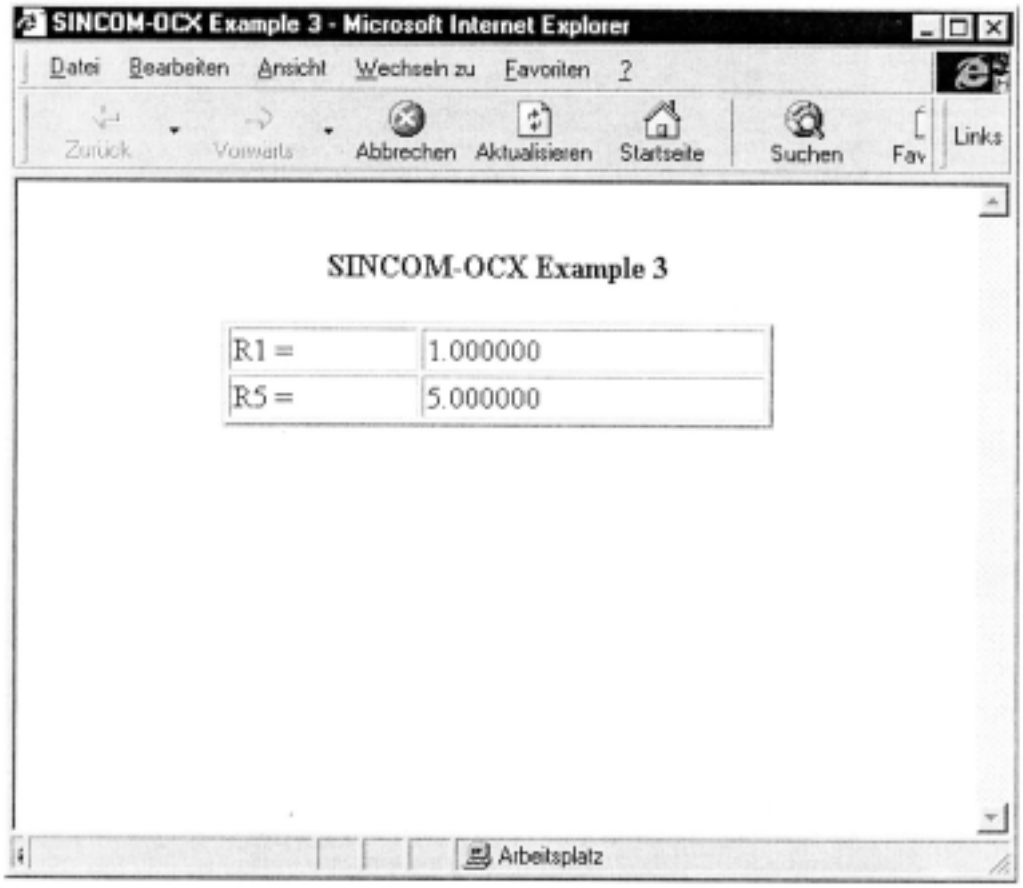
この例では，R パラメータ読込みの起動を MS Internet Explorer を使用して説明します。読込みの起動機能（別称ではホットリンク (hotlink)）をイネーブルすると，変数セットでのデータの変更は変更ごとに SINCOM が SINCOM-OCX へすぐに通知します。この例では，R パラメータの R1 および R5 に含まれる「 Set01 」という名称の付いた変数セットを使用します。

変数セットは制御装置の c:\add_on\scvarset.ini ファイルに定義されています。

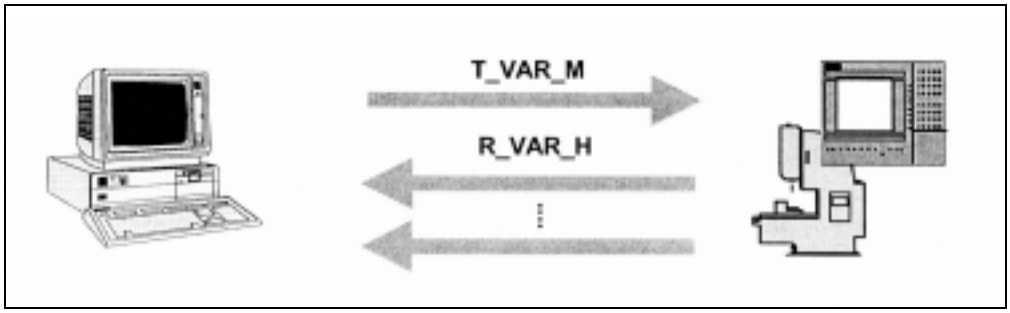
```
[Set01]
Mode=0
Host=FLR1
Var01=/Channel/Parameter/R[1]
Var02=/Channel/Parameter/R[5]
```

MS Internet Explorer を使用するには，SINCOM-OCX が既にインストールされている必要があります。

MS Internet Explorer で表示されている \Examples\Example3\Example3.html ファイル。

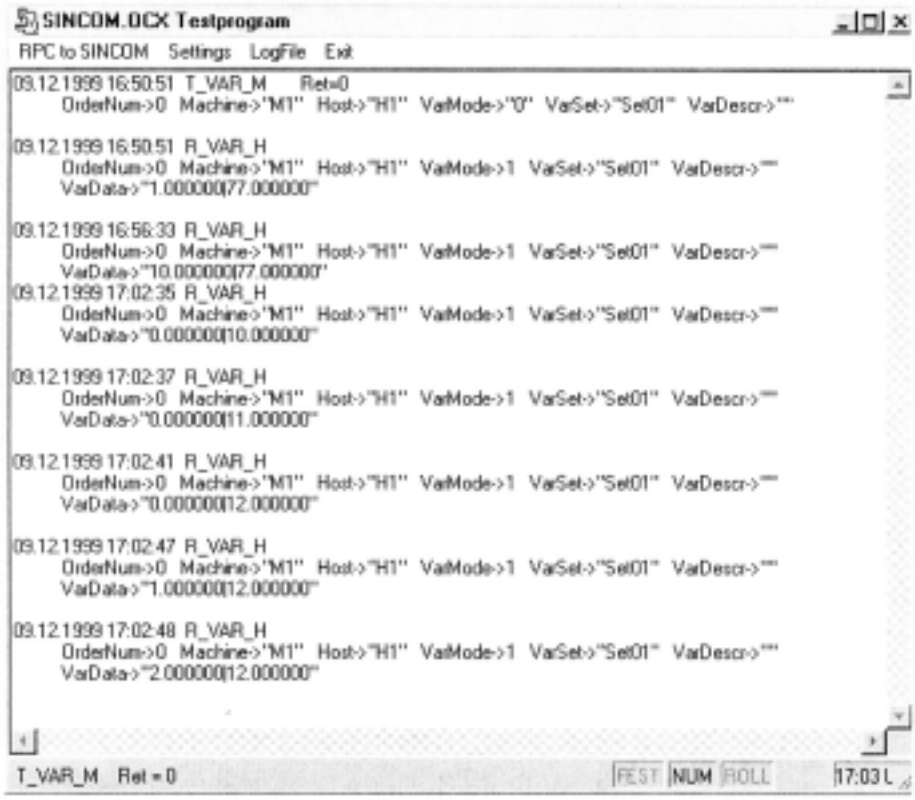


R パラメータ読込みの起動フローチャート



HTML ページがロードされている時に，VBScript 機能を使用して T_VAR_M() の RPC を SINCOM へ送ります。この RPC を使用して，「 Set01 」セットの現在の変数値を要求します。R_VAR_H() の RPC を使用して，このデータは SINCOM からすぐさま送られます。

RPC: R_VAR_H() の RPC を使用して，SINCOM はセット（ R1 あるいは R5 ）での変数の変更を知らせます。



HTML ページのソースコード

SINCOM-OCX は，HTML コードの <OBJECT> タグでリンクされています。
<OBJECT> タグ内には，SINCOM-OCX の属性が記録されています。

\Examples\Example3\Example3.html ファイル

```
<HTML>
  <HEAD>
        <TITLE>SINCOM-OCX Example 3</TITLE>
  </HEAD>

  <BODY>

<OBJECT classid=CLSID:EDF199C1-4F2E-11D3-9DC3-00A0249B4877
                                                    id=Machine1>

 <PARAM NAME="MachineID"                    VALUE="M1">
 <PARAM NAME="MachineIP"                    VALUE="195.208.2.233">
 <PARAM NAME="MachinePort"                  VALUE="3011">
 <PARAM NAME="MachineTimeout"               VALUE="5">
 <PARAM NAME="HostID"                       VALUE="H1">
 <PARAM NAME="HostPort"                     VALUE="3010">

</OBJECT>

   <P align=center><STRONG>SINCOM-OCX Example 3</STRONG
                                                    </P>
   <TABLE border=2 align=center width=60% id=TABLE1>

    <TR>
          <TD> R1 = </TD> <TD><LABEL id=R1Param></LABEL> </TD>
    </TR>

    <TR>
          <TD> R5 = </TD> <TD><LABEL id=R5Param></LABEL> </TD>
    </TR>

    </TABLE>

   </BODY>
```

HTML ページは，次の 3 つの VBScript 機能を含んでいます。

Window_OnLoad	HTML ページがロードされた時に呼出されます。
Machine1_RxVARxH	RPC R_VAR_H を受取る時に呼出されます。
Machine1_RxREPORTxH	R_REPORT_H の RPC を受取る時に呼出されます。

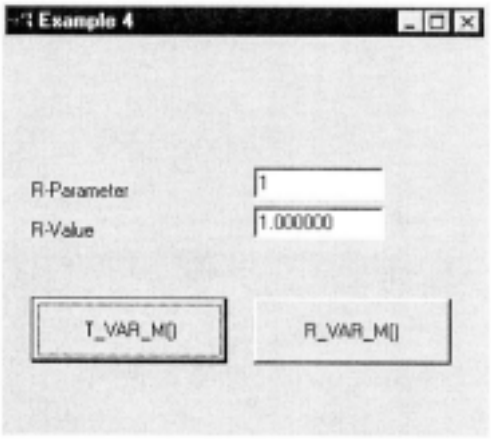
\Examples\Example3\Example3.html ファイルの続き

```
<SCRIPT LANGUAGE="VBScript">

    Option Explicit

    Sub Window_OnLoad

        dim ret

        ret = Machine1.T_VAR_M(0, 0, "Set01", "")
        if ret <> 0 then MsgBox "T_VAR_M()->" & ret

    End Sub

    Sub Machine1_RxVARxH( OrderNum, VarMode, VarSet, VarDescr,
                                                                    VarData )

        dim pos

        pos = InStr( VarData, "|" )

        if pos = 0 then
                R1Param.innerText = VarData
          else
                R1Param.innerText = Left(VarData, pos-1)
                R5Param.innerText = Mid (VarData, pos+1)
          end if

    End Sub

    Sub Machine1_RxREPORTxH( OrderNum, Typ, Number, Time,

                                                Flag, ResInt1, ResInt2, ResByte )

        If Number(0) <> 0 Then
          MsgBox "On RPC R_REPEOR_H ( ... Number(0)->" &
                                                                Number(0)&")"

          End If

       End Sub

    </SCRIPT>

</HTML>
```

10.5.4 例 4 - R パラメータの読込みおよび書込み（Visual J++）

この例では，例 2 とは異なり MS Visual J++ 6.0 SP3 を使用します。

フローチャートについては，例 2 と同じです。

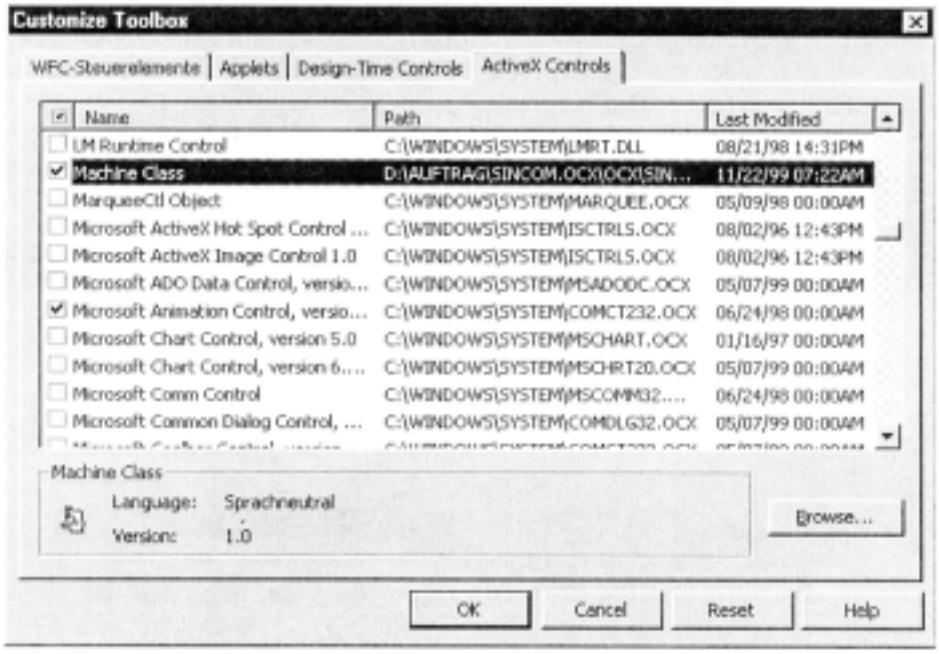


SINCOM-OCX を MS Visual J++ に組込む

MS Visual J++ 開発環境によって，ActiveX コンポーネントを使用することができます。

組込みを行うには，メニューの [Tools] ->[Customize ToolBox] ->[ActiveX Controls] を開きます。

このフォームでは，[Machine Class] チェックボックスにチェックを付けてください。

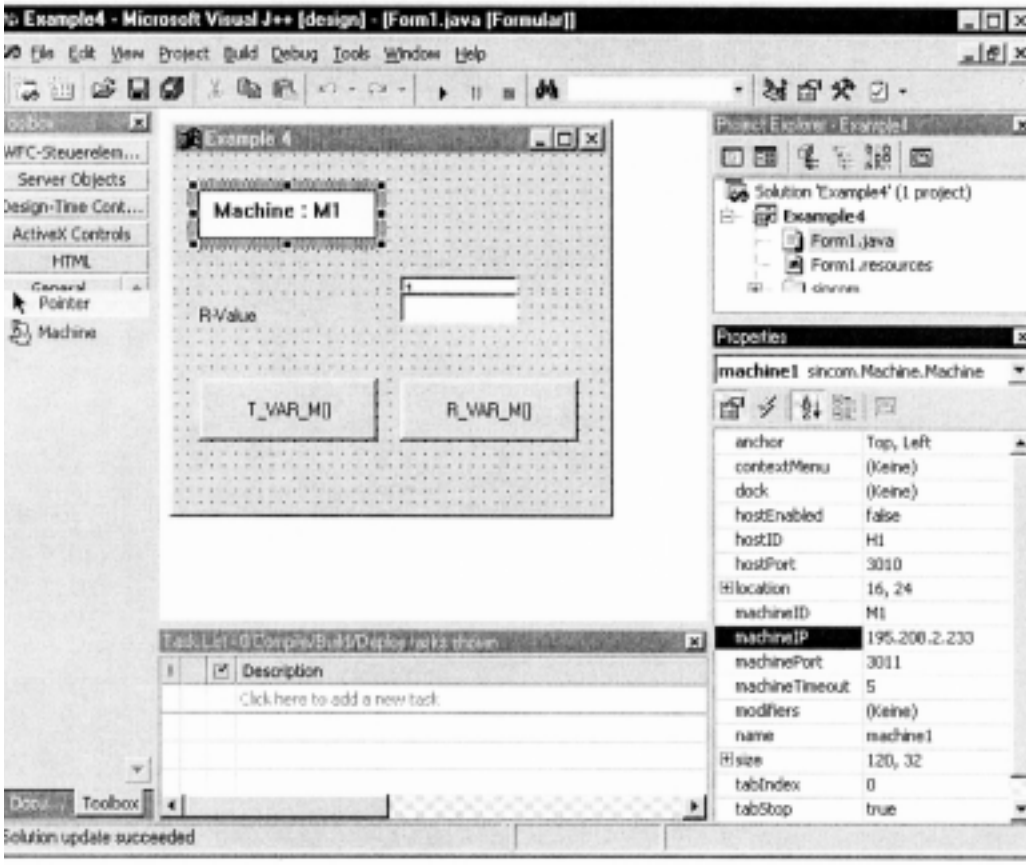


SINCOM-OCX の組込みには，Visual J++ で sincom ディレクトリにクラスを追加します。

Visual-J++ アプリケーションのソースコード

R パラメータの入力フィールドには，読込みおよび書込みの **R パラメータ数**を定義することができます。**R 値**の入力フィールドには，現在読込まれている R パラメータ値，あるいは書込まれて新たに入力された値が表示されます。

どちらかのボタンをクリックして，T_VAR_M() あるいは R_VAR_M() の RPC を送ります。



Examples\Example4\Form1.java ファイル

```
private void cmdT_VAR_M_click(Object source, Event e)
{
    // read R parameter

    long ret;
    String[]     VarDescr     = new String[1];          // item for access
    String[]     VarSet       = new String[1];

    VarSet[0]      =  "";
    VarDescr[0]   =  "/Channel/Parameter/R[" + txtRParam.getText() + "]";

    ret = machine1.T_VAR_M(0,0,VarSet,VarDescr );
    if ( ret != 0 ) MessageBox.show("T_VAR_M() -> " + ret);
}

private void cmdR_VAR_M_click(Object source, Event e)
{
    // write R parameter

    long ret;
    String[]    VarDescr     =  new String[1];          // item for access
    String[]    VarSet       =  new String[1];
    String[]    VarData      =  new String[1];

    VarSet[0]      =  "";
    VarDescr[0]   =  "/Channel/Parameter/R[" + txtRParam.getText() + "]";
    VarData [0]   =  txtRValue.getText();

    ret = machine1.R_VAR_M(0, 0, VarSet, VarDescr, VarData);
    if ( ret != 0 ) MessageBox.show("R_VAR_M() -> " + ret);
}

private void machine1_RxVARxH(Object source,sincom.Machine.RxVARxHEvent e)
{
        // show R parameter in the form

        txtRValue.setText( e.VarData );
}

private void machine1_RxREPORTxH(Object source,
                                                    sincom.Machine.RxREPORTxHEvent e)
{
    int     ErrorNr = e.Number.getVariantArray()[0].getInt();
    if  (  ErrorNr != 0 )
    {
     MessageBox.show( "On RPC R_REPEOR_H ( ... Number(0)->"
                                                                        + ErrorNr +")");
    }
}
```

パート 2 PLC/NCK インタフェース

1 RKS と マシン PLC のインタフェース

1.1 説明

コンピュータ通信ソフトウェア（PKS）とマシン PLC のインタフェースを説明します。

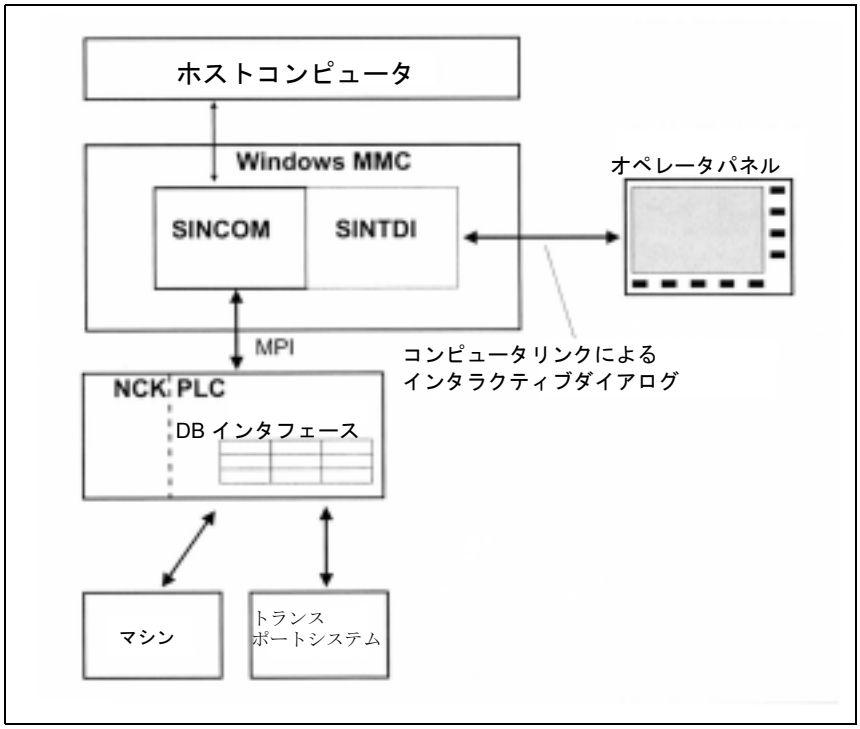


図 1.1 概要

コンピュータリンク用ソフトウェアと PLC の通信には、インタフェース DB が必要です。そのために Siemens Standard のデータブロック 12 が予約されています。このデータブロックはユーザーーが作成してください。DB インタフェースのデータエレメントはブロック内に構築され、それぞれが何らかの形でインタフェースに関連します（例えば、グローバルデータ、ドッキング位置データ、NC プログラム割当て、など）。これらのブロックは表のフォーマットで表示されます。ブロックは全て、インタフェース DB 内に次々と保存されます。

タイプ "int(WORD)" および "Long(DWORD)" のバイナリデータ要素は、DB 内に S7 フォーマット（リトル・エンディアン）で保存されます。MMC がこれらのデータ要素にアクセスするときには、データをインテルフォーマット（ビッグ・エンディアン）に変換します。名前を表すデータ要素は、ASCII 文字を使用するバイトフィールドとして構築されます。

このインタフェースは表のフォーマットで記述されます。"Access " 「アクセス」コラムは、そのフィールドを記述した人を示します。このコラムでは、次のような略語が使用されます：

- RKS　MMC プラットフォームでのコンピュータリンクソフトウェア（ホストコンピュータとは直結していない）
- Operator　コンピュータリンクのインタラクティブプログラム
- PLC　PLC ユーザープログラム

内部リソースの通信ロードを最小にするため、RKS インタフェースに変更がある毎に「PLC からのリクエスト」を介して中継します（インタフェースの一部）。RKS はこのリクエストを瞬時に検出します（ホットリンク）。

Machine　マシン

グローバルデータ	
最初のドッキング位置	ドッキング位置数はグローバルデータに定義される
2 番目のドッキング位置	
...	
n 番目のドッキング位置	
最初のプログラム割当て	プログラム割当てはオプションで、割当て数はグローバルデータに定義される。
2 番目のプログラム割当て	
...	
n 番目のプログラム割当て	

トランスポートシステム

グローバルデータ	
最初のドッキング位置	ドッキング位置数は、グローバルデータに定義される。
2 番目のドッキング位置	
...	
n 番目のドッキング位置	
最初のトランスポートジョブ	トランスポートジョブ数は、グローバルデータに定義される。
2 番目のトランスポートジョブ	
...	
n 番目のトランスポートジョブ	

1.2 グローバルデータ

表 1.1 グローバルデータリスト

データ要素	略名称	データタイプ	アクセス元	オフセット
PLC からのリクエスト	PLCReq	Byte	PLC/RK S	0
変更トリガ	Trigger	Byte	PLC	1
RKS からのリクエスト	SCReq	Byte	RKS/PL C	2
マシンモード	MODE_PLC	Byte	PLC	3
RKS モード	MODE_RKS	Byte	RKS/ オペレータ	4
ツールデータ用フラグ ,	DataTyp	Byte	PLC	5
マガジン番号	MagNum	Word	PLC	6，7
位置番号	PlaceNum	Word	PLC	8，9
T 番号	Tnum	Word	PLC	10，11
ドッキング位置数	DockPosCount	Byte	PLC	12
プログラム割当て数	NC4WpcCount	Byte	PLC	13
マシンステータス	MachineStatus	Byte	PLC	14
NC モード	MachineMode	Byte	PLC	15
予約 1	Reserve1	Word	PLC	16，17
予約 2	Reserve 2	Word	PLC	18，19

（注）

データブロックの個々の入力は、以下のテーブルシーケンスで説明します。

PLC からのリクエスト :

表 1.2 PLC からのリクエスト

ビット番号	機能	アクセス元
0	変更済みワークキャリアステータス	PLC - 1/RKS - 0
1	リポートツール	PLC - 1/RKS - 0
2	ステータス変更	PLC - 1/RKS - 0

PLC は、このバイトを使用してインタフェースの変更を指示します。PLC はリクエストバイトをイネーブルにすると必ずトリガバイトに次のビットをイネーブルにすることになります（以下を参照してください）。

処理が終了した後に RSK によって 0 に設定されて初めて、PLC はリクエストバイトを再び書き込めるようになります。

変更済みワークキャリアステータス

ワークキャリアのいづれかのステータスが PLC によって変更される（1.3「マシンのドッキング位置データ」参照）か、あるいはマシン内でパレット動作が生じた場合には必ず、PLC によって「ワークキャリステータスの変更」が設定されます。

ツールをリポートする

ホストコンピュータにツールをリポートすべき場合には必ず、PLC によって「ツールをリポートする」が設定されます（例、ツール • ブレイク）。データ要素は、リポートされるべきツールを特定します。

「マガジン番号」および「ロケーション番号または T 番号」

ステータス変更

PLC は、ホストコンピュータに報告する必要があるステータス変更（マシンモード、RKS モード、マシンステータス、NC モード）の全てに " ステータス変更 " をセットします。それによって、RKS はホストコンピュータに R_MACHINE_H () を送ります（例、ワークキャリア、NC Start、NC End、モード変更、など）。

トリガを変更する

コンピュータリンクは、このバイトに DDE ホットリンクをセットアップします。

PLC は、PLC で複数の変更が生じた際にこのバイトに 1 ビットをセットします。新規トリガのそれぞれに対して、PLC は次のビットをセットし、最後のビットをリセットしなければなりません ; ビット 7 の後は、再びビット 0 から始まります。

RKS からリクエストする :

表 1.3 コンピュータ通信ソフトウェアへのリクエスト

ビット番号	機能	アクセス元
0	同期フラグ	RKS
1	ユニットを停止する	RKS
2	ユニットを起動する	RKS
3	ドッキング位置データへの書きこみアクセスをリクエストする	RKS

個々のビットは、RKS によってセットおよびリセットされます。

同期フラグ

「同期フラグ」は、ホストコンピュータによってセットおよびリセットされます。(-> /1/ C_SYNCH_M () パート 1 5.14「同期」参照) 同期プロセスの期間中、マシンのステータスは同一でなければなりません。PLC は、新規のマシニングオペレーションを開始することも、パレット動作を実行することもできません。

ユニットを停止する

「ユニットを停止する」は、ホストコンピュータによってセットおよびリセットされます。(-> /1/ C_MODE_M () パート 1 5.13「MODE 選択」参照)「ユニットを停止する」は、PLC がユニット（ドライブ）を停止するようリクエストします（マシンモード、ユニットの停止も参照してください）。

ユニットを起動する

「ユニットを起動する」は、ホストコンピュータによってセットおよびリセットされます。(-> /1/ C_MODE_M () パート 1 5.13「MODE 選択」参照)「ユニットを起動する」は、PLC はユニット（ドライブ）を起動するようリクエストします（マシンモード、ユニットの起動も参照してください）。

ドッキング位置データへの書き込みアクセスをリクエストする

ドッキング位置データ（ワークキャリアステータス、追従加工、加工サイド）を変更しようとする際には必ず、RKS によって「ドッキング位置データへの書き込みアクセスをリクエストする」が設定されます。PLC がすでに「マシンモード」データ要素に「ドッキング位置データへの書き込みアクセスの許可」フラグを設定している場合、変更できるのは RKS のみとなります。RKS は、変更を実行してから後にリクエストをリセットしなければならず、ついで PLC も「ドッキング位置データへの書き込みアクセスの許可」フラグをリセットしなければなりません。この協同調整により、RKS がパレット動作の結果、誤ったドッキング位置を書き込まないよう防止できます。

マシンモード

表 1.4 マシンモード

ビット番号	機能	アクセス元
1	ユニットの停止 (作業日の終わり)	PLC
2	ドッキング位置データへの書き込みアクセスの許可	PLC

構成機器の停止

このステータスが検出されると、PLC によって「ユニットの停止」が設定されます。このリクエストは、リクエストフラグのビット 1 に設定されます。

ドッキング位置データへの書き込みアクセスの許可

リクエスト「ドッキング位置データへの書き込みアクセスをリクエストする」に応じて、「ドッキング位置データへの書き込みアクセスの許可」が PLC によって設定されます。この調整により、RKS がパレット動作の結果、誤ったドッキング位置を書き込まないよう防止できます。

RKS モード :

表 1.5 コンピュータ通信ソフトウェアモード

ビット番号	オペレーティングモード	アクセス元
0	ホストコンピュータモード無人	オペレータ
1	ホストコンピュータモード無人	オペレータ
2	手動モード	オペレータ
3	特殊モード	RKS, オペレータ
4	オフライン	RKS

オペレータは、RKS ダイアログで RKS モードをインタラクティブに設定することができます。特殊モードも、C_MODE_M () を使用してホストコンピュータによって選択および選択解除することができます。RKS が接続に割り込みがあることを検出した場合、RKS は「オフライン」用のビットをセットします。オフラインビットがアクティブになると、RSK からホストコンピュータへのデータ送信は行われなくなります。

ホストコンピュータモード　有人／無人

ホストコンピュータモード「有人」と「無人」では、NC Start は PLC によって始動されます（ホストコンピュータの主導による）;「ホストコンピュータモード　無人」と「ホストコンピュータモード　有人」の違いは、無人および有人生産時に発生したフォールトに対応するさまざまな対策のいずれを使用する可能性があるかということと関係しています。

手動モード

「手動モード」では、NC Start は自動ではなくなりますが、マシンの材料フローは自動制御されます。

特殊モード

「特殊モード」では、NC Start もマシンの材料フローもどちらも自動制御されません。

オフライン

「オフライン」とは、ホストコンピュータとの接続が切断された状態をいい、すなわちホストコンピュータにデータは送信されません。ホストコンピュータからの RPC が到着したことで RKS が接続の再構築を検出すると、オフラインは解除されます。

ツールデータ用フラグ

ツールデータ識別子を使用して、3 セットのツールデータからホストコンピュータに転送する 1 セットを選択することができます。これらのデータセットに含まれるデータエリアは環境設定プログラムによって定義されます。識別子 21、22 および 23 を使用することができます。識別子はツールデータと一緒にホストコンピュータに転送されます（パート 1 4.1「ツールデータ」参照)。

マガジン番号、ロケーション番号、T 番号：

リポートされるツールは、マガジン番号とロケーション番号、あるいは T 番号によって特定されます。T 番号が特定されると、マガジン番号とロケーション番号は 0 にセットされなければなりませんし、逆の場合も同様です。

PLC リクエストの場合：「リポートツール」RKS がリクエストされデータ要素が読み込まれます：

- マガジン番号 ,
- ロケーション番号 ,
- T 番号 ,
- ツールデータ用フラグ ,

その後、ホストコンピュータにツールデータが送られます。そして、RKS はこれらのデータ要素を削除します（データ要素にゼロが書き込まれる)。

（注）注記

ツールの荷重およびツールの除荷時に発生するツールメッセージは、PLC ユーザープログラムでは開始することはできません（詳細については、パート 1 を参照してください)。

ドッキング位置数

マシンのドッキング位置数は、マシンの作動中は静的に保存されます。この数は、インタフェースのドッキング位置ブロックの数と一致します（1.3 「ドッキング位置データ」参照)

プログラム割当て数

マシンのプログラム割当て数は、マシンの作動中は静的に保存されます。この数は、インタフェースのプログラム割当てブロックの数と一致します（1.4 「NC プログラム割当て」参照)。

マシンステータス

表 1.6 マシンステータス

ビット番号	機能	アクセス元
0	マシンは起動されている	PLC
1	マシンの欠陥が検出された	PLC
2	マシンの再始動	PLC

NC モード :

表 1.7 NC モード

ビット番号	機能	アクセス元
0	Automatic　自動	PLC
1	MDA	PLC
2	JOG	PLC
3	TEACH IN	PLC

マシンステータスおよびNC モードは、R_MACHINE_H を使用してホストコンピュータにリポートされますが、コンピュータリンクサーバ上では評価されません。

予約 1 および 2 :

これらの変数は、機械メーカが PLC を介して要求した場合に使用されます。変数は R_MACHINE_H () を使用してホストコンピュータにリポートされますが、これらの変数は通常はホストコンピュータでは処理されません。

1.3 マシンのドッキング位置データ

ドッキング位置データは各々がマシンステーションを記述します（加工ステーション、イン／アウトステーション)。マシンステーションの数は、グローバルデータ内の「ドッキング位置数」データ要素に保存されます。

表 1.8 ドッキング位置データ

データ要素	略名称	データタイプ	アクセス元
ドッキング位置 status	DockPos Status	Byte	PLC
ワークキャリアステータス	WPCStatus	Byte	PLC/RKS
ワークキャリア	WPC	Byte[6]	PLC
加工サイド	ClampCube Side	Word	PLC/RKS
追従加工	FB	Byte	RKS
予約 1	Reserve1	Byte	PLC

ドッキング位置ステータス

- Bit 0 = フォールト
- Bit 1 = ディスエーブル

ビットアレイは、ドッキング位置の実際のステータスを記述します。ビットアレイは、PLC によって設定されます。イネーブルになっているビットがない場合、ドッキング位置が使用できます。I/O シグナルに応じて**「フォールト」**ビットが設定され解除されます。フォールトとの原因は、リポート機能（-> R_REPORT, パート 1 5.5「メッセージ」参照）を使用してコンピュータにリポートされます。PLC は、「フォールト」ステータスを有するステーション同士のパレットトランスポートは実行しません。ドッキング位置が**「ディスエーブル」**になっている場合、トランスポートシステムによってアプローチすることはできません。

ワークキャリアステータス：

表 1.9 ワークキャリアステータス

ビット番号	機能	アクセス元
0	新規到着 (プログラム割当て無し)	PLC
1	加工の要求 (プログラム割当て有り)	RKS
2	プログラム選択を準備する	PLC
3	プログラムの選択完了	RKS
4	加工が進行中	PLC
5	加工の終了	PLC
6	加工の中断	PLC
7	加工の要求なし (バッファリングのみ)	PLC

新規到着

PLC は、ステータス「新規到着」を新規に到着したワークキャリアに割り当てます。(例外：加工の要求なし)。このステータスからホストコンピュータがプログラム割当てを実行します。プログラム割当てが完了すると、コンピュータリンクサーバによって「加工要求」ステータスが設定されます。

加工要求

PLC は、現在実行中の加工作業が終了し次第、ステータス「加工要求」を使用してワークキャリアに「プログラム選択を準備する」ステータスを設定します。

プログラム選択

プログラム選択が完了すると、すなわちホストコンピュータによってパレットに割当てられたプログラムが NCK にロードされ実行が選択されると、コンピュータリンクサーバがアクティブなワークキャリアに対して「プログラム選択完了」ステータスをイネーブルにします。これで PLC は NC Start をトリガすることができます。必ず全ての安全基準が満たされて初めて PLC によって NC Start がトリガされるように確認することは、製造業者の責務であります（例、安全ドアの閉鎖など)。

加工が進行中

PLC は、加工が開始されると「加工が進行中」ステータスをセットします。

加工作業が完了すると、PLC は対応するワークキャリアに「加工の終了」ステータスをセットします。ステータス「加工の終了」を持つワークキャリアは、PLC によってアンローディングステーションに個別にトランスポートされます。

加工の終了

「追従加工」フラグが設定されている場合、ワークキャリアは加工ステーションに残留します。「加工の終了」ステータスに応じて、コンピュータリンクサーバが再び「加工の要求」ステータスをセットします。このアクションに応じて、PLC は「プログラム選択を準備する」ステータスを使用して RK サーバに追従加工用のプログラムを選択するようリクエストします。以下の手順は最初の加工作業と同様です。

加工の中止

「加工の中止」ステータスは、フォールトの後もワークキャリアが処理されることがないように設定されます。このフラグは主に無人生産中に発生します。このフラグを持つワークキャリアは、他のマシン上でその後の作業に使用することはできません。ただし、保管ロケーションにトランスポートすることはできます。

加工の要求なし

トランスポート制御バッファのためだけに設置されたワークキャリアは、「新規到着」ステータスではなく「加工の要求なし」ステータスに割当てられます。トランスポート制御系はこのデータを PLC に転送します。ホストコンピュータはこのステータスを持つワークキャリアにはプログラムを割当てません。

ワークキャリア：

ドッキング位置に現在位置決めされているワークキャリアの名称（例、"WST01"）。このデータは、PLC によって入力されます。それによって、確実にデータはトランスポートシステムから、あるいは直接ワークキャリアから転送することができます。ドッキング位置にワークキャリアが　1 つもない場合、アレイにはバイナリ 0 が入力されます。

追従加工

このフラグは、加工ステータス「プログラム選択完了」がセットされると同時にコンピュータリンクサーバによってイネーブルにされます。現在の作業に引き続き追従加工作業が行われるかどうかを PLC に伝達します。PLC はこの情報を使用してマシン内のワークキャリアのトランスポートを制御します。

加工サイド

このデータは、加工ステータス「プログラム選択完了」がセットされると同時にコンピュータリンクサーバによってイネーブルにされます。PLC は、この値を使用して加工用のクランプキューブをサイドを設定したり、またはこの値を NCK に渡します。

キューブを使用して、コンピュータリンクはプログラム割当てを基準にサイド用加工の順番を定義します。PLC が送った加工順序を修正する必要がある場合、プログラム割当てデータはコンピュータリンクサーバの環境設定ファイルに入力することで別の PLC データブロックにミラーすることができます。それにより、データへの PLC 読み込みアクセスが可能になります。PLC によって選択された加工サイドは、コンピュータリンクサーバの「加工サイド」フィールドにリポートされます。これは、加工ステータス「プログラム選択完了」が設定されると同時に行われます。コンピュータリンクサーバは、PLC によって指定されたサイドに対するプログラム選択を実行します。その後の手順は同じです。

1.4 NC プログラム割当て (オプション)

このインタフェースはオプションです。このインタフェースは、ホストコンピュータではなく PLC がそれぞれのサイドの加工順序を指定するような場合、クランプキューブを使用するワークキャリアに使用できます。これらのデータは RKS が管理し、ここでは読み取り用として PLC によってミラーされます。

NC プログラム割当てブロックの数は、グローバルデータのデータ要素「プログラム割当て数」に保存されます（1.3「マシンのドッキング位置データ」）。

表 1.10

データ要素	略名称	データタイプ	アクセス元
ワークキャリア	WPC	Byte[6]	RKS
加工サイド	ClampCubeSide	Word	RKS
NC プログラムフラグ	NCProgramMark	Byte	RKS
加工ステータス	Status	Byte	RKS
NC プログラム	NC program	Byte[128]	RKS

ワークキャリア，加工サイド

NC プログラム割当ては、NC プログラムを「ワークキャリア」および「クランプキューブサイド」（加工サイド）に割当てます。

NC プログラムフラグ

NC プログラムフラグを使用すると、クランプキューブの複数サイドに同一の NC プログラムが使用されているかどうか表示することができます。RKS によるエントリでは、ワークキャリアの第 1 サイドには "NCProgramMark" = 1 が設定されます。次のサイドが異なる NC プログラムを使用している場合、"NCProgrammMark" = 2 を設定するなど、以下同様に設定します。今、第 3 のサイドが第 1 サイドと同じ NC プログラムを使用しているならば、サイド 1 と同一の "NCProgrammMark" が割当てられます。このように設定することにより、同一の NC プログラムを使用するサイドが連続して加工されるような実行順序に制御することができるようになります。

同一の NC プログラムを使用して加工されるクランプキューブサイドには、同一の NC プログラムフラグに割当てられます。

加工ステータス

- Bit 1 = 加工のプラン
- Bit 4 = 加工が進行中
- Bit 5 = 加工の終了

リストのサイズ

NC プログラム割当てのリストの大きさは設定可能で、必要な長さを DB に設定できます。大きさを求めるには、マシンのドッキング位置数にクランプキューブの使用可能なサイドの最大数を掛けます。

2 データブロックインタフェースのプロシージャ

2.1 説明

通信は、前述のインタフェース DB を使用して行われます。以下のセクションでは、どの構成要素がインタフェースフィールドとデータの書き込みや読み出しのやり取りを行うか、またそれがいつ行われるかについて述べています。

2.1.1 ワークキャリアの到着

ワークキャリアがマシンのドッキング位置に到着すると、PLC はワークキャリアの名前を読み取るか、あるいはトランスポートシステムがワークキャリアの名前をマシンに渡す必要があります。さらに、トランスポートシステムは、ワークキャリアが加工用の供給を受けているか、あるいはバッファリングのみであるかを、マシンに通知しなければなりません（情報がホストコンピュータからマシン [NC4WPC_M] にリポートされない場合）。トランスポート制御系とマシン間で行われるこの情報の転送に使用されるプロシージャついては、本マニュアルでは説明していません。

PLC は、ドッキング位置ステータス、ワーク番号およびワークステータスを、対応するインタフェース DB に書き込み、そしてリクエストフラグを作成してステータスシグナルに変更があったことを示し、変更トリガを増やします。
次いで、RKS はホストコンピュータに R_MACHINE_H () を送信します。

2.1.2 PLC/NCK と RKS とのプロダクションダイアログ

加工を必要とするワークキャリアがマシンに到着したことを R_MACHINE_H () から検出すると、1 またはそれ以上の NC プログラム割当てが NC4WPC_M () に送られます。RKS はこの情報を内部リストと、オプションで NC プログラム割当用のインタフェースに入力します。

R_MACHINE 用データの決定

ステータスやドッキング位置データのインタフェースは、YS 840DI モード (Automatic, MDA, JOG) およびプログラムステータスと一緒に読みこまれるなければなりません。ホストコンピュータにリポートされたモードは、コンピュータリンクモードと YS 840DI モードのコンビネーションとなります。

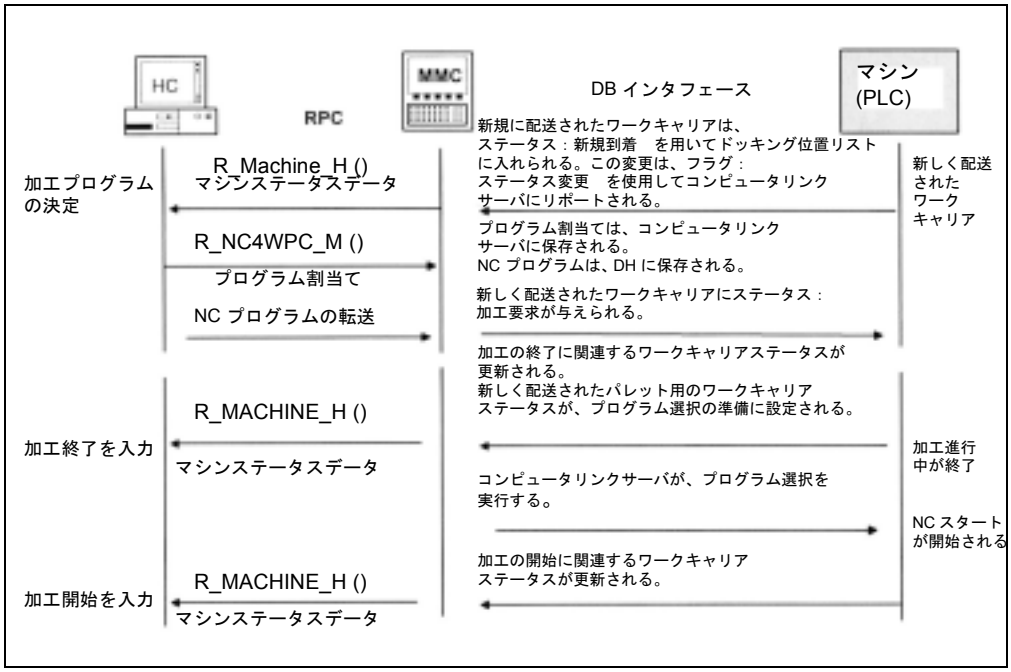

HC
RPC
MMC
DB インタフェース
マシン
(PLC)
新規に配送されたワークキャリアは、
ステータス：新規到着　を用いてドッキング位置リスト
に入れられる。この変更は、フラグ：
ステータス変更　を使用してコンピュータリンク
サーバにリポートされる。
新しく配送
された
ワーク
キャリア
R_Machine_H ()
マシンステータスデータ
加工プログラム
の決定
R_NC4WPC_M ()
プログラム割当て
プログラム割当ては、コンピュータリンク
サーバに保存される。
NC プログラムは、DH に保存される。
NC プログラムの転送
新しく配送されたワークキャリアにステータス：
加工要求が与えられる。
加工の終了に関連するワークキャリアステータスが
更新される。
新しく配送されたパレット用のワークキャリア
ステータスが、プログラム選択の準備に設定される。
R_MACHINE_H ()
加工終了を入力
マシンステータスデータ
加工進行
中が終了
コンピュータリンクサーバが、プログラム選択を
実行する。
NC スタート
が開始される
加工の開始に関連するワークキャリア
ステータスが更新される。
R_MACHINE_H ()
加工開始を入力
マシンステータスデータ

図 2.1 流れ

3 コンピュータリンクの相互作用プログラム

3.1 コンピュータリンクの相互作用プログラム

コンピュータリンクの相互作用プログラムには、次のような機能があります：

- コンピュータリンクのステータスを選択する
- ホストコンピュータに NC プログラムをリクエストする
- ホストコンピュータに NC プログラムを転送する

相互作用ブプログラムは、コンロトーラのユーザーツリーで独立エリアアプリケーションとして利用できます。

3.2 コンピュータリンクのステータス

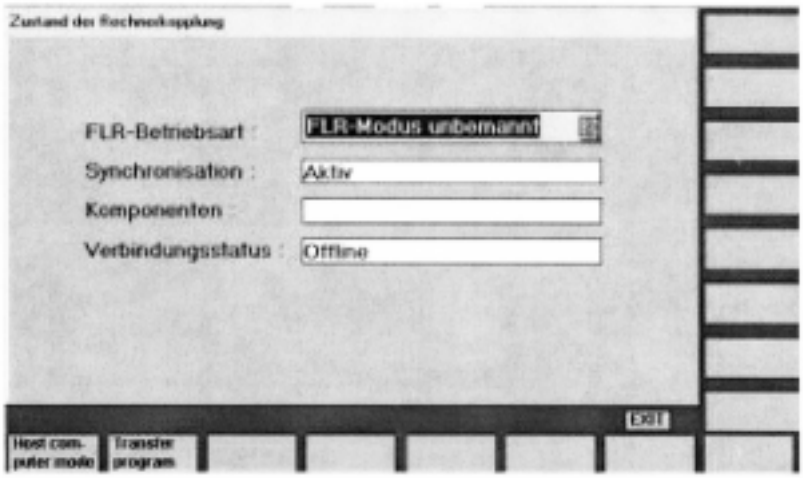


図 3.1 スクリーンでホストコンピュータのモードを変更することができます

コンピュータリンクモード

- ホストコンピュータモード　無人
- ホストコンピュータモード　有人
- 手動モード
- 特殊モード

コンピュータリンクの関連ステータスも表示されます。

3.3 NC プログラムの転送

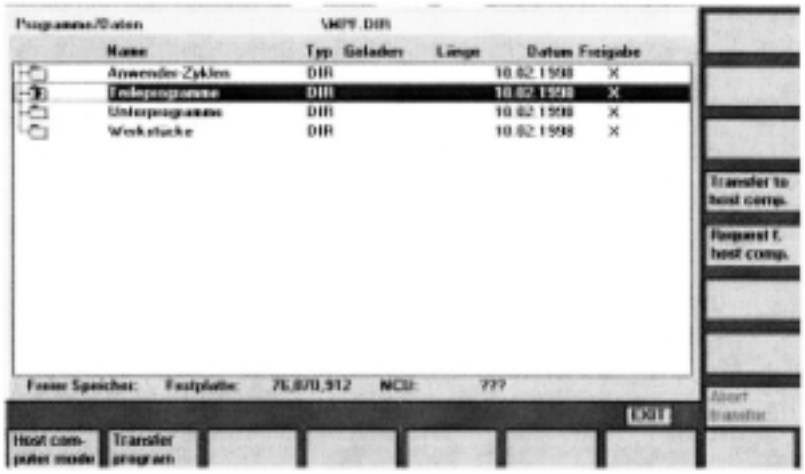


図 3.2 NC プログラム／ファイルの転送

ユーザーはスクリーン上で、データ管理システムから製造ホストコンピュータへ NC プログラムや他のファイルを転送することができます。また、ソフトキー Request from host computer をクリックして、**ホストコンピュータからファイルをリクエスト**することもできます。

3.3.1 ホストコンピュータへプログラムを送信する

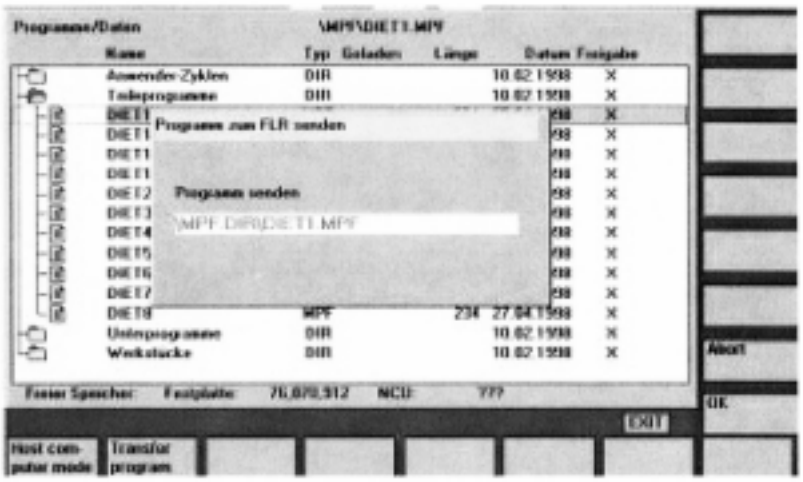


図 3.3 ホストコンピュータへプログラムを送信する

3.3.2 ホストコンピュータからプログラムをリクエストする

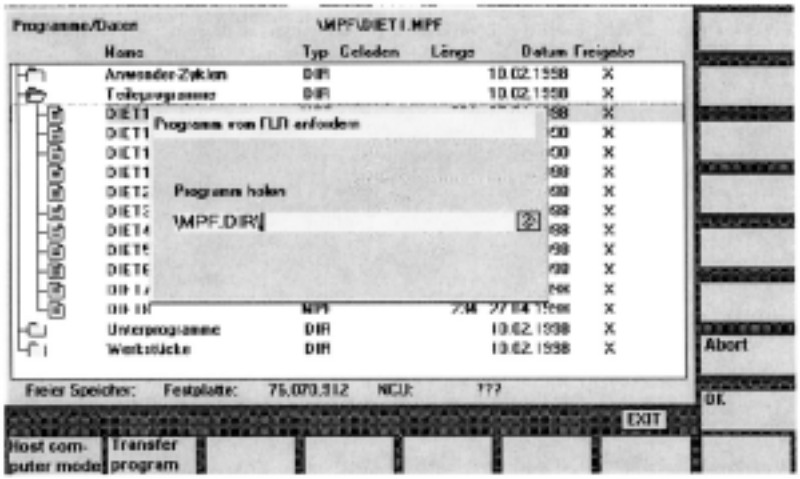


図 3.4 ホストコンピュータからプログラムをリクエストする

スクリーン上で、ユーザーはホストコンピュータからリクエストする NC プログラムを入力することができます。

4 RKS と TPS PLC のインタフェース

4.1 説明

コンピュータリンクソフトウェアと TPS PLC 間で通信を行うには、インタフェース DB が必要です。そのために当社標準 DB が予約されています。DB のセットアップはユーザーが行ってください。DB インタフェースのデータ要素はブロックに構築されていて、各々が何らかの形でインタフェースと関連しています（例、グローバルデータ、トランスポートジョブ、ドッキング位置データ）。このブロックは、表のフォーマットで表示されます。全てのブロックがインタフェース DB に連続して保存されます。

タイプ "int(WORD)" および "Long(DWORD)" のバイナリデータ要素は、DB 内に S7 フォーマット（リトル・エンディアン）で保存されます。MMC がこれらのデータ要素にアクセスするときには、データをインテルフォーマット（ビッグ・エンディアン）に変換します。名前を表すデータ要素は、ASCII 文字を使用するバイトフィールドとして構築されます。

このインタフェースは表のフォーマットで記述されます。"Access " 「アクセス」コラムは、そのフィールドを記述した人を示します。このコラムでは、次のような略語が使用されます：

- RKS　コンピュータ通信ソフトウェア（製造ホストコンピュータと直結）
- PLC　PLC ユーザープログラム

内部リソースの通信負荷を最小にするため、RKS インタフェースに変更がある毎に "request from the PLC"「PLC からリクエストする」を介して中継されます（インタフェースの一部）。RKS はこのリクエストを瞬時に検出します。

4.2 グローバルデータ

表 4.1 グローバルデータリスト

データ要素	略名称	データタイプ	アクセス元	オフセット
PLC からのリクエスト	ANF_PLC	Byte	PLC/RKS	0
変更トリガ	Trigger	Byte	PLC	1
RKS へのリクエスト	ANF_RKS	Byte	RKS	2
トランスポートジョブ番号	ANZ_TO	Byte	PLC	3
ドッキング位置番号	ANZ_HALT	Byte	PLC	4
マシンステータス	Machine Status	Byte	PLC	5
NC モード	Machine Mode	Byte	PLC	6
マシンモード	MODE	Byte	PLC	7
予約 1	Reserve 1	Word	PLC	8，9
予約 2	Reserve 2	Word	PLC	10，11

PLC からリクエストする

表 4.2 「 PLC からリクエストする」のステータスリスト

ビット番号	機能	アクセス元
0	トランスポートジョブデータ変更	PLC - 1/RKS - 0
1	ドッキング位置データ変更	PLC - 1/RKS - 0
2	ステータス変更	PLC - 1/RKS - 0

PLC は、このバイトを使用してインタフェースの変更を指示します。PLC はリクエストバイトをイネーブルにすると必ずトリガバイトに次のビットをイネーブルにすることになります（以下を参照してください）。

処理が終了した後に RSK によってゼロに設定されて初めて、PLC は再びリクエストバイトに書き込めるようになります。

トランスポートジョブ変更

トランスポートジョブの実行中に PLC のステータスが変更された場合は必ず、PLC によって「トランスポートジョブ変更」が設定されます。

ドッキング位置データ変更

ドッキング位置で PLC がデータが変更された場合には必ず、PLC によって「ドッキング位置データ変更」が設定されます。

ステータス変更

PLC は、ホストコンピュータに報告する必要があるステータス変更（マシンモード、マシンステータス、NC モード）の全てに「ステータス変更」を設定します。そして、RKS はホストコンピュータに R_ TPS_H () を送ります。

変更トリガ

コンピュータリンクは、このバイトに生じた変更に即時に対応します。

PLC は、PLC で複数の変更が生じた際にこのバイトに 1 ビットをセットします。新規トリガのそれぞれに対して、PLC は次のビットをセットし、最後のビットをリセットします ; ビット 7 の後は、再びビット 0 から始まります。

RKS にリクエストする :

ビット番号	機能	アクセス元
0	同期フラグ	RKS
1	ユニットを停止	RKS
2	ユニットを起動	RKS
3	オフライン	RKS

同期フラグ

「同期フラグ」は、ホストコンピュータによってセットおよびリセットされます。(-> /1/C_SYNCH_M () パート 1 5.14「同期」参照) 同期プロセスの期間中、トランスポートシステムは同一でなければなりません。PLC は、パレット動作を実行することはできません。

ユニットを停止する

「ユニットを停止する」は、ホストコンピュータによってセットされます。(-> /1/ C_MODE_M () パート 1 5.13「MODE 選択」参照)「ユニットを停止する」は、PLC がユニット（ドライブ）を停止するようリクエストします（マシンモード、ユニットの停止も参照してください)。

ユニットを起動する

「ユニットを起動する」は、ホストコンピュータによってセットされます。(-> /1/ C_MODE_M () パート 1 5.13「MODE 選択」参照)「ユニットを起動する」は、PLC はユニット（ドライブ）を起動するようリクエストします。

オフライン

ホストコンピュータとの接続が切断されると、コンピュータリンクによって**オフライン**がセットされます。機械メーカは、例えば、このオフラインフラグを評価し、警告灯を管理することができます。

トランスポートジョブ数：

トランスポートジョブ数は、ジョブの実行中に静的に保存されます。この値が、ホストコンピュータから PLC に転送されるトランスポートジョブの最大数を指定します。この数は、インタフェースのトランスポートジョブブロック数と同じになります。

ドッキング位置数：

ドッキング位置数は、タスクの実行中にスタティックに保存されます。この数は、インタフェースのドッキング位置データブロック数と同じです。

マシンのステータス：

ビット番号	機能	アクセス元
0	マシンがアクティブ	PLC
1	マシンフォールトの検出	PLC
2	マシンの再始動	PLC

NC モード：

ビット番号	機能	アクセス元
0	自動	PLC
1	MDA	PLC
2	JOG	PLC
3	TEACH IN	PLC

マシンステータスおよび NC モードは、R_ TPS _H を使用してホストコンピュータにリポートされますが、コンピュータリンクサーバ上では評価されません。

マシンモード：

表 4.3

ビット番号	機能	アクセス元
0	ホストコンピュータモード	PLC
1	ユニットの停止	PLC

ホストコンピュータモード

ホストコンピュータモードは、ユーザーダイアログを介して PLC によってセットおよびリセットされます。

ユニットの停止

このステータスが検出されると、PLC によって**ユニットの停止**が設定されます。このリクエストは、リクエストフラグのビット 1 に設定されます（以下を参照してください）。

予約１、予約２：

これらの変数は、機械メーカが PLC を介して要求した場合に使用されます。変数は R_MACHINE_H () を使用してホストコンピュータにリポートされますが、これらの変数は標準ではホストコンピュータでは処理されません。

4.3 トランスポートジョブ

表 4.4 トランスポートジョブのデータ

データ要素	略名称	データタイプ	アクセス元
ソースドッキング位置	SdockIdx	Word	RKS
デスティネーションドッキング位置	DdockIdx	Word	RKS/PLC
ワークキャリア	WPC	Byte[6]	RKS
ワークキャリアタイプ	WPCTyp	Byte	RKS
バッファリング用	BufferFlag	Byte	RKS
プライオリティ	Priority	Byte	RKS
チェーン番号	ChainNum	Byte	RKS
伝達媒体	Vehicle	Byte	RKS/PLC
トランスポートステータス	TPOStatus	Byte	RKS/PLC

ソースおよびデスティネーションドッキング位置 :

ソースおよびデスティネーションドッキング位置は、ホストコンピュータに使用されているようなリファレンスではなく、インデックスです。このインデックスは、トランスポートシステム上のドッキング位置データ内にドッキング点を位置決めします。PLC は、同じインデックスを使用してこのドッキング位置の座標を管理します。RKS は、Ini ファイルの割当てリストを使用してこの機能を構築します。

ワークキャリア

PLC は、ワークキャリアの名称を使用して妥当性検査を行います :

ワークキャリアタイプ :

ワークキャリアタイプは、データの追加項目で、ワークキャリアのタイプやサイズを含むことができます。

バッファリング用 :

ワークキャリアがマシンにトランスポートされるが加工が行われない場合に、バッファフラグが設定されます（補助バッファステーション）。トランスポート制御系（ TPS ）は、この情報をマシンに渡します。

プライオリティ :

プライオリティは、補助単位の情報です。複数のジョブが転送される場合、プライオリティを使用し、ジョブを処理する順番を制御することができます。

チェーン番号 :

チェーン番号は、補助単位の情報です。2 つの保存ロケーションを持つトランスポート伝達媒体と 1 つしかドッキング位置を持たないマシンでは、チェーン番号を用いて 2 つのトランスポートジョブを理論的に合体させることができます。

伝達媒体 :

伝達媒体は、補助単位の情報です。複数のトランスポート伝達媒体を持つトランスポートシステムでは、このパラメータを使用してトランスポートに使用する伝達媒体を定義することができます。

トランスポートステータス :

TPOStatus:

表 4.5

ビット番号	機能	アクセス元
0	新規トランスポートジョブ	SINCOM
1	トランスポートジョブの開始	PLC
2	伝達媒体上の WPC	PLC
3	ジョブの完了	PLC
4	エラー , 実行不能なジョブ	PLC
5	エラー , 別のデスティネーションにアプローチ (PLC によって DdockPos に代替のデスティネーションが入力される)	PLC

新規トランスポートジョブが起動されると、SinCOM は TPOStatus のビット 0 をイネーブルにし、ビット 1 から 7 をクリアにします。

処理のステータスにしたがって、ビット 1 から 5 がそれぞれ PLC によって設定されます。

4.4 トランスポートシステムのドッキング位置データ

表 4.6

データ要素	略名称	データタイプ	アクセス元
第 1 ドッキング位置ステータス	DockPosStatus	1 byte	PLC
第 1 ワークキャリアステータス	WPCStatus	1 byte	PLC
第 1 ワークキャリア	WPC	6 bytes	PLC
...			

ドッキング位置の数は、設定可能です。この数が DB に必要な長さを決定します。

ドッキング位置ステータス

表 4.7

ビット番号	機能	アクセス元
0	フォールト	PLC
1	ディスエーブル	PLC
2	使用中	PLC

ビットアレイは、ドッキング位置の実際のステータスを記述します。ビットアレイは、PLC によって設定されます。イネーブルになっているビットがゼロの場合、ドッキング位置が使用できます。I/O シグナルに応じて**「フォールト」**ビットが設定されたり解除されたりします。フォールトの原因は、リポート機能（-> R_REPORT, パート 1 5.5「メッセージ」参照）を使用してコンピュータにリポートされます。PLC は、**「フォールト」**ステータスを有するステーション同士のパレットトランスポートは実行しません。ドッキング位置が**ディスエーブル**になっている場合、トランスポートシステムによってアプローチすることはできません。

「割当て済」は、まず内部使用のために、PLC によって設定することができます。ホストコンピュータ（および RKS ）は、ドッキング位置が割当てられているかどうかをワークキャリアから検出することができます。

ワークキャリアステータス

使用しません。

ワークキャリア

現在ドッキング位置に位置決めされているワークキャリアの名前（例、"WST01"）。このデータは、PLC によって入力されます。ドッキング位置にワークキャリアがない場合、アレイにはバイナリ 0 が入力されます。

4.5 TPS へのトランスポートジョブ

4.5.1 機能シーケンス

インタフェースエントリ

RKS は、トランスポートジョブ用インタフェースの第 1 フリーデータレコードに、ホストコンピュータから受け取ったデータを入力します。ソースおよびデスティネーションドッキング位置は、ホストコンピュータで使用されるドッキング位置名は PLC に認識されていないため、インデックスを用いてリファレンスさせる必要があります。

PLC アクション

トランスポート PLC はジョブを開始する際に、ステータス TPOStatus = job started「ジョブの開始」を設定しなければなりません。RKS がこのステータスを読みこんでホストコンピュータに伝達できるように、「PLC にリクエストする」バイト内でビット 0 がイネーブルになる必要があります。RKS はこのリクエストバイトをリセットし、インタフェースから送られた全てのリポートデータを読み込み、ホストコンピュータに R_TPS_H コールを送信します。

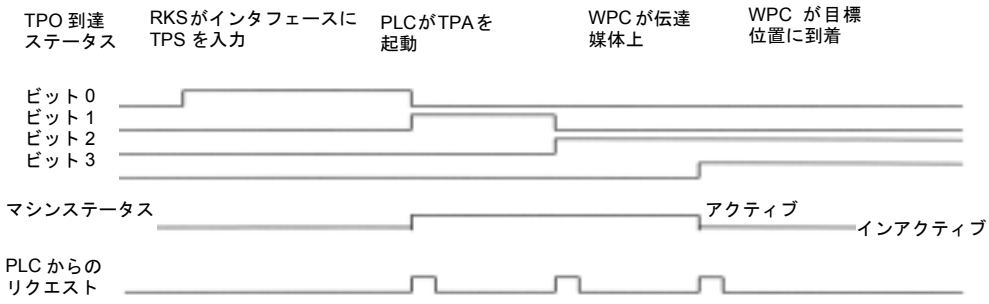
ワークキャリアがすでにトランスポート伝達媒体に転送されている場合、PLC は TPOStatus = " 伝達媒体の WPC" をイネーブルにし、さらに「PLC にリクエストする」バイトのビット 0 をイネーブルにしてホストコンピュータにもう一つ別の R_TPS_H コールを送信することができます。この中間ステータスのリポートは絶対に必要というわけではありません。トランスポート作業の終了時にリクエストビットをイネーブルにすることもできます。この設定をすると、また別の R_TPS_H コールがホストコンピュータに送られます。SDockPos は、DockPos[0] に転送されます。トランスポート伝達媒体の数は、DockPos[1] に入力されます。

ワークキャリアがすでにそのデスティネーションドッキング位置に配送されている場合、PLC は TPOStatus = "Job complete" をイネーブルにして、再び「PLC へのリクエスト」バイトにビット 0 をイネーブルにしなければなりません。

トランスポート PLC は、ジョブを実行できない（例えば、ソースドッキング位置が空である、あるいはディスエーブルである）場合、ステータス TPO-Status = "error, job not executable" を設定する必要があります。指定したデスティネーションに送られずに別のドッキング位置にワークキャリアが配送された場合、トランスポート PLC は DDockPos に新規デスティネーションを入力し、ステータス TPO-Status = "error, alternative destination approached" をイネーブルにします。

トランスポートジョブの終了時に、MachineStatus = "inactive", DockPos[0] = DdockPos などによって別の　R_TPS_H コールが開始されます。

TPOStatus が変更されて、その変更が R_TPS_H () によってホストコンピュータに送られると必ず、「PLC へのリクエスト」バイトでビット 0 がイネーブルになります。



4.6 PLC レベルでのユーザーによる手動トランスポート操作

コンピュータリンクは、ホストコンピュータと接続されている場合、どのオペレーティングモードでも、ステータス変更やワークキャリア動作をホストコンピュータにリポートすることができます。このコンピュータリンクにより、PLC レベルでユーザーによって手動で実行されたワークキャリア動作はホストコンピュータにリポートされます。そのため、PLC プログラムは該当するソースおよびデスティネーションドッキング位置番号をトランスポートジョブインタフェースに入力し、TPOStatus を「トランスポートジョブの開始」または「ジョブ完了」に設定する必要があります。「PLC へのリクエスト」バイトのビット 0 が 0 に設定されると、データは RSK に読み取られ、ホストコンピュータに転送されます。ワークキャリアの動作はプラントイメージに表示することができます。

5 環境設定データ

5.1 説明

SCCONFIG 環境設定ファイル

コンピュータリンクに必要な環境設定データは、レジストリに入力されます。SCCONFIG.EXE を使用して、これらのデータを作成したり変更することができます。

環境設定データは、マシン関連データとコンピュータ関連データに分類されます。コンピュータリンクにより複数のコンピュータが接続されている場合、それぞれのコンピュータ用のコンピュータ関連データ（名前や IP アドレスなど）を持つデータレコードは 1 つです。このデータレコードが、それぞれのコンピュータに対して、そのコンピュータがどの RCP を受け取り、そのコンピュータにはどの RPC を送信すべきかを決定します。

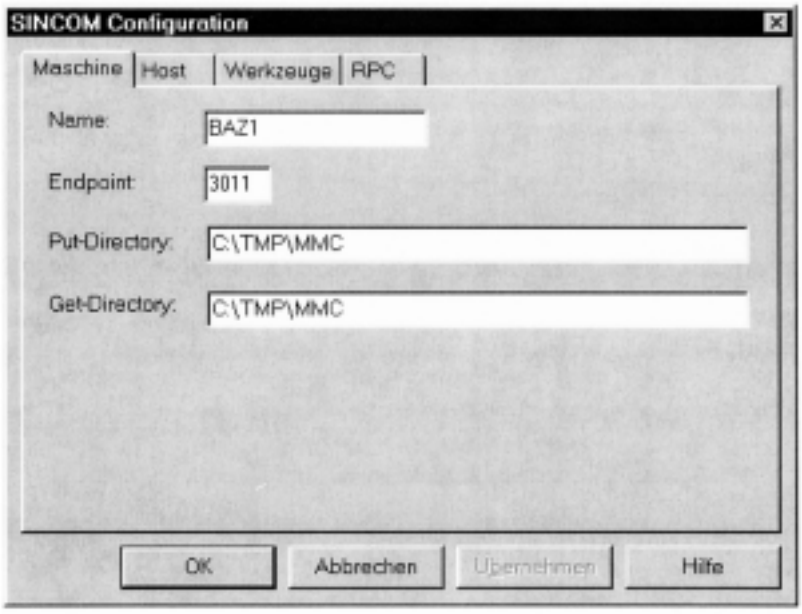


図 5.1

Name

Name「名前」は、それぞれの RPC コールを持つセカンドパラメータとして入力されるマシンの名称です。

Endpoint

IP アドレスに加えて エンドポイント番号が、RPC には必要です；通常、仮番号を使用できます。

Put directry

Put directory は、SINCOM がファイルをホストコンピュータに転送するまで保存しておく臨時ディレクトリとして機能します。

Get directry

Get directory は、ファイルがホストコンピュータから転送されるときの臨時ディレクトリとして機能します。このファイルは、SINCOM によって処理されない場合にはこのディレクトリに保存します。

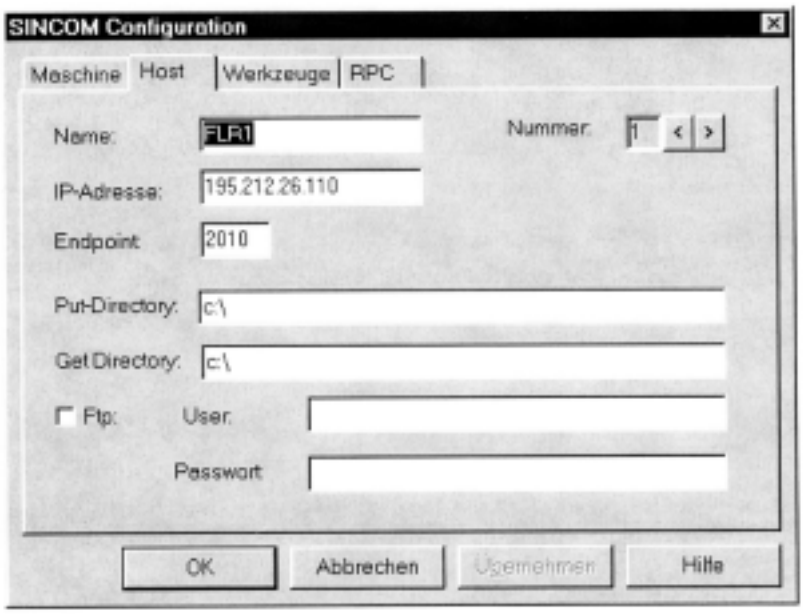


図 5.2

概要

SINCOM は、複数のホストコンピュータとの通信を可能にします。それぞれのホストコンピュータに関して、このような画面書式に記入する必要があります。ホストコンピュータの切換えは、右上の矢印によって行います。

ホスト

Name は、それぞれの RPC コールを持つファーストパラメータとして入力される「ホスト」の名前です。

IP Adress

IP Adress はホストコンピュータ用の IP アドレスです。

Put Directory

Put directory は、ホストコンピュータが SINCOM へファイルを転送するのに使用するディレクトリです。

Get Directory

Get directory は、そこへホストコンピュータが SINCOM からファイルを転送するディレクトリです。

Ftp.

ホストコンピュータがウィンドウズコンピュータ（NT または Win95 ）の場合、Ftp は選択解除されなければなりませんが、ユーザーやパスワードフィールドには関係ありません。

UNIX コンピュータなど、その他のホストコンピュータの場合、データは FTP を介して転送されます。Ftp ボックスを選択し、ユーザーおよびパスワードにホストコンピュータのログオン用の有効なエントリを入力する必要があります。この特定のユーザーが、Put Directory を用いて指定されたディレクトリへの書き込みアクセスならびに Get Directory を用いて指定されたディレクトリへの読み取りアクセスを行わなければなりません。

ディレクトリ

ホストコンピュータ上ならびに MMC 上に専門ディレクトリが存在しているか、あるいは初めて SINCOM が始動されるまでに作成されていなければなりません。SINCOM は、ディレクトリを作成することはできません。

ツールデータの読み取り

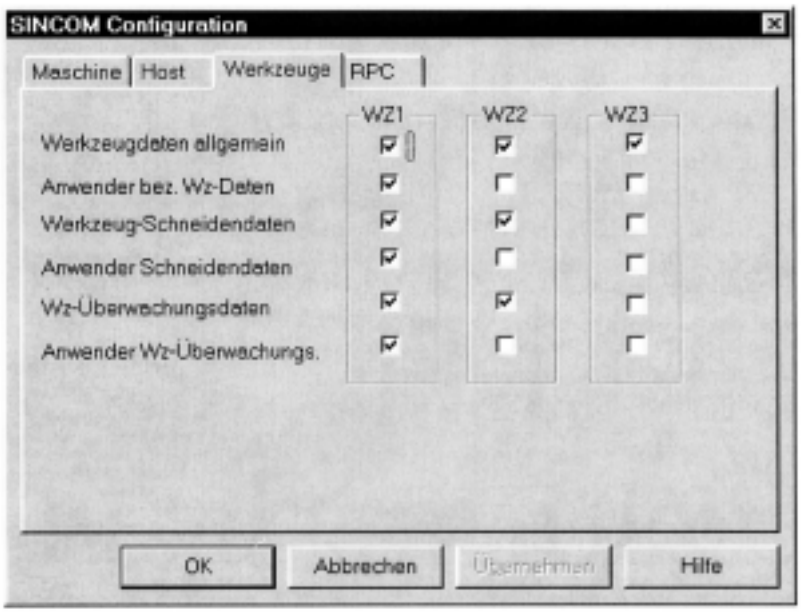


図 5.3

NC ツールのツールデータの読み取りには、データ保存エリア毎に読み取りコールが必要となります。

ツールデータをホストコンピュータに転送する場合、毎回ホストコンピュータにツールデータ全体を転送する必要はありません。したがって、3 つのツール構造が定義されます（WZ1 ～ WZ3）。通常、第 1 構造には既存のデータ保存エリアが全て含まれます（ユーザー関連データ保存エリアがない場合、選択解除してください）。他の 2 つの構造は、範囲を狭めて必要に応じて定義します。

RPC 機能

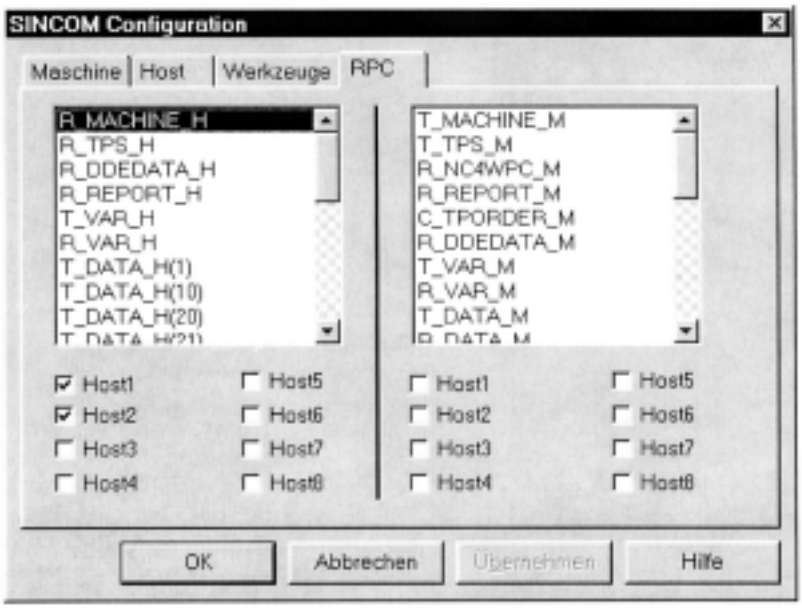


図 5.4

それぞれの RPC 機能に対して、どのホストコンピュータから送信するか（RPC マスク、左側）や、どのホストコンピュータからどの RPC 機能を受信するか（RPC マスク、右側）を定義する必要があります。

5.1.1 環境設定データの例

環境設定データを使用する ASCII ファイルは次のようなものになります :

```
[HKEY_CURRENT_USER\Software\SIEMENS\SINCOM\Host1]
"Name"="FLR1"
"IpAdr"="195.212.26.110"
"Endpoint"="2010"
"FTPUser"=""
"FTPPassword"=""
"HostDirGet"="h:\\"
"HostDirPut"="h:\\"
"Ftp"=dword:00000000

[HKEY_CURRENT_USER\Software\SIEMENS\SINCOM\RPC_H]
"R_MACHINE_H"=dword:0000ffff
"R_TPS_H"=dword:0000ffff
"R_DDEDATA_H"=dword:0000ffff
"R_REPORT_H"=dword:0000ffff
"T_VAR_H"=dword:00000000
"R_VAR_H"=dword:00000000
"T_DATA_H(1)"=dword:00000001
"T_DATA_H(10)"=dword:00000001
"T_DATA_H(20)"=dword:00000001
"T_DATA_H(21)"=dword:00000001
"T_DATA_H(22)"=dword:00000001
"T_DATA_H(23)"=dword:00000001
"T_DATA_H(26)"=dword:00000001
"T_DATA_H(27)"=dword:00000001
"T_DATA_H(28)"=dword:00000001
"T_DATA_H(90)"=dword:00000001
"R_DATA_H(1)"=dword:000000ff
"R_DATA_H(10)"=dword:000000ff
"R_DATA_H(20)"=dword:000000ff
"R_DATA_H(21)"=dword:00000001
"R_DATA_H(22)"=dword:00000001
"R_DATA_H(23)"=dword:00000001
"R_DATA_H(50)"=dword:00000001
"R_DATA_H(90)"=dword:00000001

[HKEY_CURRENT_USER\Software\SIEMENS\SINCOM\RPC_M]
"T_MACHINE_M"=dword:0000ffff
"T_TPS_M"=dword:0000ffff
"T_DATA_M"=dword:00000001
"R_DATA_M"=dword:00000001
"R_NC4WPC_M"=dword:00000000
"R_DDEDATA_M"=dword:0000ffff
"R_REPORT_M"=dword:0000ffff
"C_DELETE_M"=dword:00000001
"C_TPORDER_M"=dword:00000001
"R_DATA_M(1)"=dword:00000001
"R_DATA_M(10)"=dword:00000001
"R_DATA_M(26)"=dword:00000001
"R_DATA_M(27)"=dword:00000001
"R_DATA_M(28)"=dword:00000001
"R_DATA_M(90)"=dword:00000001
"C_DELETE_M(1)"=dword:00000001

[HKEY_CURRENT_USER\Software\SIEMENS\SINCOM\Settings]
"Machine"="FMS-TPS_____"
"EndpointMach"="3011"
```

```
"MMCDirGet"="C:\\TMP\\MMC"
"MMCDirPut"="C:\\TMP\\MMC"
"TraceFilesize"=dword:00000005
"TraceLevel"=dword:00000002
"ToolData1"=dword:0000003f
"ToolData2"=dword:00000015
"ToolData3"=dword:00000001
```

A 付録

インタフェース定義言語 (IDL)

使用上の注意

次のファイルは、SCTEST インストレーションのセカンドデリバリディスケットに入っています：

- SCHOST.IDL
- SCMACH.IDL
- SCHOST.ACF
- SCMACH.ACF

IDL ファイルは、機能コールとそのパラメータを記述します。IDL コンパイラ用の ACF ファイルを使用して、内部結合または外部結合が必要であるかどうか決定されます。ホストコンピュータと複数のマシンの通信には、外部結合が使用されます。

添付の ACF ファイルには両方のタイプに該当する説明が含まれていますが、どちらか一方についてのみ記載されています。必要なフォームがアクティブになっていて、かつ不要なフォームが「コメント」になっていることを確認してください。

IDL コンパイラは、これらのファイルからクライアントおよびサーバスタブならびにヘッダを生成します。

HOST WINDOWS NT または WIN95 上で使用される場合、VC++ 開発システムで充分であります。この開発システムにはファイルと IDL ファイルが含まれていますが、含まれていない場合 RPC にとって必要となります。

Microsoft IDL コンパイラをコールする：

MIDL SCHOST.IDL /osf	ホストコンピュータで RPC を使用する場合 (例 , R_MACHINE_H)
MIDL SCHMACH.IDL /osf	SINCOM 上で RPC を使用する場合 (例 , T_MACHINE_M)

DCE に互換性を持たせるには /osf パラメータを示す必要があります。

他のオペレーティングシステムでは、DCE RPC 開発システムが必要です。

! 重要
必ず DCI RPC 開発システムを使用し、絶対に SUN RPC を使用しないでください。

SCTEST インストレーションには、最も重要なプログラムパートの説明、ならびに SINCOM ホストコンピュータアプリケーション用の VC++ 5.0 お試しプログラムのソースが含まれています。

RPC コールのシーケンス：

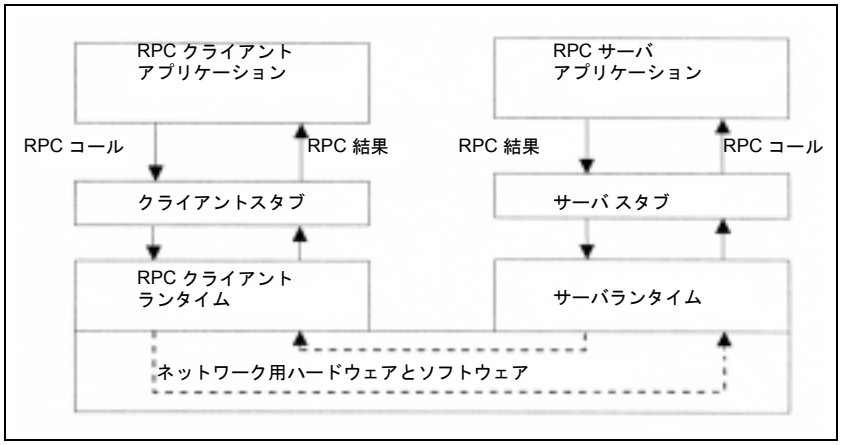


図 A.1 RPC コールのシーケンス .06/98 SINCOM Computer Link

ホストコンピュータ用の機能

```
[ uuid(d3d7d860-c15a-11d0-a0cb-00a0244ce687),
バージョン (1.0),
pointer_default(unique)
]
インタフェース SINCOMHOST
{

const long maxWPCLen = 6;
const long maxMPos = 3;

long R_MACHINE_H([in, string] unsigned char* pszHost,
                 [in, string] unsigned char* pszMachine,
                 [in] long OrderNum,
                 [in] long MachineMode,
                 [in] long MachineStatus,
                 [in, string] unsigned char* pszNCProgram,
                 [in] long ClampCubeSide,
                 [in] long DockPos[maxMPos],
                 [in] long DockPosStatus[maxMPos],
                 [in] unsigned char pszWP C[maxMPos][maxWPCLen],
                 [in] long WPCStatus[maxMPos],
                 [in] long ResInt1,
                 [in] long ResInt2,
                 [in, string] unsigned char* pszResByte );

const long maxTPos = 2;

long R_TPS_H([in, string] unsigned char* pszHost,
             [in, string] unsigned char* pszMachine,
             [in] long OrderNum,
             [in] long MachineMode,
             [in] long MachineStatus,
             [in] long TpOStatus,
             [in] long DockPos[maxTPos],
             [in] long DockPosStatus[maxTPos],
             [in] unsigned char pszWPC[maxTPos][maxWPCLen],
             [in] long ResInt1,
```

```
                [in] long ResInt2,
                [in, string] unsigned char* pszResByte );

const long maxAlarms = 10;

long R_REPORT_H([in, string] unsigned char* pszHost,
                [in, string] unsigned char* pszMachine,
                [in] long OrderNum,
                [in] long Type,
                [in] long Number[maxAlarms],
                [in] long Time[maxAlarms],
                [in] char Flag[maxAlarms],
                [in] long ResInt1,
                [in] long ResInt2,
                [in, string] unsigned char* pszResByte );
long R_MESSAGE_H([in, string] unsigned char* pszHost,
                [in, string] unsigned char* pszMachine,
                [in] long OrderNum,
                [in, string] unsigned char* pszMessage,
                [in] long ResInt1,
                [in] long ResInt2,
                [in, string] unsigned char* pszResByte );
long T_DATA_H([in, string] unsigned char* pszHost,
                [in, string] unsigned char* pszMachine,
                [in] long OrderNum,
                [in] long SFc,
                [in, string] unsigned char* pszName1,
                [in, string] unsigned char* pszName2 );
long R_DATA_H([in, string] unsigned char* pszHost,
                [in, string] unsigned char* pszMachine,
                [in] long OrderNum,
                [in] long SFc,
                [in, string] unsigned char* pszName1,
                [in, string] unsigned char* pszName2,
                [in] long Date,
```

```
                [in] long LastFile );


long T_VAR_H([in, string] unsigned char* pszHost,
                [in, string] unsigned char* pszMachine,
                [in] long OrderNum,
                [in] long VarMode,
                [in, string] unsigned char* pszVarSet,
                [in, string] unsigned char* pszVarDescr );


long R_VAR_H([in, string] unsigned char* pszHost,
                [in, string] unsigned char* pszMachine,
                [in] long OrderNum,
                [in] long VarMode,
                [in, string] unsigned char* pszVarSet,
                [in, string] unsigned char* pszVarDescr,
                [in, string] unsigned char* pszVarData );


long R_DDEDATA_H([in, string] unsigned char* pszHost,
                [in, string] unsigned char* pszMachine,
                [in] long OrderNum,
                [in, string] unsigned char* pszData );


void Shutdown_H(void);
}
```

MMC 用の機能

```
[ uuid(d6542300-c15a-11d0-a0cb-00a0244ce687),
バージョン (1.0),
pointer_default(unique)
]
インタフェース SINCOMMACHINE
{

long T_MACHINE_M([in, string] unsigned char* pszHost,
                [in, string] unsigned char* pszMachine,
                [in] long OrderNum);

long T_TPS_M([in, string] unsigned char* pszHost,
```

```
        [in, string] unsigned char* pszMachine,
        [in] long OrderNum);

long T_DATA_M([in, string] unsigned char* pszHost,
        [in, string] unsigned char* pszMachine,
        [in] long OrderNum,
        [in] long SFc,
        [in, string] unsigned char* pszName1,
        [in, string] unsigned char* pszName2 );

long T_VAR_M([in, string] unsigned char* pszHost,
        [in, string] unsigned char* pszMachine,
        [in] long OrderNum,
        [in] long VarMode,
        [in, string] unsigned char* pszVarSet,
        [in, string] unsigned char* pszVarDescr );

long R_NC4WPC_M([in, string] unsigned char* pszHost,
        [in, string] unsigned char* pszMachine,
        [in] long OrderNum,
        [in, string] unsigned char* pszWPC,
        [in, string] unsigned char* pszNCProg,
        [in] long Date,
        [in] long NCPLength,
        [in] long ClampCubeSide,
        [in] long TpFlag,
        [in] long NCExtern,
        [in] long ResInt1,
        [in] long ResInt2,
        [in, string] unsigned char* pszResByte );

long R_REPORT_M([in, string] unsigned char* pszHost,
        [in, string] unsigned char* pszMachine,
        [in] long OrderNum,
        [in] long Type,
        [in] long Number,
        [in] long ResInt1,
        [in] long ResInt2,
        [in, string] unsigned char* pszResByte );
```

```
long R_MESSAGE_M([in, string] unsigned char* pszHost,
                 [in, string] unsigned char* pszMachine,
                 [in] long OrderNum,
                 [in, string] unsigned char* pszMessage,
                 [in] long ResInt1,
                 [in] long ResInt2,
                 [in, string] unsigned char* pszResByte );

long R_DATA_M([in, string] unsigned char* pszHost,
              [in, string] unsigned char* pszMachine,
              [in] long OrderNum,
              [in] long SFc,
              [in, string] unsigned char* pszName1,
              [in, string] unsigned char* pszName2,
              [in] long Date,
              [in] long LastFile );

long R_VAR_M([in, string] unsigned char* pszHost,
             [in, string] unsigned char* pszMachine,
             [in] long OrderNum,
             [in] long VarMode,
             [in, string] unsigned char* pszVarSet,
             [in, string] unsigned char* pszVarDescr,
             [in, string] unsigned char* pszVarData );

long R_DDEDATA_M([in, string] unsigned char* pszHost,
                 [in, string] unsigned char* pszMachine,
                 [in] long OrderNum,
                 [in, string] unsigned char* pszApplication,
                 [in, string] unsigned char* pszTopic,
                 [in, string] unsigned char* pszItem,
                 [in, string] unsigned char* pszData );

long C_DELETE_M([in, string] unsigned char* pszHost,
                [in, string] unsigned char* pszMachine,
                [in] long OrderNum,
                [in] long SFc,
                [in, string] unsigned char* pszName1,
```

```
                [in, string] unsigned char* pszName2 );

long C_MODE_M([in, string] unsigned char* pszHost,
                [in, string] unsigned char* pszMachine,
                [in] long OrderNum,
                [in] long Mode );

long C_SYNCH_M([in, string] unsigned char* pszHost,
                [in, string] unsigned char* pszMachine,
                [in] long OrderNum,
                [in] long SynchFlag);

long C_TPORDER_M([in, string] unsigned char* pszHost,
                [in, string] unsigned char* pszMachine,
                [in] long OrderNum,
                [in] long SDockPos,
                [in] long DDockPos,
                [in, string] unsigned char* pszWPC,
                [in] long WPCType
                [in] long BufferFlag,
                [in] long Priority,
                [in] long ChainNum,
                [in] long Vehicle,
                [in] long ResInt1,
                [in] long ResInt2,
                [in, string] unsigned char* pszResByte );

void Shutdown_M(void);

}
```

(OEM) インタフェース MMC ⇔ NCK/PLC

（注）
次の章は、OEM-MMC からそのまま転載しています。

概要

NCDDE サーバは、MMC と YS 840DI のインタフェースです。MMC はこのインタフェースを使用して NC 、PLC およびドライブのデータの全てにアクセスします。

アプリケーション製作者は NCDDE サーバから次の 3 つのサービスを受けることができます：

変数サービス	NC 、PLC およびドライブデータにアクセスする
ドメインサービス	MMC から NCK に、あるいは NCK から MMC にファイルをコピーする
PI サービス：	NC のプログラム呼び出しサービスを開始する

一般事項

アプリケーションと NCDDE サーバ間の通信は、WINDOWS DDE (Dynamic Data Exchange　動的データ交換) インタフェースによって実現されます。このインタフェースは、どの WINDOWS 開発環境でも使用できます。

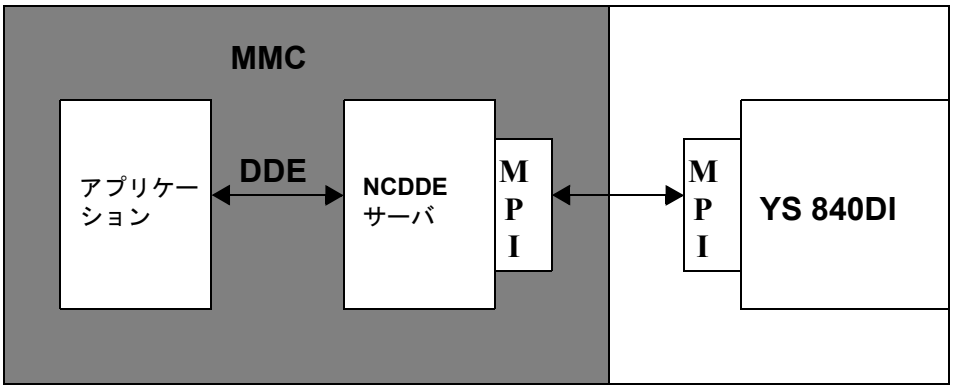


図 A.2 概要

ユーザーが NCDDE を特定の開発環境に合致するように適応させるための初期化ファイルを使用して、NCDDE サーバの環境設定を行うことができます。検査用に使用できる制御がありますか？データにアクセスできる NCU は 1 つですか、それとも多数ありますか？

（注）：WINDOWS 環境を使用するため、YS 840DI との通信には、「時間保証」はありません。したがって、実時間タスクは MMC では解決できないので、OEM パッケージ NCK を使用する NCU で直接解決してください。

DDE のベーシック

概要

WINDOWS オペレーティングシステムは、DDE をサポートしていますので、アプリケーション開発者は WINDOWS プロセスから別の WINDOWS プロセスへデータを転送することができます。

DDE の機能

DDE は、WINDOWS 環境下で行われる動的データ交換に次のような特徴を与えます：

- DDE は、WINDOWS アプリケーション同士の通信です。
- DDE は、クライアント・サーバモデルにしたがって 2 つのプロセスで実行されます。
- プロセスの 1 つはクライアントとして機能する：これは、サーバにデータをリクエストします。
- もう 1 つのプロセスは、サーバとして機能します：これは、クライアントにデータを提供します。
- 接続は、クライアントが設定します。
- プログラムは、クライアントだけでなくサーバとしても機能します。
- 通信は、内部の WINDOWS プロトコルに基づき条件が指定されます。

DDE 接続の設定

DDE サーバとの接続を設定するに当たって、クライアント開発者は次の事柄に精通している必要があります：

- リンクサーバ　DDE サーバの名前
- リンクトピック　テーマ
- リンク項目　アクセスするデータ項目
- リンクモード　接続のタイプ

DDE リンクモード

- リクエスト　クライアントは一度限りデータの問い合わせをする
- ウォームリンク　サーバはデータが変更されたことをクライアントに通知するので、クライアントはこのデータ項目にアクセスすることができる。
- ホットリンク　データがすでに変更されている場合、サーバは現在のデータ値を自動的にクライアントに送る。
- ポーク　クライアントはサーバにデータ項目があればそれを書き込むように指示する。
- 実行　クライアントサーバに指令があればそれを実行するように指示する。

（注）：DDE 接続についてさらに詳しくお知りになりたい場合は、
次の書籍を読むことをお薦めします：

マイクロソフト出版：Charles Petzold　Programming under Windows 3.1「Wndows 3.1 環境でのプログラミング」

マイクロソフト出版：Charles Petzold　Windows NT Programming「Wndows NT プログラミング」

NCDDE サーバの環境設定

初期化ファイル MMC.INI

説明

NCDDE サーバの初期化は、ファイル MMC.INI セクション [GLOBAL] を用いて行います。このファイルは OEM システムのディレクトリ \MMC2 にあります。そこで、Link Server 「リンクサーバ」および Link Topic「リンクトピック」のどちらとローカル NCDDE サーバの接続を設定すべきか決定されます。Link Server 「リンクサーバ」Link Topic「リンクトピック」という用語は、第 0 章に説明しています。

指令として設定するか、あるいは MMC.INI ファイルの 4 つの文字列をそれぞれ起動することによって、3 つの基本的な NCDDE サーバの環境設定を行うことができます：

- NC との接続を設定する（デフォルト）
- 1 つまたは複数の NC との接続を設定する（ M:N- 特長については、第 0 章を参照のこと）
- PC のローカルオペレーションモード
  開発者は、NC に接続せずに自分の PC でその場で自分の作成したアプリケーションを試験することができます。この場合、NCDDE サーバは代替値を提供しますが、これは指令 "NEW" (第 0 章) を使用して定義し、指令 "ANIMATE" (第 0 章) を使用して修正し、アクティブ NC をシミュレーションすることができます。

（注）：この 3 つの環境設定方法はいずれか 1 つしかアクティブになりません。

（注）：文字列の初めにつけられたセミコロン（ ; ）は、その文字列がコメントであることを示しています。

NcddeServiceName

NCDDE サーバの DDE-Link-Service 「DDE リンクサービス名」。デフォルト名は、"ncdde" です。

（注）：第 8 章の例は全て、"NcddeServiceName= ncdde" と仮定したものです。
別の名前がつけられている場合、この例を有効に機能させるために次のように修正してください。

Ncdde-MachineName

ここでは、標準アプリケーションの NCU 名を入力します。
すでに "MachineSwitch" が入力されている場合は、複数の NCU で切り替えをすることができます（ M:N 機能については 第 0 章を参照してください）。

NcddeDefault-MachineName

これにより M:N 機能を開始することができます。すなわち、MMC が始動されると NCU が接続されます。

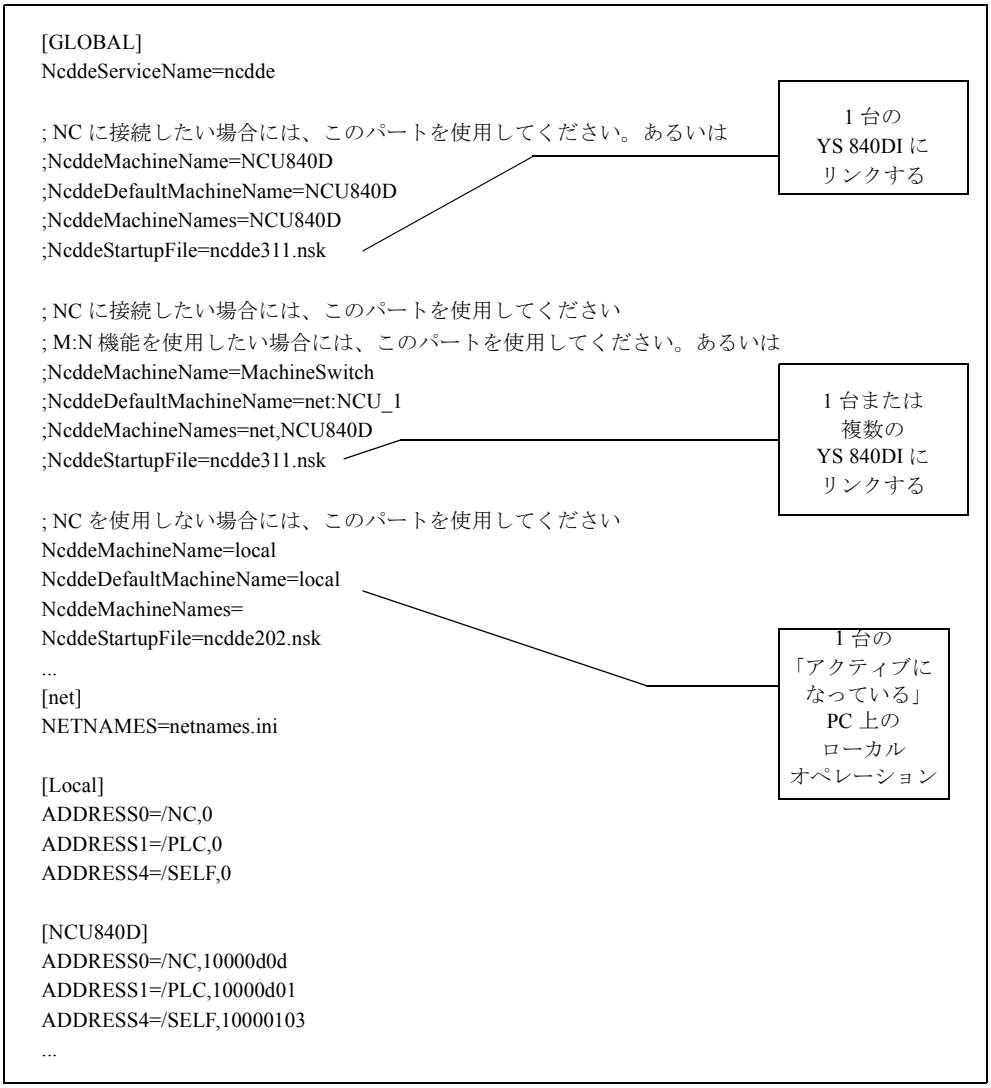
Ncdde-MachineNames

ここでは、接続することができる NCU の名前を入力します。個々に入力されるそれぞれの NCU 名に関しては、ファイル MMC.INI に同一の名前のセクションが存在していなければなりません。

NcddeStartupFile

NCDDE サーバを開始するとロードされる NSK ファイル (第 0 章)

例 A-1 ファイル MMC.INI からの抜粋



NCDDE サーバの指令ファイル

NSK ファイル

指令ファイル（拡張子 NSK を持つ）には、例えば NCDDE 接続が参照する Link-Items 「リンク項目」などが含まれています。これらのファイルに、第 0 章に述べた指令が含まれている場合もあります。

これらのファイルでは、アクセスすることができるデーター Link Items －が説明されています。

またさらに、CALL 命令を使用して NSK ファイルが包含されている場合もあります。それによって、構造化が可能になります。例 8-2 に、Link-Item (LastError) ならびに CALL 命令を使用する MMC 用のグローバル変数の構造化が示されています。

（注）：CALL 命令を用いて、自分で作成した NSK ファイルをロードすることもできます。NSK ファイルは MAP 機能を使って作成することができます (第 0 章)。

例 A-2 ディレクトリ \mmc2 内のファイル NCDDE311.NSK

```
REM NSK ROOT FOR 840D
REM ================================================
REM
REM WRITE-ACCESS FOR NC-BUSADDRESS
  LINK("/Nck/Nck/busAddress",200,"7 31 0 0 E0# /NC 1 0 11",10)
  LINK("/Nck/Nck/busState",300,"",0);
REM
REM ACCESS TO CONNECTION ERROR STATE
  LINK("LastError",1,"",0);
REM
REM IMPORT 840D BASIC NC VARIABLES
  CALL(nc311.nsk)
REM
REM IMPORT 840D BASIC PLC VARIABLES
  CALL(plc311.nsk)
REM
REM IMPORT ADDITIONAL LINK VARIABLES
  CALL(add311.nsk)
REM
REM IMPORT COMIC STARTS
  CALL(comic.nsk)
REM
```

複数の NC と接続する

M : N 機能

この機能を使うと、複数の MMC と複数の NCU を接続することができます。例えば、たった 1 台の MMC から 2 つの NCU に保存されているデータにアクセスすることができます。ファイル NETNAMES.INI はこの基本環境設定用に解釈されます。

パートの接続

セクション [conn MMC_1] は、MMC が接続されるパートナーを指定します。

ネットワークパラメータ

セクション [param network] では、転送速度が設定されますが、バスの環境設定によって異なります :

BTSS 1.5 Mbit

MPI 187.5 KBit

バスノード

セクション [param NCU_n] では、NC および PLC のバスアドレス、ならびに NCU 名が設定されます。MMC はこれらの名前を使用して NCU にアドレス指定します。NCU はそれぞれに関して記述を受け取ります。

例 A-3 ファイル NETNAMES.INI

```
; バスアドレスへのオーナーの TECHNICAL リファレンス
; コンピュータ別
[own]
owner=              MMC_1
; 可能性のある接続の説明
[conn MMC_1]
conn_1=             NCU_1
conn_2=             NCU_2

; 有効なネットパラメータの説明
; btss =1,5MBit
; mpi =187,5 KBit
[param network]
bus=                 btss

; すべてのバス参加者用のバスアドレス
[param MMC_1]
mmc_address= 1

[param NCU_1]
nck_address= 10
plc_address= 10
name=Standard_Machine

[param NCU_2]
nck_address= 11
plc_address= 11
name=Test_Machine
```

DDE 接続の設定

概要

この章では、Visual Basic および Visual C++ を使用して NCDDE サーバと DDE の接続を設定する方法について説明します。

（注）：以下の例では、DDE 通信には Standard Visual Basic Control (VBX) "LABEL" だけしか使用しません。しかし、OEM アプリケーションでは、DDE 通信に OEM Visual Basic 制御（例えば、当社の DCTL）を使用する必要があります。

以下の例がうまく機能するためには次の事項が満たされていなければなりません：

開発環境

- 推奨は、MS Visual Basic 3/4.0_16
- YS 840DI で直接 PC からこの例を試験するには、MPI 接続ならびに NC オペレーションに環境設定された NCDDE が必要です。
  YS 840DI を持たずに NCDDE サーバを使用する場合、データによってはアクセスできないものもあります。
- NCDDE サーバ (C:\MMC2\NCDDE.EXE) が始動している必要があります（例えば、ファイルマネージャ／エクスプローラを介して）。

Visual Basic を用いて DDE を設定する

DDE クライアント接続を設定することができる Standard Visual Basic Controls (VBX) の例、

- ラベル
- テキストボックス
- ピクチャ

Link - Service および Link - Topic は、Property (属性) "LinkTopic" に組み込まれています。これらは、縦棒“｜”によって区切られます
（例えば、LinkTopic = "ncdde ｜ local"）。

一度限り変数を読み出す

次の例はワーク座標系の第 1 チャンネルの第 1 軸の実際の位置を**一度**読み出します。下の例では、NCDDE サーバーをローカルオペレーション用に環境設定し、また NcddeServiceName を ncdde に設定する必要があります。すなわち、この場合、NCK にはアクセス**できません**。このタイプの読み出しでは、LinkMode は 2 にセットしておく必要があります。

（注）：値を一度だけ読み出すには、LinkMode を 2 に設定してください。第 1 チャンネルは、Link Request を用いて値をリクエストします。

例 A-4 一度限り変数を読み出す

```
Sub Form_Load ()
    Label1.LinkTopic = "ncdde|local"
    Label1.LinkItem="/Channel/GeometricAxis/actToolBasePos[u1,1]"
    Label1.LinkMode = 2
    Label1.LinkRequest
End Sub
```

（注）：チャンネル識別子 "u1" が指定されていない場合、自動的に第 1 チャンネルにアクセスします。

変更時に更新する

次の例は、"label1" の機械座標系の第 2 チャンネルの第 3 軸の実際の位置を**自動的に**更新します（ホットリンク）。すなわち、現在の実際の位置が表示されます。

（注）: ホットリンクでは "LinkMode" は必ず 1 に設定してください。

例 A-5 変更時に更新する

```
Sub Form_Load ()
    Label1.LinkTopic = "ncdde|ncu840d"
    Label1.LinkItem="/Channel/MachineAxis/actToolBasePos[u2,3]"
    Label1.LinkMode = 1 'Hotlink
End Sub
```

変更時に通知する

この例では、最初の PLC バイトが変更された場合（Warmlink）、NCDDE サーバはアプリケーション／クライアントを通知します。その後、Label1 の "Sub LinkNotify" が自動的に実行されます。その場合、データを得るために "LinkRequest" をコールする必要があります。したがって、表示される前に、データをチェックしたり、修正や変更を加えることができます。

（注）: 変更時に通知するには (Warmlink) 、" LinkMode" を 3 に設定してください。

例 A-6 変更時に通知する

```
Sub Form_Load ()
    Label1.LinkTopic = "ncdde|ncu840D"
    Label1.LinkItem = "/PLC/Input/Byte[1]"
    Label1.LinkMode = 3
End Sub

Sub Label1_LinkNotify ()
        Label1.LinkRequest
End Sub
```

NC データを書き込む

この例では、クライアントは第 1 チャンネルの第 1 の R- パラメータ R[1] に値「4」を**書き込みます**。

（注）データ (Poke) を書き込むためには、"LinkMode" を 2 に設定してください。LinkPoke がこの値を書き込みます。

例 A-7 NC データを書き込む

```
Sub Form_Load ()
    Label1.LinkTopic = "ncdde|ncu840d"
    Label1.LinkItem = "/Channel/Parameter/R[1]"
    Label1.LinkMode = 2 ’ 手動
    Label1.Caption = "4"
    Label1.LinkPoke
End Sub
```

PLC データを書き込む

この例では、クライアントは PLC のフラグバイト 5 に値 "250" を書き込みます。

例 A-8 PLC データを書き込む

```
Sub Form_Load ()
    Label1.LinkTopic = "ncdde|ncu840d"
    Label1.LinkItem = "/PLC/Memory/Byte[5]"
    Label1.LinkMode = 2 ’ 手動
    Label1.Caption = "250"
    Label1.LinkPoke
End Sub
```

指令を実行する

実行できる指令は、第 0 章に記載されています。

次の例は、ファイル "test.mpf" の MMC から NCK への転送を開始します。

（注）: 指令（Execute）を実行するには、"LinkMode" を 2 に設定してください。
LinkExecute が指令を実行します。

例 A-9 指令を実行する

```
Sub Form_Load ()
    Label1.LinkTopic = "ncdde|ncu840d"
    Label1.LinkMode = 2
    Label1.LinkExecute "COPY_TO_NC(""C:\NC\test.mpf"",
                        /NC/_N_MPF_DIR/N_TEST_MPF, trans)"
End Sub
```

Visual C/C++ を使用する DDE 接続の設定

概要

C/C++ は DDE インタフェースの全ての機能を利用することができます。特に、DDE インタフェースに非同期コールを使用することができます。DCTL などの OEM Visual Basic Controls を使用している場合、Visual Basic でも使用できます。

（注）: Wndows 環境下で C プログラミングに精通していて、かつ OEM パッケージのシーケンス制御のインテグレーションのほんの**部分的**にしか必要としない、あるいは**全く**必要としない OEM ユーザーーには、C/C++ を使用する DDE を推奨します。

C / C++ を使用する DDE アクセス

この例では、以下のプログラム間での Hotlink (Advise) 接続（確認応答）を設定する方法を示します

- C/C++ Program　C/C++ プログラム
- Variable "/Channel/GeometricAxis/toolBaseDistToGo[1]"
- NcddeServiceName = ncdde
- NcddeMachineName = local

変数値の変更は、DDEML にリポートされたコールバックルーチンによって XTYP_ADVDATA トランザクションで受信されます。

例 A-10 C レベルの Hotlink

```
DWORD    idInst;   // DdeInitialize を使用して生成される
HSZ hsz     Service, hszTopic, hszItem;     // ストリングが処理する
HCONV     hConv;   // 会話が処理する

hszService  = DdeCreateStringHandle ( idInst ,          "ncdde" ,   ZERO );
hszTopic    = DdeCreateStringHandle ( idInst ,          "local" ,    ZERO );
hszItem      = DdeCreateStringHandle ( idInst ,
               "/Channel/GeometricAxis/toolBaseDistToGo[1]" , ZERO );

hConv       = DdeConnect(idInst,hszService,hszTopic,ZERO);
                                        // サーバへの接続
                                        // Hotlink が追従する
if ( DdeClientTransaction ( (LPBYTE)ZERO , 0 , hConv , hszItem ,
            CF_TEXT ,XTYP_ADVSTART|XTYP_ACKREQ , 1000 , ZERO )
            = =TRUE) { }                // Hotlink 接続がつながる
```

（注）: the C/C++ DDE プログラミングをもっと知りたい人には、次の書籍を推薦します：
マイクロソフト出版：Charles Petzold, Programming under Windows 3.1 「Windows 3.1 環境でのプログラミング」
マイクロソフト出版：Charles Petzold, Windows NT Programming「Windows NT プログラミング」

MS Excel から DDE 接続を設定する

概要

Excel 環境で、セル公式を用いて変数用の NCDDE サーバインタフェースと Advise (Hotlink) 接続を構築することができます。

EXCEL セルのシンタックス

セル内に = NcddeServiceName|NcddeMachineName!Variablen を定義することができます。

Excel に PLC ビットを表示する

次の例では、Excel （ドイツ版）のセルとデータブロック 100 のバイト 9 の第 3 ビットとの Advise (Hotlink)- 接続が示されています。

この変数名は："/Plc/DataBlock/Bit[c100,9.3]" です。NCDEE サーバによって接続されたマシンの名前は "ncu840D" です。

例 A-11 MS Excel に PLC ビットを表示する

	A
1	=ncdde|ncu840D!'/Plc/DataBlock/Bit[c100,9.3]'

	A
1	1

左側にセル公式が表示され、右側にその結果が順次更新されて表示されます。

変数サービス

概要

NCDDE サーバの変数サービスによって、2 種類のデータアクセスを行うことができます :

- シングル変数アクセス
- アレイ変数アクセス

追加のデータフォーマットおよび必要であればアレイレンジを使用して、Link Item に変数を定義することができます。これにより、ほとんどの場合さらに変換を必要としない形で NCDDE にデータをリクエストすることができます。

（注）: 第 11 章 Reference 「リファレンス」および変数オンラインヘルプに、アクセスすることができる変数の全記述があります。

NCDDE 変数のフォーマット

NCDDE 変数用命令のフォーマットは、Link Item の最後に付加されています。データの内部準備により、タイプを整数、フローティング数、およびテキストにフォーマットすることができます。

このフォーマットは、C プログラミング言語の ’printf’ 拡張フォーマットに指定されます。NCDDE フォーマット命令のシンタックスには、次のようなものがあります :

フォーマット :	""	<Params>	<’printf’-Format>
引数 :	’!’ ’b’	<Params>	// ビットストリングに変換する
	’!’ ’d’	<Params>	// d for double as 64 ビットフローティング
	’!’ ’l’	<Params>	// l for long as 32 bit integer
	’!’ ’t’	<Params>	// t for text as string
	’!’ ’#’	<Params>	// # , 変数のインデックス // 32 ビット整数としてアクセス

対応する DDE 変数のデータタイプは、第 11 章または "NCDDE variable help"「NCDDE 変数ヘルプ」にあります。

（注）: データ選択および実際に読み出された変数のタイプが整合しない場合、データフォーマットの自動変換は実行されません。すなわち、誤ったデータが表示されます。

数値のフォーマット

ここでは、**第 2** 軸の実際の位置は NC から読み出されて、最大 11 桁まで、そして小数点以下最大 3 桁で表示されます。

フォーマットをしない場合、小数点以下しか得られません。

例 A-12 最大 11 桁と常に小数点以下 3 桁をフォーマットする

```
Sub Form_Load ()
    Label1.LinkTopic = "ncdde|ncu840d"
    Label1.LinkItem = "/Channel/MachineAxis/actToolBasePos[2]
                                                        (""!d%11.3lf"")"
    Label1.LinkMode = 2 'Manual   手動
    Label1.LinkRequest
End Sub
```

十六進法ファーマットに変換する

この例では、PLC からフラグバイト 5 が読み出されて、先行ゼロをつけて表示されています。

例 A-13 十六進法フォーマットに変換する

```
Sub Form_Load ()
    Label1.LinkTopic  = "ncdde|ncu840d"
    Label1.LinkItem   = "/PLC/Memory/Byte[5] (""!l%02lx"")"
    Label1.LinkMode  = 2 'Manual   手動
    Label1.LinkRequest
End Sub
```

ビットストリングへ変換する

この例では、フラグバイト 5 が読み出され、32 ビットストリングとして表示されています。

例 A-14 ビットストリングへ変換する

```
Sub Form_Load ()
    Label1.LinkTopic = "ncdde|ncu840d"
    Label1.LinkItem = "/PLC/Memory/Word[5] (""!b%16.16s"")"
    Label1.LinkMode = 2 ' マニュアル
    Label1.LinkRequest
End Sub
```

結果 : 10101010101010101

PLC からストリングを読み出す

この例では、バイト 20 から開始するデータモジュール 81 から 10 バイトが読み出され、終端ゼロを使用するストリングとして表示されています。

例 A-15 PLC からストリングを読み出す

```
Sub Form_Load ()
   Label1.LinkTopic = "ncdde|ncu840d"
    Label1.LinkItem = "/PLC/DataBlock/Byte[c81,20,#10] (""!l%lc"")"
    Label1.LinkMode = 2 ' マニュアル
    Label1.LinkRequest
End Sub
```

結果 : Hello

シングルアクセス

Sequence Control (参照、第 7 章) 内で作業している場合、LinkTopic としてグローバル変数 "g_chNCDDEServiceName" を使用するのが都合が良いでしょう。NCDDEServiceName と NcddeMachineName は、ファイル MMC.INI に入力されているので、必ずこの変数に含まれています。これらの変数は縦棒記号 ("|") によって分けられます。

3 つの変数へのシングルアクセス

最初の 3 つのジオメトリ軸名を読み出します。

例 A-16 3 つの変数へのシングルアクセス

```
Sub Form_Load
     achsname(0).LinkTopic = g_chNCDDEServiceName
     achsname(0).LinkItem = "/Channel/MachineAxis/name[1]"
    achsname(0).LinkMode = 2
     achsname(0).LinkRequest
     achsname(1).LinkTopic = g_chNCDDEServiceName
     achsname(1).LinkItem = "/Channel/MachineAxis/name[2]"
     achsname(1).LinkMode = 2
     achsname(1).LinkRequest
     achsname(2).LinkTopic = g_chNCDDEServiceName
     achsname(2).LinkItem = "/Channel/MachineAxis/name[3]"
     achsname(2).LinkMode = 2
     achsname(2).LinkRequest
End Sub
```

PLC- ビットアクセス

次の Link Item を使用して、入力バイト 2 のビット 4 にアクセスすることができます。

/Plc/Input/Bit[2.4]

PLC- バイトアクセス

次の Link Item を使用して、出力バイト 4 にアクセスすることができます。

/Plc/Output/Byte[4]

PLC- ワードアクセス

次の Link Item を使用して、レジスタワード 4 にアクセスすることができます。

/Plc/Memory/Word[8]

その他の変数へのアクセス方法は、第 11.1.5 章に記載しています。

アレイ変数アクセス

アプリケーション

複数のデータを　1 つのエリアから読み出す場合、アレイアクセスを使用するのが良いでしょう。そうすると、単一の変数への複数アクセスと比べて、NCDDE サーバのコンピューティング負荷を効率的に軽減することができます。例 8-16 に、この良くない例をあげます。

（注）: アレイアクセスを使用すると、通信時間を著しく減少できるので、データアクセスだけでなくコンプリートシステムの速度もスピードアップされます。

シンタックス

ここでは、アレイエリアのシンタックスについては、簡単にふれるだけにします。

Variablenname[c, u, StartIndex, [EndIndex]]

パラメータ

表 A.1 アレイデータアクセスのパラメータ

名前	説明
変数名	NCK/PLC 変数の名前 (参照、第 11 章)
c	NCK 変数にアクセスする場合 (参照、第 11 章): **カラムインデックス** c は、コラムを示し、多次元アレイだけに適用される PLC 変数にアクセスする場合、変数 c はアクセスされる**データモデュール**の特徴を決定する。
u	NCK 変数に**のみ**使用されるユニットインデックス (例、チャンネル) : u はユニットを示す
StartIndex	読み出される変数のインデックス アレイにアクセスする際に、読み出しの最初の値が与えられる
EndIndex **(オプション)**	**アレイアクセス**のみ 読み出される値の数を指定する

軸名アレイにアクセスする

次の例では、NCK から最初の 3 つの軸名が読み出されています。結果は、フォーマット "X1Y1Z1" のこれらの軸名、例えば X1,Y1,Z1 を含むストリングです。Visual Basic 機能 "Trim$" や "Mid$" を使用すると、この結果ストリングから軸名を抽出することができます。

例 A-17 軸名アレイにアクセスする

```
m_a_namen.LinkTopic = g_chNCDDEServiceName
m_a_namen.LinkItem = "/Channel/MachineAxis/name[u1,1,3]"
m_a_namen.LinkMode = 2
m_a_namen.LinkRequest

テキストアレイからシングル名を抽出する

achsname1.Caption = Trim$(Mid$(m_a_namen.Caption,1,2))
achsname2.Caption = Trim$(Mid$(m_a_namen.Caption,4,2))
achsname3.Caption = Trim$(Mid$(m_a_namen.Caption,7,2))
```

軸名アレイにアクセスする

次の例では、2 番目のチャンネルの 2 つの軸の軸名が読み出されて、軸 3 から開始されます。軸 3 および 4 の名前が読み出されます。

次の文字列を除いて、例 8-16 と同じになります。

例 A-18 軸名アレイにアクセスする

```
...
LinkItem = "/channel/machineaxis/name[u2,3,4]"
...
```

PLC アレイデータにアクセスする

次の例では、PKC から十六進フォーマットの 2 桁の数としてバイト 2 からバイト 4 の DB 8 の 3 つのバイトが読み出されます。そして、これらのバイトは、Visual Basic 機能 "Trim$" および " Mid$" を用いて抽出されます。

例 A-19 PLC アレイデータにアクセスする

```
abel1.LinkTopic = "ncdde|ncu840d"
Label1.LinkItem = "/PLC/Datablock/Byte[c8,2,4](""!l%02lx"")"
Label1.LinkMode = 1 ’ ホットリンク

十六進フォーマットのシングルバイトを抽出する

byte_1 = Trim$(Mid$(Label1.Caption,1,2))
byte_2 = Trim$(Mid$(Label1.Caption,3,2))
byte_3 = Trim$(Mid$(Label1.Caption,5,2))
```

PLC が数を指定するにアクセスする

次の例では、PLC から 4 桁の十六進法数としてワード 2 から始まる DB 8 の 5 つのワードを読み出します。ワードは、"_" を使って区切られます。

例 A-20 PLC アレイデータの指定数を読み出す

```
Label1.LinkItem = "/PLC/Datablock/Word[c8,2,#5](""!l%04lx_"")"
```

R パラメータアレイにアクセスする

この例では、3 つの R パラメータ R3, R4 および R5 の値を読み出します：
R3 = 2.2
R4 = 3.5
R5 = 4.9.

例 A-20a R パラメータアレイにアクセスする

```
Label1.LinkTopic = "ncdde|ncu840d"
Label1.LinkItem = "/CHANNEL/PARAMETER/R[U1,3,5]"
Label1.LinkMode = 2 ’Manual　手動
Label1.Caption = ":2.2:3.5:4.9"
Label1.LinkPoke
```

ファイル転送サービス (ドメインサービス))

概要

ドメインサービスを使用して、エリア（ドメイン）MMC と NCK/PLC の間でファイル転送を行うことができます。

MMC と NCK/PLC の間のファイル転送用に利用できる指令は全部で 5 つあります (表 A-1)。このファイル転送はバックグラウンドで実行されます。

エリア同士の拡張コピー機能を使用することができます。これらの機能は、特に NC でのプログラム編集に適しています。新規機能については、第 0 章に詳しい説明があります。

表 A.2 ドメインサービスの指令

指令	説明
COPY_FROM_NC	NCK から MMC へ転送する
COPY_FROM_NC_BINARY	NCK (PLC) から MMC へ転送する
COPY_TO_NC	MMC から NCK へ転送する
COPY_TO_NC_BINARY	MMC から NCK (PLC) へ転送する
MAP_ACC_NC	NCK から ACC ファイルをロードし、DDE インタフェースで使用できるようにする

データ転送のステータスは、特別ステータス変数を用いてモニタすることができます。

MMC と NCK/PLC でデータ転送をする

説明

これらの機能を使用すると、MMC と NCK/PLC の間でデータ／データファイルを転送することができます。

アプリケーション

パートプログラムやツールデータを NCK に転送する、あるいは S7 や C プログラムを PLC に転送する場合、これらの機能は最適です。

拡張子 "BINARY" を**持たない**機能は、例えばパートプログラムのようなファイルを NC に転送することができます。NCDDE サーバは、データにブロックヘッダを追加します。このヘッダには、NCK ファイルシステムのブロックおよびパスのサイズやデータが含まれています。

（注）: NCK へデータを転送するのに使用する

（注）: データストリームには必ず NC ブロックヘッダが追加されているので、NCK へファイルを転送する場合には使用できません。

BINARY 機能

拡張子 "BINARY" を**持つ**機能は、パートプログラムなどのファイルを NC に転送したり、また PLC モジュールを PLC へ転送することもできます。NCDDE サーバは、データにブロックヘッダを追加することなく、これらのファイルを転送します。

（注）: この機能は、PLC および NCK へファイルを転送するのに使用できます。

（注）: PLC モジュールは必ず、PLC のパッシブファイルシステムにコピーされます。その時点では、まだアクティブになっていません。パッシブモジュールを起動する必要があります (例 0-33 と比較してください)。

シンタックス

コピー機能は、シンタックスに追従するストリングとして書き込んでください：

COPY_FROM_NC (WinFile,NcFile,TransferState)
COPY_TO_NC (WinFile,NcFile,TransferState)

COPY_FROM_NC_BINARY (WinFile,NcFile,TransferState)
COPY_TO_NC_BINARY (WinFile,NcFile,TransferState)

パラメータ

表 A.3 指令のパラメータ

COPY_TO/FROM_NC(_BINARY)

名前	説明
WinFile	MMC エリアの情報のソースとデスティネーション
NcFile	NCK/PLC 環境用のファイル名
TransferState	転送ステータスの特徴を決める変数

パラメータ WinFile

このパラメータは、MMC サイドの情報のソースとデスティネーションを記述します。

最初の文字がタイプを指定します。

このパラメータは、Wndows 環境のデフォルトファイル名です。ドライブ仕様、パス名、およびファイル名（例、"C:\NC\test.MPF"）を含んでいます。

パラメータ WinFile を使用するパイプライン処理

WinFile の最初の文字が、@-character （アットマーク）の場合、パラメータはパイプ名として翻訳されます。機能 COPY_TO_NC と組合わせて、"Copy via pipes" サービスを実行することができます。

（注）: 最大 500 バイトまでの大きさのブロックに読み出したり書き込むのに適しています。それより大きなブロックは NCDDE サーバにより拒否されます。

NCK/PLC に転送 (ダウンロード) 時に、DDE ポークがパイプラインに書き込みをするので、NCK/PLC に直接転送することができます。空のポークは、転送の終了を意味します（参照例 0-25）。

NCK/PLC に転送 (アップロード) 時に、DDE はパイプラインが空になっているようリクエストします。転送の実行時にはパイプラインは空ではありません。リクエストが空のデータを呼び戻した場合、転送の終了を意味します。

パラメータ WinFile を使用する Shared Memory アクセス

WinFile の最初の文字が、# 記号 で、それに十六進フォーマットが続く場合、Global Heap に割当てられた WINDOWS Shared Memory として翻訳されます。Wndows 機能 GlobalAlloc を使用して割当てたメモリは、次の構造を使用して初期化する必要があります。このヘッダに後続して使用可能なデータが続きます。次の例は、Visual Basic で使用する場合です。

```
struct NCDDE_DOMAINMAP_HEADER {
  unsigned short handle;          // バッファ処理 (HGLOBAL) ( クライアントが事前設定する )
  unsigned short header_size;     // ヘッダの長さ ( クライアントが事前設定する )

  unsigned long shared_size;      // データエリアの利用可能な長さ
                                  // ( クライアントが事前設定する )

  unsigned long fill_count;       // データエリアの有効バイト数
                                  // ( ダウンロード時にクライアントが事前設定する
                                  // そしてアップロード時にサーバにより設定される )

  unsigned long state;            // 転送ステータス変数に対応する
                                  // 転送指令の
                                  // < 100: 転送が実行中 ,
                                  // "state" すでに転送された
                                  // ファイルの割合をほぼ反映する
                                  // ==100: 転送が問題なく完了した
                                  // > 100: 転送がエラーにより中断された ,
                                  // "state"  ncdde エラーコードを示す
                                  // ( サーバにより設定される )

  unsigned long file_mod_time; // ファイル修正時間
                                     value:0 現在の時間を示す
                                  // ( ダウンロード時にクライアントが事前設定する
                                  //) そしてアップロード時にサーバにより設定される

  unsigned long server_private; // サーバ別データ ( サーバにより設定される )

  unsigned long client_private; // クライアント別データ ( クライアントが設定する )

  unsigned long magic;           // 追加のタイプチェック用の署名
                                 // 値は常に NCDDE_MAGIC = 0xF6F7F8F9
                                 // ( クライアントが事前設定する )
};
```

パラメータ NcFile

パラメータ "NcFile" は、NCK/PLC 環境のパス名です。この名前は、環境設定が可能なパートに構築されます。このパラメータは影響を受ける NCK にアドレス指定したり、さらに NCK 環境のドメインパスのアドレス指定するのに必要です。

NCK のドメインは、NC ファイル名を使用して NCDDE サーバを介してアドレス指定されます。

/NC エリア :　　PLC または NCK

/_N_MPF_DIR　　NC 用のパス仕様

/_N_WS03_MPF　　ファイル名

パラメータ TransferState

パラメータ TransferState は、バックグラウンドで実行される転送のステータスをリターンするのに使用されるサーバ（変数タイプ：一定）のローカル変数名です。この変数が指定されていない場合、サーバから作成されます。

変数 TransferState により、ファイル転送のステータスの特徴が決定されます :

表 A.4 転送ステータスの特徴を決定する

転送ステータス	値	意味
転送が開始	0	CNC のオープニングプロトコルが処理中である (ファイルを開く)
転送が実行中	1 bis 98	転送が実行中である。この数が転送済みのファイルの割合をほぼリポートする。(注記を参照のこと).
転送が終了	99	CNC のクロージングプロトコルが処理中である (ファイルを閉じる)
転送が完了	100	エラーなしにジョブが実行された
転送がエラーにより中断	>100	転送が中止される TransferState c にはリポート済みのエラーコードが含まれる (11.7 章).

値の範囲が選択された結果、値 <=100 は正常な状態を示しています。この場合、他の値は全てエラーがある状態を示します。

（注）: 変数が 1 から 99 の範囲にある場合、その変数はファイル転送に使用することはできません。

ファイル転送を中止する

ファイル転送を中止するには、転送変数を該当するエラーコードを用いて上書きする必要があります。すなわち、"LONG" （4 バイト）に定義されている転送バイトの変数はそれぞれ 0 **以外の値**を持っています。
該当するエラーコードの例 : 16909060

ビジュアライゼーション（可視化）

転送ステータスのビジュアライゼーションには、変数 TransferState を、例えば Advise/Hotlink 接続を介してバー表示で使用することができます。

（注）: バイナリモードでの転送や、パイプを使用する転送では、ファイルの大きさについての情報は利用できません。したがって、TransferState には現在転送済みデータの割合を含めることはできません：常に 50% となります。

非常に短いファイルでは、表示が 1 から 99 に飛ぶ場合があります。これは Hotlink の原理の問題で、クライアント／アプリケーションが NCDDE サーバからデータを間に合う速度で呼び出すことができないことが原因です。

パートプログラムをアップロードする

次の例は、パートプログラム "BSP.MPF" をディレクトリ "C:\NC" のファイル "test.mpf" にコピーします。このファイル "test.mpf" は新規に作成されます。パートプログラム "BSP.MPF" は、NCK に入っていなければなりません。

例 A-21 パートプログラムをアップロードする

```
Sub Form_Load ()
    Label1.LinkTopic = "NCDDE|ncu840d"
    Label1.LinkMode = 2
    Label1.LinkExecute "COPY_FROM_NC (C:\nc\test.mpf,
                        /NC/_N_MPF_DIR/_N_BSP_MPF,trans)"
End Sub
```

Shared Memory 「共有メモリ」アクセス

次の例 (例 8-22 と例 8-23) はパートプログラム "TEST.MPF" を共有メモリエリアにコピーし、ここからタイプ "String" の Visual Basic 変数 FILE$ へコピーします。機能 Memoryread を使用して VB からこれらのデータにアクセスすることができますが、書き込みしかできません。C++ を使用すればこの機能をさらに効果的に利用することができます。

（注）: Visual Basic からビジュアライゼーションする場合、全ての要素をインテルフォーマットの NCDDE_DOMAINMAP_HEADER 構造に、すなわち予約されているハイバイトおよびローバイトを使用して、書き込まなければなりません。

例 A-22 共有メモリ　パート 1: "general declarations"

```
Windows SDK functions
Declare Function GlobalAlloc Lib "Kernel" (ByVal wFlags As Integer, ByVal dwBytes As
Long) As Integer
Declare Function GlobalFree Lib "Kernel" (ByVal hMem As Integer) As Integer
Declare Function GlobalLock Lib "Kernel" (ByVal hMem As Integer) As Long
Declare Function GlobalUnlock Lib "Kernel" (ByVal hMem As Integer) As Integer

Declare Function GlobalHandleToSel Lib "toolhelp.dll" (ByVal hMem%) As Integer

Declare Function MemoryWrite Lib "toolhelp.dll" (ByVal wSel%, ByVal dwOffset&, lpvBuf
As Any, ByVal dwcb&) As Long
Declare Function MemoryRead Lib "toolhelp.dll" (ByVal wSel%, ByVal dwOffset&, lpvBuf As
Any, ByVal dwcb&) As Long

Const HeaderSize = 32
```

例 A-23 共有メモリ　パート 2

```
Sub Form_Load ()
Dim ncdde_global_memory$
Static byte (1 To HeaderSize) As Integer
Dim ptrToBuffer, ergPtr As Long
        LenOfFile$ = Space$(5)
        'Hotlink, so the NC connection is established when Copy_FROM_NC is called
        Label2.LinkTopic = "NCDDE|NCU840D"
        Label2.LinkItem = "/Bag/State/resetActive"
        Label2.LinkMode = 1
        ' allokieren des Shared Memory
        handleglobal = GlobalAlloc(&H2102, 1024)
        ptrToBuffer = GlobalLock(handleglobal)
        'build up NCDDE_DOMAINMAP_HEADER
        'handle
        byte(1) = handleglobal Mod 256
        byte(2) = handleglobal \ 256
        ' header length
        byte(3) = HeaderSize
        byte(4) = 0
        ' Number of usable data area
        byte(5) = &HE0
        byte(6) = &H3
        byte(7) = 0
        byte(8) = 0

       ' Initialize the next 20 bytes with 0
        for i=9 to 28
            byte(i) = 0
        next i

        ' Initialize the NCDDE-Magic also
        byte(29) = &HF9
        byte(30) = &HF8
        byte(31) = &HF7
        byte(32) = &HF6
        ' Build up string
        For i = 1 To HeaderSize
            ncdde_global_memory$ = ncdde_global_memory$ + Chr$(byte(i))
        Next i
  ' Initialize Shared Memory area
  nBytes& = MemoryWrite(GlobalHandleToSel(handleglobal), 0&, ByVal
                                ncdde_global_memory$, Len(ncdde_global_memory$))
  Label1.LinkTopic = "NCDDE|NCU840D"
  Label1.LinkMode = 2 'Copy_From NC
  execCommand = "COPY_FROM_NC(#" + Hex$(byte(2)) + Hex$(byte(1)) +
                        ",/NC/_N_MPF_DIR/_N_TEST_MPF,trans)"
  Label1.LinkExecute execCommand
  ' Read data area from Shared Memory
  nBytes& = MemoryRead(GlobalHandleToSel(handleglobal),8, ByVal LenOfFile$, 4)
  ' Read length of usable data from Shared Memory
  nDataLen& = (Asc(Mid$(LenOfFile$, 2, 1)) * 256) + Asc(Mid$(LenOfFile$, 1,1))
  File$ = Space$(nDataLen&)
 ' Read usable data from Shared Memory
  nBytes& = MemoryRead(GlobalHandleToSel(handleglobal), HeaderSize, ByVal
                                File$, nDataLen&)
  erg = GlobalUnlock(handleglobal)
  erg = GlobalFree(handleglobal)
End
```

パートプログラムをダウンロードする

次の例では、"test.mpf" という名前のファイルをディレクトリ "C:\NC" から NC ディレクトリ "_N_MPF_DIR" へコピーします。NCK では、パートプログラムの名前は "BSP.MPF" です。

例 A-24 パートプログラムをダウンロードする

```
Sub Form_Load ()
    Label1.LinkTopic = "NCDDE|ncu840d"
    Label1.LinkMode = 2
    Label1.LinkExecute "COPY_TO_NC(C:\NC\test.MPF,
            /NC/_N_MPF_DIR/_N_BSP_MPF,trans)"
End Sub
```

パイプライン処理を使用してパートプログラムをダウンロードする

次の例では、パイプライン構造のメカニズムを利用したアプリケーションを示しています。ファイル PIPE1.MPF が NC で作成され、そこに NC ブロック "G01 F11111 X5555" が書き込まれます。

例 A-25 パイプライン処理を使用してパートプログラムをダウンロードする

```
Sub Form_Load ()
    ’Pipe starten
    Label1.LinkTopic = "NCDDE|ncu840d"
    Label1.LinkMode = 2
    Label1.LinkExecute "COPY_TO_NC(@pipe,
                        /NC/_N_MPF_DIR/_N_PIPE1_MPF,trans)"
    ’Pipe beschreiben
    Label2.LinkTopic = "NCDDE|NCU840D"
    Label2.LinkMode = 2
    Label2.LinkItem = "@pipe"
    Label2.Caption = "G01 F11111 X5555"
    Label2.LinkPoke
    ’beenden der Pipe
    Label2.Caption = ""
    Label2.LinkPoke
End Sub
```

S7 モジュールを PLC にダウンロードする

モジュール "OB1.PLC" を PLC のパッシブファイルシステムに転送する。

（注）: PLC モジュールは必ず、PLC のパッシブファイルシステムにコピーしてくだい。この時点では、モジュールはまだアクティブになっていません。パッシブモジュールをアクティブにする必要があります (例 0-33 を参照してください)。

例 A-26 S7 モジュールを PLC にダウンロードする

```
Label1.LinkItem = "ncdde|ncu840d"
Label1.LinkMode = 2
Label1.LinkExecute "COPY_TO_NC_BINARY(C:\TMP\OB1.PLC,
                        /PLC/_0800001P, trans)"
```

MMC と NC/PLC 間のコピー機能拡張

説明

これらの機能を使用すると、NCK/PLC と MMC 間でファイルの転送をやりとりすることができます。

アプリケーション

これらの機能は特に、シングルブロック、パートプルグラムのセクションの転送、あるいは NC に保存されているパートプログラムの編集に適しています。

（注）注機能の標準変数と「バイナリ」変数の違いについては、第 0 章に説明があります。

シンタックス

拡張されたコピー機能は、以下のシンタックスに追従するストリングとして書き込まれます。

COPY_FROM_NC
(WinFile,NcFile,seekPos,seekLen,compareString,skipCount)
COPY_FROM_NC(_BINARY)
(WinFile,NcFile,seekPos,seekLen,compareString,skipCount)
COPY_TO_NC(WinFile,NcFile,seekPos,seekLen,compareString,skipCount)
COPY_TO_NC(_BINARY)
(WinFile,NcFile,seekPos,seekLen,compareString,skipCount)

パラメータ

パラメータを表にしました。

表 A.5 指令 COPY_TO/FROM_NC のパラメータ

名前	説明
WinFile	MMC エリアの情報のソースとデスティネーション :
NcFile	NCK/PLC 環境でのファイル名
SeekPos	シークポインタ : コピー手順の開始点 ブロックの識別子は B 、文字の識別子は C
SeekLen	ウィンドウの大きさ : 転送されるエリア ブロックの識別子は B 、文字の識別子は C
CompareString	サーチストリング、最大長 32 文字
SkipCount	スキップが判明したサーチストリングの数

これらの指令は、そのサブ指令が全て完全に処理されるとリターンします。指令の実行中にエラーが発生した場合、変数 LastError を使用して分析することができます。

次の例では、新規指令の典型的なアプリケーションをいくつか示します。

プログラムパートのファイル転送

パートプログラム "TP1.MPF" の最初の 1024 バイトのディレクトリ "C:\NC" にあるファイル "test.dat" へのファイル転送

例 A-27 プログラムパートのファイル転送

```
Sub Form_Load ()
  Label1.LinkTopic = "NCDDE|NCU840D"
  Label1.LinkMode = 2
  Label1.LinkExecute"COPY_FROM_NC(C:\NC\test.dat,
                                  /NC/_N_MPF_DIR/_N_TP1_MPF,1 ,1024 , , 0 )"
End Sub
```

シングルブロックを転送する

ブロック 2 から 4 のパートプログラム X.MPF へのパイプ転送。既存のブロックは上書きされます。

例 A-28 シングルブロックを転送する

```
Sub Form_Load ()
    Label1.LinkTopic = "NCDDE|NCU840D"
    Label1.LinkMode = 2
    Label1.LinkExecute " COPY_TO_NC_BINARY ( @xpipe ,
                              /NC/_N_MPF_DIR/_N_X_MPF , B2 , 3 , , 0 )"
End Sub
```

ブロックを 1 つ転送する

パートプログラム TEST.MPF の第 2 ブロックへのテキスト転送（最大テキスト長： 200 バイト）。第 2 ブロックは上書きされます。

例 A-29　　ブロックを 1 つ転送する

```
Sub Form_Load ()
    Label1.LinkTopic = "NCDDE|NCU840D"
    Label1.LinkMode = 2
    Label1.LinkExecute "COPY_TO_NC ( ""!This is the content of the 2 nd block "" ,
    / NC/_N_MPF_DIR/_N_TEST_MPF, B2 , 1 , , 0 )"
End Sub
```

ドメイン間の MAP 機能

MAP_ACC_NC

説明

この機能を使用すると、グローバルユーザーデータ （GUD）と NCK マシンデータを NCDDE サーバに公示することができます。これらのデータは、NCK に置かれた拡張子 ACC を持つファイルに保存されます。これらのデータには、変数のアクセス記述が含まれます。

アプリケーション

指令 MAP_ACC_NC を使用すると、ACC ファイルを NCK から読み出して、DDE インタフェースで使用できるように準備することができます。すなわち、これらのファイルに対応する接続が NCDDE サーバに作成されるか、または NCDDE サーバに通知されます。

（注）この機能を使用して、ユーザーは NCDDE サーバに新規 NCK データを通知することができます。この機能を使用しないと、これらの変数／データにはアクセスすることができないでしょう。

この指令は、拡張子を持つ指令 COPY_FROM_NC と同じ働きをします：ACC-ファイルから得た情報はデコードされ、DDE インタフェースでのプレゼンテーション用に整えられます。

シンタックス

コールは、次のシンタックスに追従します :

MAP_ACC_NC

(WinFile, NcFile, TransferState, Area, DataBlock, Timeout, Prefix)

パラメータ

表 A-6 で、パラメータをさらに詳しく述べます。最初の 3 つのパラメータは、他のドメインサービスのパラメータに類似しています (第 0 章を参照してください)。パラメータの概要を次の表にしました。

表 A.6 指令 MAP_ACC_NC のパラメータ

名前	説明
WinFile	MMC エリアでの情報のソースとデスティネーション
NcFile	NCK/PLC 環境でのファイル名
TransferState	変数が転送ステータスを決定する
Area	ACC データのエリアアドレス、11.1.1 章、表 11-1 に説明。次が完全なリストです : エリア — エリアアドレス NCK — 0 モードグループ — 1 チャンネル — 2 軸 — 3 ツールオフセット — 4 フィードドライブ — 5 メインスピンドルドライブ — 6 予約 — 7
DataBlock	Variable Service 「変数サービス」のモジュールタイプ : 十六進値の範囲は 00 から FF, 11.3.1 章に説明、例えば (引用): モジュール識別子 — モジュール番号 (DataBlock) システムステータスデータ (Y) — 10 グローバルユーザーデータ (GUD) — 17 OEM ツールデータ (TU) — 24 マガジンディレクトリ (TMV) — 2B
Timeout	NCK-MMC トランザクション実行時間のモニタリング（単位：秒）
Prefix	ストリング、ACC 変数の直前に挿入される

（注）パラメータ WinFile が拡張子 .NSK を持つファイルの場合、
ACC だけでなく Domain Service も NSK ファイルを生成し、このファイルに対応する LINK 指令が含まれます。

ACC ファイル

```
/NC/_N_NCK_GD2_ACC ; グローバル NCK ユーザー変数 MGUD
/NC/_N_CH02_GUD_ACC ; 第 2 チャンネルのグローバルユーザー変数
/NC/_N_AX_SEA_ACC ; 軸別セッティングデータ
/NC/_N_CH_TEA_ACC ; チャンネル別 NC マシン
```

ドライブマシンデータ用の接続を設定する

MAP_ACC_NC	指令ヘッダ
L:\MMC2\NCMDACC.NSK	Windows 環境でのファイル名
/NC/_N_VS_DIR/_N_VS_TEA_ACC	NC- ドメイン
trans	変数 TransferState
5	エリア , 数 5 は "drives" のエリアアドレスを意味する
7F	DataBlock, アドレス 7F はモジュール "service values of drives" 「ドライブのサービス値」を意味する
10	タイムモニタリング , この場合 10 秒
/ACC/driveVSA/MD/	プレフィックス、後にデータにアクセスする際に使用されるストリング

例 A-30　　ドライブマシンデータ用の接続を設定する

```
Sub Form_Load ()
    Label1.LinkTopic = "NCDDE|NCU840D"
    Label1.LinkMode = 2
    Label1.LinkExecute "MAP_ACC_NC (L:\MMC2\NCMDACC.NSK,
    /NC/_N_VS_DIR/_N_VS_TEA_ACC,trans,5,7F,10,/ACC/driveVSA/MD/)"
End Sub
```

すでに設定された接続にアクセスする

次の構成要素を持つ以前の例ですでに設定されているリンクにアクセスする：

/ACC/driveVSA/MD/	MAP 指令の前のコールから生じたプレフィックス
$MD_TORQUE_THRESHOLD_X[1]	マシンデータの名前 , $ で始まる

例 A-31　　設定された接続にアクセスする

```
Sub Form_Load ()
    Label1.LinkTopic = "NCDDE|NCU840D"
    Label1.LinkMode = 2
    Label1.LinkItem= "/ACC/driveVSA/MD/$MD_TORQUE_THRESHOLD_X[1]"
    Label1.LinkRequest
End Sub
```

グローバルユーザー変数にアクセスする

8.13 章に、設定された接続にアクセスする方法を説明しています。

MAP_ACC_NC 指令のいくつかの例

パラメータ WinFile および NcFile の後にはコンマとスペースが 1 つづつきます。

例 A-31a　　MAP_ACC_NC 指令の例

全てのマシンデータ：

MAP_ACC_NC(c:\tmp\c.nsk, /NC/_N_COMPLETE_TEA_ACC, trans,0,1A,10,/MD/)

チャンネル 1 のチャンネルマシンデータ

MAP_ACC_NC(c:\tmp\ch1.nsk, /NC/_N_CH1_TEA_ACC, trans,2,1A,10,/CH1/)

すべての軸指定マシンデータ

MAP_ACC_NC(c:\tmp\ax.nsk, /NC/_N_AX_TEA_ACC, trans,3,1A,10,/AX/)

すべての NC グローバルセッティングデータ：

MAP_ACC_NC(c:\tmp\sea.nsk, /NC/_N_NC_SEA_ACC, trans,0,16,10,/SEA/)

全ての軸指定セッティングデータ

MAP_ACC_NC(c:\tmp\axs.nsk, /NC/_N_AX_SEA_ACC, trans,3,16,10,/AXSEA/)

PI サービス

概要

PI サービスを使用して、NCK と PLC に指令を転送することができます。11.1.4 章に PI サービスの完全なリストが記載されています。

NCDDE サーバの PI サービスには次のようなものがあります :

PI_START	NCK に指令を実行するように指示する
PI_START_BINARY	PLC に指令を実行するように指示する
PI_STOP	NCK に指令の実行を中止するように指示する
PI_STOP_BINARY	PLC に指令の実行を中止するように指示する
PI_RESUME	NCK に中止された実行をレジュームするように指示する
PI_RESUME_BINARY	PLC に中止された実行をレジュームするように指示する

PI_START(_BINARY)

説明

この機能を使用すると、MMC から NCK へ指示を送ることができます。

アプリケーション

この機能は、NCK でジョブを開始するのに適しています。

NCK への転送には、バイナリの転送は使用しないほうが良いでしょう。
バイナリ転送は、PLC 、NC 、およびドライブへのデータ転送に適しています。

シンタックス

PI サービスをコールする指令文字列は、シンタックスに追従します :

PI_START(サーバ名 , パラメータ 1, パラメータ 2 ... パラメータ n, PI 名)

PI_START_BINARY (サーバ名 , パラメータ , PI 名)
NCK 用の PI 名は _N_ で始まり、その後に 6 文字続きます。PLC に適用される規則とは少し異なります。

パラメータ

パラメータの機能は使用される PI サービスによって異なるため、パラメータについてはオンラインヘルプに詳しく説明しています。

パートプログラムを選択する

この例では、PI サービス "SELECT" （チャンネルでの実行用プログラムを選択する）がパートプログラム " BSP.MPF" をどのように選択するか示しています。

ただし、間違いなくこの指令のエリアパス（NC ファイルのパスではありません）を入力してください。

例 A-32　　パートプログラムを選択する

```
Sub Form_Load ()
    Label1.LinkTopic = "NCDDE|NCU840D"
    Label1.LinkMode = 2
    Label1.LinkExecute "PI_START(0d0d,201,/_N_MPF_DIR/_N_BSP_MPF,
    _N_SELECT)"
End Sub
```

OB 1 を起動する

すでにパッシブファイルシステムに保存されている OB1 をアクティブにします：

例 A-33　　OB 1 を起動する

```
Sub Form_Load ()
    Label1.LinkTopic = "NCDDE|NCU840D"
    Label1.LinkMode = 2
    Label1.LinkExecute "PI_START_BINARY( /PLC, ""@1d1@1d0@@0800001P"",
    _INSE)"
End Sub
```

パートプログラムの選択を中止する

この例では、PI サービス "SELECT" （チャンネルでの実行用プログラムを選択する）がパートプログラム " BSP.MPF" をどのように中止するか示しています。

例 A-34　　パートプログラムの選択を中止する

```
Sub Form_Load ()
    Label1.LinkTopic = "NCDDE|NCU840D"
    Label1.LinkMode = 2
    Label1.LinkExecute "PI_STOP(0d0d,201,/_N_MPF_DIR/
    _N_BSP_MPF, _N_SELECT)"
End Sub
```

OB 1 の起動を中止する

例 A-35　　OB 1 の起動を中止する

```
Sub Form_Load ()
    Label1.LinkTopic = "NCDDE|NCU840D"
    Label1.LinkMode = 2
    Label1.LinkExecute "PI_STOP_BINARY( /PLC,
    ""@1d1@1d0@@0800001P"", _INSE)"
End Sub
```

その他の NCDDE サーバ指令

概要

その他の NCDDE サーバ指令は、表 0-7 に記載しています：

表 A.7 その他の NCDDE サーバ指令

指令	意味
NEW	ローカル変数を生成する
FREE	変数を削除する
ANIMATE	ローカル変数を常に変更する
CALL	ファイルで NCDDE 指令を実行する
PLC_MEMORYRESET	PLC メモリをリセットする

NEW

説明

NCDDE サーバにアクセスするためのローカル／内部変数を生成します。

アプリケーション

指令 NEW を使用すると、NCDDE サーバに新規ローカル変数が作成されます。

この変数にアクセスするのに NCK との通信は必要はありません。名前 VarName を持つ変数がすでに存在している場合、既存の変数は削除され、新しい変数が生成されます。（第 0 章の FREE 指令と同じ）。

シンタックス

NEW (VarName , Value)

表 A.8 NEW のパラメータ

パラメータ	シンタックス	説明
VarName	<String>	生成される変数の名前
Value	<Parameter>	変数の初期化値

内部変数を生成する

NCDDE サーバに変数 "test" を生成し、値 10.0 を使用して初期化します。

例 A-36 内部変数を生成する

```
Sub Form_Load ()
    Label1.LinkTopic = "NCDDE|NCU840D"
    Label1.LinkMode = 2
    Label1.LinkExecute " NEW ( test , 10.0 )"
End Sub
```

FREE

説明

NCDDE サーバの変数を削除する

アプリケーション

指令 "FREE" によって指令 "NEW" や "LINK" が生成した変数を削除することができます。ファイル転送サービス (第 0 章) が変数をステータス変数として現在使用中である場合、指令 "FREE" は拒絶されます。変数に Advise Links (Hotlinks) が作られている場合、そのリンクは削除されます。その他の NCK と PLC を使用するトランザクションも終了されます。

シンタックス

FREE (VarName)

表 A.9 FREE のパラメータ

パラメータ	シンタックス	説明
VarName	< String >	削除される変数の名前

内部変数を削除する

NCDDE サーバの変数 "test" を削除します。

例 A-37 内部変数を削除する

```
Sub Form_Load ()
    Label1.LinkTopic = "NCDDE|NCU840D"
    Label1.LinkMode = 2
    Label1.LinkExecute " FREE( test )"
End Sub
```

ANIMATE

説明

この機能により、NCDDE サーバは "NEW" を使用して生成したローカル変数の値を常に変更し続けます。値は、ほぼ　1 秒ごとに増加されます。

アプリケーション

自分で作成したアプリケーションのテストに使用できます。

シンタックス

Animate　　(VarName)

パラメータ

表 A.10 Animate のパラメータ

パラメータ	シンタックス	説明
VarName	< String>	変更される変数の名前

内部変数を変更する

NCDDE サーバの変数 " test" の値を常に変更し続けます。

例 A-38 内部変数を変更する

```
Sub Form_Load ()
    Label1.LinkTopic = "NCDDE|NCU840D"
    Label1.LinkMode = 2
    Label1.LinkExecute " ANIMATE( test )"
End Sub
```

CALL

説明

指令ファイルの翻訳

アプリケーション

指令 CALL NCDDE を使用すると、ファイルに記録されている指令を実行することができます。

ファイルのラインがそれぞれ指令として NCDDE サーバに送られます。ファイルにはコメントやスペースが含まれている場合もあります。NCDDE 指令ファイルにはすべて拡張子 .NSK がついています。

（注）この機能を使用すると、NCDDE サーバを自分で作成したアプリケーション用にカスタマイズすることができます。

シンタックス

CALL (FileName)

パラメータ s

表 A.11 CALL のパラメータ

パラメータ	シンタックス	説明
FileName	< String >	NCDDE 指令ファイルの名前

例

"\MMC2\ NCDDE311.NSK" ファイルを参照してください

PLC_MEMORYRESET

説明

NCDDE サーバの指令 PLC_MEMORYRESET を使用して PLC メモリをリセットすることができます。エリアアドレスとして、/PLC を指定してください。

アプリケーション

PLC メモリをリセットします

シンタックス

PLC_MEMORYRESET(AreaAdr)

パラメータ

表 A.12 PLC_MEMORYRESET のパラメータ

パラメータ	シンタックス	説明
Adrea Adr	< String >	エリアアドレス

PLC をリセットする

PLC をリセットするには、必ず前もって中止していなければなりません。

例 A-39 PLC をリセットする

```
Sub Form_Load ()
    Label1.LinkTopic = "NCDDE|NCU840D"
    Label1.LinkMode = 2
    Label1.LinkExecute "PLC_MEMORYRESET(/PLC)"
End
```

OEM ビジュアルベーシックコントロール (VBX ファイル)

概要

このコントロールを使用すると、Visual Basic の標準コントロールを使用する DDE 接続の不具合をいくつか補うことができます。

（注）OEM アプリケーションを開発する場合には、NCDDE サーバへのアクセスには OEM Visual Basic Controls を使用することを推奨します。

例えば、"label" や "text field" のような標準コントロールでは、DDE 接続が可能になります。しかし、この通信にはいくつか不満足な点があります：

- イベントが喪失する
  Linkmode = 1 で DDE 変数の値が変更された際に VB プログラムのチェンジプロシージャがコールされるという保証はありません（唯一の対策はタイマーコントロールを介して値をポールすることです)。
- DDE の機能性がネストされない
  DDE チェンジプロシージャでは、コントロールの DDE 機能をさらに起動することはできません（この場合の対策も、タイマーなどを使用することぐらいしかありません)。
- 同期トランザクションしか認識されない
  Hotlinks の設定やリクエストのレスポンスタイムインタバルが非常に長くかかります。これらのアクションに 2 つ以上の CPU （NCK 、PLC）が関わる場合は特に長くなります。
- リソースの要求が大きい
  それぞれのコントロールに、DDE 、DDE 会話を使用するインスタンスがインストールされています。
  この会話は、2 つの Windows ハンドルを使用しているので、ブロッキングによりユーザーのリソースが被害を受ける可能性があります。
- NCDDE で LastError が都合よく処理されない
  NCK との通信を NCDDE を介して行う場合、NCDDE サーバは詳しいエラーの分析に DDE 変数 LastError を出します。この変数は DDE 会話のそれぞれに個別であり、DDE リターンが DDE_FNOTPROCESSED の場合にのみこの変数は有効です。

File DDECTL.VBX

（注）DDECTL コントロールは、データとして転送されたエラーはデコードしません：NCDDE サーバから送られた #- 記号やスペースも含まれます。

DDECTL を使用する DDE 通信の改良

DDECTL には、フラグパラメータを使用するイベントプロシージャがあります。イベントが失われると、Paint メカニズムを介して別のイベントがトリガされます。

→イベントが行方不明にならない

DDECTL は、VB-CDK-API の DDE インタフェースではなく DDEML を使用します。

→ DDE アクションをネストすることができる

非同期のトランザクションが構築されます

→ NCK や PLC にインストールされた Hotlinks や Requests がはるかに高速で実行されます。通信の遅延時間は、例えばスクリーン表示などに使用することができます。

新規会話のインストールには通常時間がかかるので、DDE 会話はタスクが完了しないとインストールされません。

いくつかの DDE 項目を、"|" によって区切って項目プロパティに指定することができます。Data Property では、値は "|" で区切られて表示されます。

プロパティ

DDE トランザクションは、Mode Property を設定すると開始されます。それに先立ち、次の変数を設定する必要があります :

- Convmode
- Topic
- Item.

表 A.13 DDECTL のモード

モード	意味
0	実行中の DDE アクティビティがすべて終了される
1	Item Property に含まれる項目それぞれに対して、Hotlink が設定される。 非同期リクエストが発行される。よって、最低一度はデータプロシージャによって値が与えられる。
2	項目プロパティに含まれるすべての項目に対する同期リクエスト 値はデータプロシージャによって与えられる。

Topic: "|" によって分けられた DDE サービスおよび DDE トピック (Label Control と同様)

Item: モード = 1,2 の場合 : "|" によって分けられた DDE 項目のリスト

Convmode: なし

Data: モード = 1,2 を使用する "|" によって分けられた DDE 項目の値

DCTL のイベントプロシージャ

例 A-40　　イベントプロシージャの例

```
Sub DDECtl1_DDEData (Flag As Integer)　（フラグ「整数として」）
    Flag = 1
    mydata = DDECtl1.Data
End Sub
```

（注）Windows によって成果がチェックされるので、フラグは 1 にセットされなければなりません（確認応答）。フラグが 1 にセットされていない場合、常にイベントプロシージャがコールされます。

Paint を介した再試行

イベントハンドリングの Retry メカニズムは Windows Paint を使用するので、DDECTL は必ずフォームのビジュアルパートに置いてください。

File DCTL.VBX

概要

DCTL VBX コントロールは、拡張した DDE 性能を持つグラフィックコントロールです。Standard Control Label と似ていますが、次のような点で優れています：

- Windows リソース要求を最小限にする
  "DDE Requests", "DDE Pokes" および "DDE Executes" は一時的にしかリソースを必要としません。DCTL.VBX コントロールを使用する WINDOWS Process の "DDE Hotlinks" 全体でたった 1 つの Windows ハンドルしか必要としません。
- NCDDE サーバを使用するクローズコーポレーション（閉鎖環境）：
  例えば、うまくいかなかったトランザクションには "LastError" 値が配送されます。
- スピードアップ
  サーバを使用して複数の／並行したトランザクションを行うことができるので、アプリケーションはスピードアップされます。
- 出力のスピードアップ
  スクリーン出力やインデックスフィルタリングを最適化することにより、スクリーン表示が速くなります。さらに、BASIC プログラミングがさらに簡単になります。
- 副作用の回避
  Visual Basic Controls の典型的な副作用、例えば ESCAPE キーを押すとプログラムされている接続が終了するということが避けられます。

本章では、まずこの新しいコントロールのプロパティを述べ、そして次に付加的なイベントを指摘します。終わりにいくつかの例を挙げますが、そこで DCTL.VBX のアプリケーションを示します。

プロパティ

DCTL.VBX コントロールのプロパティはほとんど、Visual Basic 標準コントロールのプロパティに対応しています。例えば、

- Style プロパティ
- Color プロパティ
- Base プロパティ
- Drag プロパティ
- Font プロパティ

プロパティには DCTL.VBX コントロールとその他の Visual Basic コントロールを区別するものもあります :

- DDE プロパティ
- HorAlignment プロパティ
- VertAlignment/Multiline プロパティ
- WordBreak プロパティ
- TabSize プロパティ
- LastError プロパティ
- Data プロパティ
- DataToCaption プロパティ
- LinkCmd プロパティ
- LinkNext プロパティ
- LinkFilter プロパティ .

DDE プロパティ

DDE プロパティには、次のようなものがあります :

LinkItem

LinkTopic (デフォルトによる事前設定 NCDDE)

LinkTimeout (LinkCmd 同期化用)

HorAlignment プロパティ

このプロパティは、キャプション表示の水平テキスト行端揃えを制御します :

表 A.14 水平テキストの行端揃え

値	プロパティ
0	左端揃え (デフォルト)
1	右端揃え
2	中央揃え

VertAlignment/Multiline プロパティ

このプロパティは、キャプション表示の垂直テキスト行端揃えを制御します；水平テキスト行端揃えの代わりに、マルチライン表示を選択することもできます。その場合、ワードラッピングは、WordBreak プロパティによって決定されます：

表 A.15 垂直テキスト行端揃え

値	プロパティ
0	垂直の中央揃え (デフォルト)
1	上端位置揃え
2	下端位置揃え
3	マルチライン

WordBreak プロパティ

プロパティ VertAlignment/Multiline が t Multiline (値 = 3) を設定している場合、プロパティ WordBreak がワードラッピングを決定します：

表 A.16 ワードラッピングのタイプ

値	プロパティ
False	CR/LF によるワードラップ (キャリッジリターン・ラインフィード・シーケンス)
True	自動ワードラップ、ワードが文字列にフィットしない場合。キャリッジリターン・ラインフィード・シーケンスも、その文字列をラップする。

TabSize プロパティ

各タブのスペース数を指定します。デフォルト値は 8 です：許容最大数は、225 です。

LastError プロパティ

このプロパティを使用して、エラーメッセージを送ることができます。

サーバを使用する DDE トランザクションが開始すると、この値はリセットされて 0 に戻ります。トランザクション実行中にエラーが発生し、DCTL 制御がこのエラーを検出した場合、詳細なエラーコードを要求します（このエラーコードにはプロパティ LastError を使用してアクセスできます)。

（注）DCTL 制御はデータとして転送されたエラーはデコードしません：NCDDE サーバから送られた #- 記号やスペースも含まれます。

変数 LastError については、11.7.1 章に説明があります。

Data プロパティ

Data プロパティは、次の DDE トランザクションの引数として使用することができます。

表 A.17 DDE トランザクションの引数

DDE トランザクション	引数
Request	リクエストされた変数値 , "DataToCaption" プロパティが FALSE に設定されている場合
Advise Link	更新された値 , "DataToCaption" プロパティが FALSE に設定されている場合
Poke	転送される値
Execute	実行される指令

DataToCaption プロパティ

DataToCaption プロパティは、DDE トランザクションによって受けとられたデータのデスティネーションを決定します。

表 A.18 データのデスティネーション

値	意味
True	データのデスティネーションは、Caption プロパティ
False	データのデスティネーションは、Data プロパティ

LinkCmd プロパティ

プロパティ LinkCmd を変更すると、DCTL 制御の DDE アクティビティが開始されます。アクティビティがない場合、LinkCmd はイコール 0 になります。

表 A.19 LinkCmd プロパティ

番号	変更後	DDE アクティビティ	終了元
1	Advise Link	AdviseLink を設定する。Advise Link の設定後にリターンする。AdviseLink は、Stop 指令を使用して削除できる。	Stop
2	Advise Link_ NotifyData	"1 - AdviseLink" と同様 , DDE Data を受け取ると、追加アクション (1)。	Stop
3	Advise Link_ NotifyDataWhenVisible	"1 - AdviseLink" と同様 , DDE Data を受け取ると、追加アクション (2) 。	Stop
4	Advise LinkAsync	AdviseLink を設定する。Advise Link が設定される前にリターンする。AdviseLink は、Stop 指令を使用して削除できる。	Stop
5	Advise LinkAsync_ NotifyData	"4 - AdviseLinkAsync" と同様 , DDE Data を受け取ると、追加アクション (1) 。	Stop
6	Advise LinkAsync_ NotifyDataWhenVisible	"4 - AdviseLinkAsync" と同様 , DDE Data を受け取ると、追加アクション (2) 。	Stop
7	Stop	AdviseLink を削除する。AdviseLink が設定された後にリターンする。	Itself
8	StopAsync	AdviseLink を削除する。AdviseLink が設定される前にリターンする。	Sync
9	StopAsync_ Notify	"8 - StopAsync" と同様 , 設定が完了すると、追加アクション (1)	Sync
10	StopAsync_ NotifyWhenVisible	"8 - StopAsync" と同様 , 設定が完了すると、追加アクション (2)。	Sync
11	Request	DDE 変数を読み出す。読み出しが完了した後にリターンする。	Itself
12	RequestAsync	DDE 変数を読み出す。読み出しが完了する前にリターンする。	Sync
13	RequestAsync_Notify	"12 - RequestAsync" と同様 , DDE 変数を読み出す。読み出しが完了すると、追加アクション (1)。	Sync
14	RequestAsync_ NotifyWhenVisible	"12 - RequestAsync" と同様 , DDE 変数を読み出す。読み出しが完了すると、追加アクション (2)	Sync
15	Execute	指令をサーバに送る。指令の実行が完了した後にリターンする。	Itself
16	ExecuteAsync	指令をサーバに送る。指令の実行が完了する前にリターンする。	Sync
17	ExecuteAsync_Notify	"16 - ExecuteAsync" と同様 , 指令の実行が完了すると追加アクション (1)	Sync
18	ExecuteAsync_ NotifyWhenVisible	"16 - ExecuteAsync" と同様 , 指令の実行が完了すると追加アクション (2)	Sync
19	Poke	DDE 変数を書き込む。書き込みが完了した後にリターンする。	Itself
20	PokeAsync	DDE 変数を書き込む。書き込みが完了する前にリターンする。	Sync
21	PokeAsync_Notify	"20 - PokeAsync" と同様 , 書き込みが完了すると追加アクション (1)	Sync
22	PokeAsync_ NotifyWhenVisible	"20 - PokeAsync" と同様 , 書き込みが完了すると追加アクション (2)	Sync
23	Sync	同期指令と同様の非同期指令を終了する。機能している非同期指令がなければ作動しない。	Itself

アクション

使用されるアクションは前揚の表に記載しています :

アクション (1)

DdeNotify をコールしようとします。Visual Basic がこの時点でイベントプロシージャをコールしない場合、あるいはイベントプロシージャのパラメータがまだ変更されていない場合、DCTL 制御は DdeNotify イベントプロシージャが変更されていない限り、このイベントを送ろうと 1 秒間に 10 回試行します。

アクション (2)

DCTL 制御は、Windows からペイントメッセージを受け取ると、DdeNotify イベントをコールします。Windows によるペイントメッセージの生成を保証するために、左上角のピクセルを DdeNotify が変更されない限り無効にしておきます。実際、この機構は制御が見えない場合でも表示を抑制します。

（注）新規 DDE アクティビティを開始する前に、必ず以前の DDE アクティビティを終了してください。表の 右端欄（終了元）のパラメータを使用すると、これを行うことができます。

同一の Windows プロセスに置かれた全ての DCTL 制御の Hotlinks は、同じ "LinkTopic" プロパティを持っている場合、1 つの DDE 接続を共有します。その他のアクティビティ（ Hotlink 以外）の DDE 接続は、その場合その場に応じて生成され、削除されます。このような理由と、DCTL がウィンドウを持たないために、Windows リソースへの要求を劇的に削減することができます。

（注）LinkCmd プロパティの変更により、LinkTopic, LinkTimeout および LinkItem プロパティが評価されます。したがって、これらのプロパティに関連するエラーの中には LinkCmd プロパティが変更されるとリポートされるものもあります。この点で評価してください。

LinkNext プロパティ

オプションのプロパティ LinkNext は、名前と場合によっては別の DCTL 制御のインデックスを所有している。

LinkNext プロパティが空でない場合、DCTL 制御が NCDDE インデックス仕様（コロン（ : ）が最後に付く 5 桁）のために AdviseLink を介して転送されたストリングをスキャンします。このストリングをインデックスされたサブストリングに分割し、LinkNext プロパティによって構築された DCTL 制御の連鎖リストにしたがって転送します。その LinkFilter プロパティがインデックスにマッチする制御は、対応するサブストリングを受け取ります。このように処理されないサブストリングは失われてしまいます。

LinkFilter プロパティ

LinkFilter 値は、0 から 65535 の範囲に及びます。その使用方法については、LinkNext のパラグラフで説明します。

DCTL.VBX 用イベント

DCTL 制御 イベントは大部分が、他の Visual Basic 標準制御と同じです :

- Click
- DblClick
- MouseDown
- MouseMove
- MouseUp
- DragDrop
- DragOver.

イベント DdeNotify

特に DDE 通信用に DdeNotify イベントが実現されました：新規 AdviseLink データの到着ならびに非同期 DDE トランザクションの結論を示します。アプリケーションの詳細については、LinkCmd (アクション (1) および (2)) のパラグラフで説明します。

シンタックス

Sub ctlname_DdeNotify (Index As Integer , Flug As Integer)

引数を使用する場合

制御アレイと Flag 内にある場合、制御の一意識別子としてのインデックス、ならびにイベントが基本レベルに実際に到着したことを DCTL 制御に知らせるフラグ

フラグ引数が変更するまで DCTL 制御は DdeNotify イベントを実行し続けるため、Flag の値はイベントプロシージャのコールの度に変更されると思ってください。この変更が行われない場合、協力式固定アクティビティが起こり、システムに不必要な負担をかけます。

DCTL.VBX を適用する

変数を読み出して表示する

DDE 変数を読み出して画面に表示する

DDE 変数を読み出し、読み出し値をすぐに使用したい場合、その値をスクリーン上に表示しなければなりません。

その場合、DCTL 制御（名前、例えば、Dctl1）を変数を表示させたい画面上に置きます。その際のコードは次のようになるはずです :

例 A-41 変数の読み出しと表示

```
Sub Form_Load ()
   Dctl1.LinkItem = "/Channel/Parameter/R[1]" （変数名）
   Dctl1.DataToCaption = TRUE （デフォルトとであれば省略できる）
   Dctl1.LinkCmd = 11（読み出しを命令する）
   （ここで、Dctl1.Caption は DDE 変数の値を保有する）
End Sub
```

Data プロパティの DDE 変数を読み出す

DDE 変数を Data プロパティに読み込む

DDE 変数を読み出し、読み出し値をすぐに使用したい場合、その値をスクリーン上に表示される必要はありません。

ラベリング DCTL 制御（名前、例えば、Dctl1）の一つをフォームで使用することができます。その際のコードは次のようになります：

例 A-42　　Data プロパティに読み込む

```
Sub Form_Load ()
   Dctl1.LinkItem = "/Channel/Parameter/R[1]"（変数名）
   Dctl1.DataToCaption = FALSE（Data プロパティへデータをルーティングする）
   Dctl1.LinkCmd = 11（読み出しを命令する）
   （ここで、Dctl1.Data は DDE 変数値を保有する）
End Sub
```

DDE 変数を書き込む

DDE 変数を書き込む

DDE 変数を書き込みたいとします。

その場合、ラベリング DCTL 制御（名前、例えば、Dctl1）の一つをフォームで使用します。その際のコードは次のようになります：

例 A-43　　変数を書き込む

```
Sub Form_Load ()
   Dctl1.LinkItem = "/Channel/Parameter/R[1]"（変数名）
   Dctl1.Data = 12（値）
   Dctl1.LinkCmd = 19（書き込みを命令する）
   （ここで、NC 変数はすでに 12 に設定されている）
End Sub
```

DDE 指令を実行する

DDE 指令を実行する

DDE 指令 をサーバに送りたいとします。

その場合、ラベリング DCTL 制御（名前、例えば、Dctl1）の一つをフォームで使用し、その際のコードは次のようになります :

例 A-44　　指令を実行する

```
Sub Form_Load ()
   Dctl1.Data = "Pi_start(/NC,001,_N_SET_OF)" （指令）
   Dctl1.LinkCmd = 15（指令を送信する）
   （ここで、指令はすでに実行されています）
End Sub
```

DDE Hotlink を表示する

DDE Hotlink を表示する

DDE 変数の値を画面に表示するとします。

その場合、DCTL 制御（名前、例えば、Dctl1）を変数を表示させたい画面上に置きます。さらに、選択するだけで、ホットリンク作成を開始し、作成タスクを DCTL 制御のバックグラウンドアクティビティにすることができます。

その際のコードは次のようになります。しかし、設計時に設定されたコード化したプロパティを実行することもできます。

例 A-45　　Hotlink から DCTL

```
Sub Form_Load ()
   Dctl1.LinkItem = "/Channel/Parameter/R[1]" （変数名）
   Dctl1.DataToCaption = TRUE（デフォルトの場合、省略される）
   Dctl1.LinkCmd = 4 （ホットリンクの作成を開始する）
End Sub
```

DDE をスピードアップする : 平行アクション

平行アクションを通じてスピードアップを図る

フォームがロードされていて、いくつかの独立した DDE アクティビティを実行する必要があるとします。その場合、速いフォームロードに関心を持つでしょう。DCTL 制御に関しては、DDE アクティビティを平行に実行するのが最善だからです。次にコードの例を示します。

例 A-46　　平行アクションを通じてスピードアップを図る

```
Sub Form_Load ()
（変数 1 の読み出しを開始する）
Dctl1.LinkItem = "/Channel/Parameter/R[1]"（変数名）
Dctl1.LinkCmd = 12（読み出しを開始する）
（変数 2 の読み出しを開始する）
Dctl2.LinkItem = "/Channel/Parameter/R[2]"（変数名）
Dctl2.LinkCmd = 12（読み出しを開始する）
（変数 3 の読み出しを開始する）
Dctl3.LinkItem = "/Channel/Parameter/R[3]"（変数名）
Dctl3.LinkCmd = 12（読み出しを開始する）
（ホットリンクを開始し表示する）
Dctl4.LinkItem = "/Channel/Parameter/R[4]"（変数名）
Dctl4.DataToCaption = TRUE （デフォルトの場合、省略される）
Dctl4.LinkCmd = 4（ホットリンクを作成する）
（指令の実行を開始する）
Dctl5.Data = "Pi_start(/NC,001,_N_SET_OF)"（指令）
Dctl5.LinkCmd = 16（指令の実行）
（ここで、変数アクセス、ホットリンク作成、および指令は平行して作業を行う。どの作業が完了したかを確認はできない。）
Dctl1.LinkCmd = 23（変数 1 の読み出しまで待機する）
Dctl2.LinkCmd = 23（変数 2 の読み出しまで待機する）
Dctl3.LinkCmd = 23（変数 3 の読み出しまで待機する）
Dctl5.LinkCmd = 23（指令が実行されるまで待機する）
（ここで、変数アクセスおよび指令が完了する、ホットリンクは、
できるだけ早く値を画面に表示する）
End Sub
```

DDE のスピードアップ : テキスト割付け

テキスト割付け機能を使用してスピードアップを図る

高周波数で多くのデータ項目を表示する必要があるとします。そして、使用しているプログラムが表示タスク用の BASIC 言語を入力できません。さらに、転送されたデータの量を最小化する必要があります。NCDDE サイドでは、アレイアクセスとアレイアクセスと "Field" データプレパレーションが組合わさってこの要求事項をサポートします。これらの機能を採用すると、DCTL 制御はマルチライン表示とインデックスフィルタリングを行えるようになります。

例 A-47　　テキスト割付け機能を使用してスピードアップを図る

```
NCDDE array access with "Field" data preparation - Dctl index filtering :
（5 つの異なる制御での 5 つの値の高周波数表示）
Dctl1.LinkItem = "/Channel/Parameter/R[1,5](!""!d%12.5g"")"（変数）
Dctl1.LinkFilter = 1（アクセスしたデータのインデックス）
Dctl1.LinkNext = "Dctl2"（次の制御へのリンク）
Dctl2.LinkFilter = 2（受容したデータのインデックス）
Dctl2.LinkNext = "Dctl3"（次の制御へのリンク）
Dctl3.LinkFilter = 3（受容したデータのインデックス）
Dctl3.LinkNext = "Dctl4"（次の制御へのリンク）
Dctl4.LinkFilter = 4（受容したデータのインデックス）
Dctl4.LinkNext = "Dctl5"（次の制御へのリンク）
Dctl5.LinkFilter = 5（受容したデータのインデックス）
Dctl1.LinkCmd = 4（ホットリンク作成の開始）

NCDDE array access - Dctl multiline display:
（カラムでの 5 つの値の高周波数表示）
Dctl1.LinkItem = "/Channel/Parameter/R[1,5](""!d%12.5g"
Dctl1.LinkItem = Dctl1.LinkItem + Chr$(13) + Chr$(10)+""")"
Dctl1.DataToCaption = TRUE （デフォルトの場合、省略される）
Dctl1.VertAlignment = 3（マルチライン選択）
Dctl1.LinkCmd = 4（ホットリンクの作成を開始する）
```

通知機能を使用する

通知機能を使用する

使用している画面表示が DDE にアクセス可能な変数に依存しているとします。その場合、この変数を DCTL 制御にホットリンクすることができます。そして、変数値が変更された場合には DCTL 通知機能を使用してレイアウトを再割付することができます。割付は時間のかかるタスクです。したがって、再割付けはフォームが画面上に見られないときには実行しないでください。

例 A-48　　変更をリポートする

```
Sub Form_Load ()
    （通知 "when visible" を使用してホットリンクを作成する基本コード）
    Dctl1.LinkItem = "/Channel/Parameter/R[1]"（変数名）
    Dctl1.LinkCmd = 6（ホットリンクの作成を開始する）
    （通知タスク用ハンドラ）
End Sub
Sub Dctl1_DdeNotify （インデックス「整数として」、フラグ「整数として」）
    Flag = Flag + 1（フラグは変わらなければならない）
    ... （再編成）
End Sub
```

エラーハンドリング

読み出し、書き込み、実行時の典型的なエラー処理

例 A-49　　エラー処理

```
On Error Goto TypicalErrorHandling
Dctl1.LinkCmd = 11（DDE アクティビティ）
...
TypicalErrorHandling:
   Select Case Dctl1.Lasterror \ 16777216（エラーソースによる選択）
   Case 2（MPI レベルエラー）
   ...（例、NC との接続なし）
   Case 3 , 5（NC/PLC レベルエラー）
   ... （例、存在する変数なし）
   Case 7（Dctl レベルエラー）
        Select Case Dctl1.Lasterror MOD 256 （エラーコードによる選択）
        Case 7（Dctl レベルタイムアウト発生）
           ...
        Case Else（その他の Dctl レベルエラー）
           ...
        End Select
   Case Else（その他のエラーソース）
        ...
   End Select
   ....
```

NCDDE アクセス用診断機能

NCDDE サーバの試験機能

概要

特に、NCDDE サーバの試験機能により、ファイルが生成された時点で NCDDE サーバに宣言されたローカル変数および外部変数に関する情報を得ることができます。試験機能は、次のようにコールします :

1. YS 840DI MMC-OEM プログラムグループの NCDDE サーバをスタートします
2. ALT+TAB を使って NCDDE プログラムに変更します、すなわち、NC 通信 DDE サーバ : アイコンが作成されます
3. アイコンをクリックします : 次のようなウィンドウが現れます。

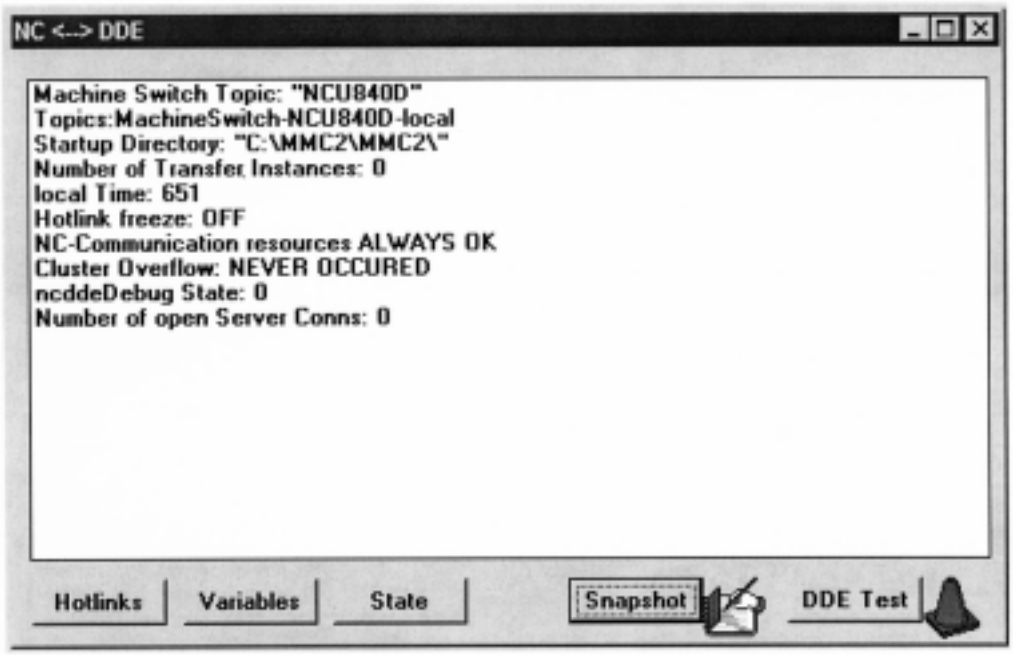


図 A.3 NCDDE サーバの標準画面

これらの機能の目的は主に、NCDDE サーバーの環境のデバッギングです。

Hotlinks

既存の Advise Links (Hotlinks と Warmlinks) を全て包含したリストが作成されます。次のような意味を持つ 5 コラムの表で構成されます。

表 A.20 ホットリンク

コラム	情報	所見
1	PDU リファレンス	内部値 : NCK および PLC を使用する通信の可能性として考えられる PDU リファレンス
2	Advise Link	LOCAL　ローカル変数へのリンク REMOTE　外部変数へのリンク PILED　別のジョブには外部 Advise Link が追加される
3	Update 時間	最後の PDU の時間が NCDDE サーバの内部ユニットでリフレッシュする
4	Lasterror 変数	Lasterror 仕様は本文書の 11.7 章に準じる。1 つの接続に対する複数のトランザクションの最後のエラーがリクエストされる場合もあるので、必ずしもサーバの DDE インタフェースでリポートされた値と同じにはならない。
5	変数名	変数名は第 11 章に準じる

Variables

ここでリストが生成され、NCDDE サーバが接続される全ての変数が含まれる、またここに変数が置かれる : "LOCAL" または "PLC/NC"。

Snapshot

このボタンを押すと、"NCDDE_X.TXT" という名前のファイルが生成される。このファイルには、ステータス、ホットリンク 、NCDDE サーバの変数が含まれる。

DDE Test

このボタンをクリックすると、次のような機能を持つ試験プログラム "DDETEST.EXE" が開始される。

表 A.21 DDE 試験指令

指令	アクション	意味
Passive	None	ステータスをリセットする , 機能はアクティブでない
Hotlink	Start	Advise Link を設定する
Request	DoIt	変数を読み出す
Poke	DoIt	変数を読み込む
Execute	DoIt	サービスを実行する

Service|Topic 例えば、: NCDDE|NCU840D 環境でインストールされた NC を指定します。

"DEFAULT_NC" はファイル "MMC.INI" からこの設定を読み出します。

指令機能は、5 つの候補のうちの 1 つをクリックして切替えます。

LastError エラーメッセージについては、11.7 章に説明します。

接続のステータス

変数 NcState

サーバは、そのローカル変数 NcState を介して CNC との接続のステータスが判明するようにします。この変数はサーバがスタートされた直後から存在します。DDE インタフェースを介して変更することができないという点で、サーバの他のローカル変数とは異なります。

変数は次のステータスのうち 1 つを持つことができます：

表 A.22 変数 NcState のステータス

値	意味
0	通常動作
1	CNC との接続に不具合がある
2	CNC との接続が全て失敗した
3	スタートアップファイルを翻訳する
4	サーバの初期化

トラブルシューティング

NCK が出すエラーメッセージ

リソースの欠如、アクセス違反、誤動作モードなどのエラー状態は、トランザクションの確認応答を介して NCK からリポートされます。NCDDE サーバがこれらのエラー状態を処理できない場合、DDE インタフェースのエラー状態を持つトランザクション Request, Poke および Execute がエラーステータスを使用して終了されます、すなわちアプリケーションは結果を得られません。

変数 LastError

最終トランザクションに関する情報を持つ変数によって、詳細な診断が与えられます。the Link Item LastError を使用して読み出すことができます。そして、読み出された後 0 に設定されます。この変数は常に NCDDE サーバに登録された最後のエラーを表示します。

変数 LastError は 4 バイトからなり、それぞれのバイトには大きい方から順に (High Byte → Low Byte) 次のようなエラーグループが含まれています：

- 上位エラークラス、エラーソース
- エラーエリア
- エラークラス
- エラーコード

多岐にわたるエラーコードの意味は、11.7 章の NCDDE エラーメッセージのパラグラフに記載しています。

NCK との接続の失敗

接続が失敗した場合、NCDDE サーバはアクティブなトランザクション Request, Poke および Execute に「ネガティブに」通知します。接続が再設定されない限り、その後のトランザクションの実行は拒否されます。また同時に、サーバは NCK との接続をレジュームしようとします。接続のステータスは、サーバのローカル変数 NcState に表示されます。

Advise Links を処理する

Advise Link 接続が失敗した場合、NCDDE サーバによってレジュームされた値は '#' です。Advise Link は、接続が再設定されると NCK で復活されます。

NCDDE サーバのリソースの欠如

NCDDE サーバ用に関してリソースの欠如が発生した場合、対応する DDE インタフェーストランザクションはエラーコードにより終了されます。

ネットワークを介したアクセス用に NCDDE サーバを環境設定する

β

ベータリリース：

これは製品特徴であり、開発環境でのみ機能します。ご使用の製品リリースには使用できません。

概要

MMC エリアでは、Workgroups オペレーティングシステム WfW

3.11 用の MS-WINDOWS または WINDOWS NT が使用されています。そのため、PC ネットワークに接続されたどの WINDOWS-PC からでも NCK のデータにアクセスすることができます。

（注）Windows NT では、次のものが付属されていなければなりません："\WINDOWS\NETDDE.EXE" プログラムとのリンクは、"AUTOSTART" フォルダに生成されます。

そのための前提条件は、MMC がハードウェアとソフトウェアを使用して Windows ネットワークに統合されていることです（図 A-4 ）。次のステップにしたがって行ってください：

- ISA スロットに挿入されたネットワークアダプタを使用する物理的な接続
- WfW を介したネットワークとの接続
- ネットワーク DDE シェアマネージャを使用する MMC のファイル SYSTEM のセクション [DDEShares] にエントリの追加
- Windows PC のファイル MMC.INI および REGIE.INI の適切な修正
- MMC のスタート
- Windows PC のスタート
- 接続の試験

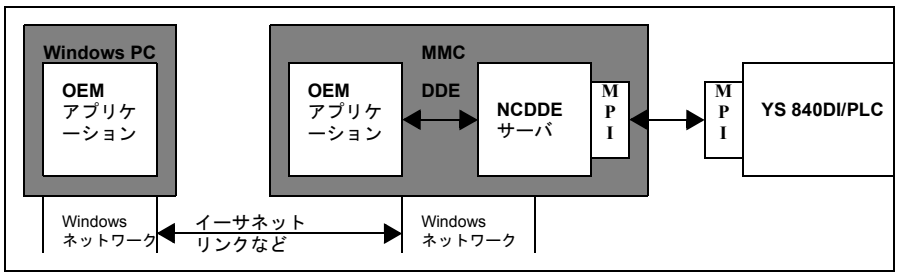


図 A.4 MMC のネットワーキング

アプリケーション

このアプリケーションを使用すると、PC 上で MMC ユーザーインタフェースを実行することができます。また、自分自身のアプリケーションを実行したり、"NETDDE" を介して NCK にアクセスすることもできます。

MMC の環境設定

NCDDE サーバは、指令 "New Share" を使用して "Network DDE Share Manager" "\MMC2\DDESHARE.EXE" を介して Windows ネットワークに認知してもらわなければなりません。OEM パッケージには、プログラム "DDESHARE.EXE" が含まれています。

次の環境設定を使用します：

表 A.23 ネットワーキング用に DDESHARE を環境設定する

識別	エントリ入力	意味
Share Name	NCU840D$	ネットワークに認知された MMC の識別子
Application Name	ncdde	ネットワークアクセスを介してシステムで今後使用されるアプリケーションの名前
Topic Name	NCU840D	DDE 接続を設定するための名前の一部 (PC のファイル MMC.INI に NcddeMachineName によって定義されているものと同じであること)
Access Type	Full	パスワードの鍵を使用しない書き込み／読み出しアクセス

（注）次の例は Windows 3.x にのみ適用されます。Windows NT では、プログラム "DDESHARE.EXE". . によってレジストリに入力されます。

「シェアマネージャ」は終了されるとき、ファイル "SYSTEM.INI" のセクション [DDEShares] に次の文字列を入力します。

DDEShares

ファイル "SYSTEM.INI" のセクション [DDEShares] への入力例。

例 A-50　　DDEShares エントリ

```
[DDEShares]
MMC2HW0$=ncdde,NCU840D,,15,,0,,0,0,0
```

（注）エントリ「Share Name」はオプションです（この場合、NCU840D$)。
エントリ「Application Name」および「Topic Name」は、MMC のファイル MMC.INI のセクション [GLOBAL] のエントリ "NcddeMachineName" および "NcddeServiceName" と同じでなければなりません。

MMC を再始動すると、ネットワークを介して NCDDE サーバにアクセスが可能になります。

Windows PC の環境設定

Windows ネットワークを介して、NCK データを実行中の MMC-OEM アプリケーションと当然通信する Windows PC では、ファイル

MMC.INI

REGIE.INI

を次のように変更する必要があります :

MMC.INI での入力

ファイル MMC.INI のセクション [GLOBAL] には、次のように入力します。

例 A-51　　MMC.INI での入力

```
[GLOBAL]
NcddeMachineName=NCU840D$ ;「Share Name」です。
NcddeServiceName=\\SIN840D\NDDE$ ; コンピュータ名です。
```

REGIE.INI での入力

Startup2 = name := ncdde

が、Windows PC のファイル REGIE.INI のセクション [StartupConfiguration] のコメントにマークされている場合、MMC システムの NCDDE サーバがネットワークを介して（すなわち、それ自体のネットワークは介さず）アドレス指定されるので、それ自体の NCDDE サーバは始動できなくなります。

例 A-52　　スタートアップエントリをファイル REGIE.INI のコメントにマークする

```
[StartupConfiguration]
; 自身の NCDDE サーバをスタートしない
; Startup1 = name := ncdde, Timeout := 20000
```

サーバのシーケンスを開始する

ネットワークと通じて NCDDE 通信を開始する前に、クライアントのシステムに置かれた NCDDE サーバ（この場合、MMC）を起動しなければなりません。そして、Windows PC を起動すると、ファイル MMC.INI および REGIE.INI に加えられた変更が有効になります。

その後は、ローカル NCDDE サーバーアクセスと同じインストラクションシンタックスを使用してください。

接続を試験する

プログラム 'NCDDE test' 「NCDDE 試験」を使用し、外部 Windows PC が MMC 102 の NCDDE サーバーに正しくアクセスできるかチェックすることができます。そのために、\\SIN840D\NDDE$|NCU840D$ を使用してエントリ "Service/Topic" 「サービス／トピック」を設定する必要があります（そして、当然ながら MMC 102 上では NCDDE サーバが走っていなければなりません）。

ネットワークの複数の MMC

複数の MMC がネットワークを介してアドレス指定される場合、これらの MMC には DDESHARE.EXE を使用して異なる Share Names を割当てなければなりません。

グローバルユーザー変数 GUD, SGUD, MGUD, UGUD, GD3 to GD9 にアクセスする

概要

グローバルユーザー変数は、NCK と 個別のチャンネルのいずれにも使用できます。

制御ごとの 1 インスタンスに NCK 別グローバルユーザー変数は存在します。この変数は、チャンネル間のプログラムコーディネーションだけでなくチャンネル独立設定にも適しています。

チャンネル別グローバルユーザー変数はそれぞれのチャンネルに 1 度は存在します。この変数は、チャンネル別設定ならびに 1 つのチャンネルで実行している異なるプログラム間でのデータ転送に適しています。

ローカルユーザーデータについても同様です。

まず第一に、ユーザー変数を定義し、それから起動してください。そうすると、NCDDE サーバがそれらにアクセスできます。クラスタ変数については、対応する NSK ファイルを生成し、統合してください。次の 5 つのステップで行います：

1. 定義ファイルを作成します
2. この定義を NCK ディレクトリの /_N_DEF_DIR にコピーします
3. INITIAL.INI ファイルをロードしてユーザーデータを *.ACC-file として起動します
4. MAP 指令を使って *.NSK-file を作成します
5. NCDDE サーバの NSK ファイルに作成した *.NSK ファイルを追加します

定義ファイル：

グローバルユーザー変数は、固定名を使用して定義ファイル（モジュール）に定義してください：

- _N_GUD_DEF GUD の場合
- _N_SGUD_DEF GD1 = SGUD の場合　当社グローバルデータ
- _N_MGUD_DEF GD2 = MGUD の場合　グローバルデータマシンツールビルダー
- _N_UGUD_DEF GD3 = UGUD の場合　グローバルデータユーザー
- _N_GUD4_DEF から _N_GUD9_DEF　GD4 から GD9 の場合

これらのファイルは NCK の /_N_DEF_DIR ディレクトリに保存されています。

グローバルデータを定義するファイルの総数は、一般マシンデータ 18118 (MM_NUM_GUD_MODULES) 値によって異なります。（詳細については、スタートアップマニュアルを参照してください）。このマシンデータのデフォルト値は 4 です。

グローバルデータの定義:

グローバルデータの定義には、以下を使用します:

- 定義ヘッダ　　DEF
- エリア NCK または　　CHAN
- タイプ　　例 REAL 、INT
- 変数名　　例 LIFTOFF_DIST
- 次元　　角括弧に入れる
- コメント　　セミコロンで始まるオプションのテキスト

詳細については、「Yaskawa Siemens 840DI ユーザーズマニュアル プログラミング編 上級説明書（NCSI-SP02-07)」をご覧ください。

定義ファイルを作成する:

定義ファイルは、NCK または MMC に作成することができます。

NCK に作成する場合 :

グローバルユーザー変数用の定義ファイルは、NCK のパートプログラムレベルに作成する場合と同様、必ず /_N_DEF_DIR ディレクトリに入れてください。このファイルには次のものが含まれています :

- 第 1 行のプログラム識別子
- パス仕様を持つコメントライン（評価の対象となる）
- 定義
- 命令 M02, M17 あるいは M30 を終了する

例 A-53　　NCK にグローバル変数を定義する

```
%_N_MGUD_DEF
; $PATH=/_N_DEF_DIR
DEF NCK REAL RUECKZUG ; NCK 用のグローバル変数を定義する
DEF CHAN INT TABLE[100] ; チャンネル別変数を定義する
DEF CHAN REAL BLF_OFFS_X
M17 ; RETURN を使用してこのラインを終了する
```

MMC に作成する場合 :

ファイル名 MGUD.DEF を持つグローバル変数用の定義ファイルは、MMC に作成する場合と同様、ディレクトリ C:\TMP に置く必要があります。このファイルには、次のものが含まれます :

- 定義
- 命令 M02, M17 または M30 を終了する

例 A-54 MMC にグローバルデータ変数を定義する

```
DEF NCK REAL RUECKZUG ; NCK 用のグローバル変数を定義する
DEF CHAN INT TABLE[100] ; チャンネル別変数を定義する
DEF CHAN REAL BLF_OFFS_X
M17 ; RETURN によってこのラインを終了する
```

（注）MMC は COPY_TO_NC ドメインサービスを使用して、このファイルを NSK の /_N_DEF_DIR ディレクトリ転送します :
COPY_TO_NC(C:\TMP\MGUD.DEF,/NC/_N_DEF_DIR/_N_MGUD_DEF,trans)

ユーザーデータを起動する

INITIAL.INI という名前のファイルを NCK にコピーすると、ユーザーデータが起動します。

このファイルは非常に短いかもしれません：M17 の後に RETURN を入力するだけです。次は、ディレクトリ C:\TMP に置かれたファイル INITIAL.INI に適用されます :

COPY_TO_NC(C:\TMP\INITIAL.INI, /NC/_N_INITIAL_INI, trans)

このように、NCK に次のような名前を持つ 2 つの ACC ファイルが生成されます :

_N_NC_GD2_ACC　　グローバルユーザー変数用

_N_CH_GD2_ACC　　チャンネル別ユーザー変数用

(MGUD = GD2 を使用する上記の例にも適用できます).

（注）これによりスタティックメモリは再フォーマットされるので、ファイル INITIAL.INI をロードする前には必ず、プログラム、フレームおよびマシンデータのバックアップを取っておいてください。

NCK 用の NSK ファイルを作成する

MAP 指令をコールすると、ACC ファイルから NCK のグローバルユーザー変数用の対応する NSK ファイルを作成することができます。これらのファイルは、ACC ファイルと同じ名前を持っています。

次に、Visual Basic でのコールの例を示します。

"MAP_ACC_NC" 指令をコールする

C:\MMC2\MGUD_NCK.NSK	: Windows 環境でのファイル名
/NC/_N_NC_GD2_ACC	: NC- ドメイン
trans	: 変数 TransferState
0	: エリア NCK
2D	: モジュールタイプ MGUD
10	: トランザクションのタイムリミット :10s
/ACC/NCK/MGUD/	: 変数名用のプレフィックスとして使用する任意のストリング

例 A-55 NCK 用の NSK ファイルを作成する

```
Sub Form_Load ()
    Label1.LinkTopic = "NCDDE|MMC2HW0"
    Label1.LinkMode = 2
    Label1.LinkExecute "MAP_ACC_NC(C:\MMC2\MGUD_NCK.NSK,  /NC
    /_N_NC_GD2_ACC, trans, 0, 2D , 10, /ACC/NCK/MGUD/)"
End Sub
```

チャンネル用の NSK ファイルを作成する

MAP 指令をコールして、ACC ファイルからそれに対応するチェンネル別グローバルユーザー変数用の NSK ファイルを作成することができます。これらのファイルは、ACC ファイルと同じ名前を持ちます。

Visual Basic 環境でのコールの例を示します。

「MAP 指令」をコールする

C:\MMC2\MGUD_CH.NSK	: Windows 環境でのファイル名
/NC/_N_CH_GD2_ACC	: NC ドメイン
trans	: 変数 TransferState
2	: エリアチャンネル
2D	: モジュールタイプ MGUD
10	: トランザクションのタイムリミット 10s
/ACC/CH/MGUD/	: 変数名用のプレフィックスとして使用する任意のストリング

例 A-56　　チャンネル用の NSK ファイルを作成する

```
Sub Form_Load ()
    Label1.LinkTopic = "NCDDE|MMC2HW0"
    Label1.LinkMode = 2
    Label1.LinkExecute "MAP_ACC_NC(C:\MMC2\MGUD_CH.NSK, /NC/
    _N_CH_GD2_ACC ,trans,2,2D,10,/ACC/CH/MGUD/)"
End Sub
```

（注）NSK ファイルは、バイナリフォーマット (*.MAP) と ASCII フォーマット (*.NSK) の両方で生成されます。

NCDDE サーバの NSK ファイル内に受信する

この例で生成されたファイル MGUD_NCK.NSK and MGUD_CH.NSK を、NCDDE サーバ NCDDE311.NSK に受信します :

```
REM IMPORT ADDITIONAL USER VARIABLES
CALL(MGUD_NCK.NSK)
CALL(MGUD_CH.NSK)
REM
```

これにより、特別変数のクラスタリングが可能になります（参照、第 0 章「NCDDE をクラスタリングする」)。

NCK ユーザー変数にアクセスする :

次の 例では、NCK ユーザー変数 LIFTOFF_DIST が NCK からどうのように読み込まれるか示しています。

例 A-57　　NCK ユーザー変数 LIFTOFF_DIST を読み込む

```
Sub Form_Load
    CtlName1.LinkTopic = g_chNCDDEServiceName
    CtlName1.LinkItem = "/acc/nck/mgud/RUECKZUG"
    CtlName1.LinkMode = 2
    CtlName1.LinkRequest
    CtlName1.LinkMode = 0
End Sub
```

チャンネル別ユーザー変数にアクセスする :.

下の例の , BLF_OFFS_X, は、チェンネル別ユーザー変数の読み込み方法です。

例 A-58　　チャンネル別ユーザー変数 BLF_OFFS_X にアクセスする

```
Sub Form_Load
    CtlName.LinkTopic = g_chNCDDEServiceName
    CtlName.LinkItem = "/acc/ch/mgud/BLF_OFFS_X[u2]" 'fur 2.Kanal
    CtlName.LinkMode = 2
    CtlName.LinkRequest
    CtlName.LinkMode = 0
End Sub
```

（注）ユーザーデータの生成ならびに適用方法のついては、Installation and Start-up Guide /IAD/ および「Yaskawa Siemens 840DI ユーザーズマニュアル プログラミング編 基本説明書（NCSI-SP02-06）に詳しい説明があります。

変数のオンラインヘルプ

概要

NCK エリアでのデータの選択や定義に際しては、変数のオンラインヘルプが OEM プログラマをサポートします。これは、Windows の他のヘルプファイルと同様に構成され、同様の機能を持っています。変数のオンラインヘルプは、OEM パッケージ MMC とは独立しており、ディレクトリ HLP の BTSS_GR.HLP （ドイツ語テキスト）という名前のヘルプファイルに入っています。

ターゲットシステム

変数のオンラインヘルプは、MMC の OEM プログラミング以外でも使用できます: MMC や PLC プログラミング環境の NC-Var Selector のカスタマイズにも使用できます。

機能

変数のオンラインヘルプは、第 11 章に記載されていて、「Yaskawa Siemens 840DI 保守説明書 別冊付録 一覧表（NCSI-SP02-11）」で詳しく説明されている NCK 変数すべてに関する情報を提供することができます。

いくつかの記述レベルを使用して、特別変数に関する情報を入手することができます。

データエリアからスタートする：

Data area ⇒ Module ⇒ Variable ⇒ Example

あるいは、モジュールを使用してアルファベット順：

Module ⇒ Variable ⇒ Example

あるいは、機能 SEARCH (FIND) を使用してキーワードをサーチする

キーワードには次のようなものがあります :

変数の短い記述	例、Spindle Type
変数名	例、Variable Spindle Type
モジュールの短い記述	例、SSP （スピンドルステータスデータ）

データのコピー

画面上のヘルプトピックから一部をコピーして、それを他のファイルに挿入することができます。

特に、自分自身の OEM プログラムに変数のオンラインヘルプの例を挿入するのに役に立ちます。次の手順で行ってください :

メニュー 「Edit」を選択します

項目 「Copy」を選択します

マウスを使って必要なテキストを選択します

「Copy」をクリックします

他のアプリケーションに切替えます

テキストを挿入します

その他の機能

変数のオンラインヘルプを使用すると、次のことも行えます :

- トピックの印刷
- 各トピックに自分のコメントを挿入する
- ブックマークをつけて頻繁に必要とする情報を早く見つける

（注）変数のオンラインヘルプに関するコメントは、Windows ディレクトリのファイル BTSS_VAR.ANN (ANN は 付属書類を示す) に、ブックマークはファイル WINHELP.BMK (BMK はブックマークを示す) に格納されています。

トラブルシューティング

NCK/PLC との接続の中断

- 接続ケーブルを点検する
- MPI ドライバのインストレーションを点検する
- MMC.INI を点検する
- WINSTART.BAT
- S7DPMPI.INI

DDE 起動に応答しない

- Link - Topic を点検する
- Link - Item を点検する
- 変数が宣言されているか？特に PLC アクセスに対してはどうか？データモジュールが宣言されているか？

Hotlinks 作成のために Form Load に時間がかかる

- DCTL 制御を使用する
- 非同期 Hotlinks を作成する

最初の実行指令が機能しない

原因

指令によっては、NCDDE サーバは NC との接続がすでに存在すると思う

ソリューション

まず、NC 変数とのホットリンクを作成する

略語

略語	意味
DockPos	ドッキングポジション
FLR	ホストコンピュータ
FTP	ファイル転送プロトコル
MMC	マンマシンコミュニケーション
NCU	数値制御装置
PLC	プログラマブルロジック制御
RKS	MMC のコンピュータリンクソフトウェア
RPC	遠隔手続き呼び出し
TCP/IP	伝送制御プロトコル／インターネットプロトコル
TPS	トランスポートシステム
WPC	ワークキャリア
WZ	ツール

エラー番号

表 A.24 SINCOM エラー番号

番号	意味
-97	ERR_RESTARTING RPC リターン値、再始動中に RPC が到着した場合
-98	ERR_ORDERLIST_FULL RPC リターン値、SIMCOM 内部オーダーリストが一杯になった場合 →待機してコールを繰り返す
-99	ERR_NOT_SUPPORTED RPC リターン値、RPS がサポートされない場合 例 T_TPS_M がマシンに送信される
-100	ERR_WRONG_MACHINE RPC リターン値、マシン名が不正な場合
-110	ERR_WRONG_HOST RPC リターン値、ホスト名が不正な場合
-200	ERR_FUNCT_BUSY 1. RPC リターン値、同一の機能がすでに実行されている場合 2. R_REPORT_H を使用する、C_TPORDER_M が先の TPA が完了する前に到着した場合
-203	ERR_R_NC4WPC_M R_REPORT_H 、R_NC4WPC_M の後にエラーが発生している
-250	ERR_DEL_FILE R_REPORT_H, DH サーバ内のファイルが削除できない場合
-300	ERR_GET_FILE RPC リターン値 , ホストコンピュータからファイルを取得できない場合
-301	ERR_NC_RENAME R_REPORT_H 、長いファイル名を短くして 8.3 にする場合
-302	ERR_NC_FILESTATUS R_REPORT_H 、R_DATA_M (SFct=1) を使用してファイルデータを設定する場合
-310	ERR_DH_CREATE R_REPORT_H 、dh_create にエラーがある場合

-320	ERR_DH_GETFILE R_REPORT_H、エラーがある場合、プログラムがデータ管理にない場合
-400	ERR_PUT_FILE RPC リターン値、ファイルがホストコンピュータに転送されない場合
-500	ERR_DDE_CONNECT R_REPORT_H 、R_DDEDATA_M () に DDE 接続エラーがある場合
-510	ERR_DDE_POKE R_REPORT_H 、R_DDEDATA_M () に DDE ポークエラーがある場合
-600	ERR_TOOLDATADESCR R_REPORT_H、間違ったデータ構造番号を使ってツールリクエストをした場合
-610	ERR_GET_TOOLDATA R_REPORT_H 、ツールリクエストの場合、ツールデータの読み出しエラー
-700	ERR_TPS R_REPORT_H, TPS が TPA に発行した確認応答にエラーがある場合
-800	ERR_T_VAR_M R_REPORT_H 、T_VAR_M にエラーがある場合
-805	ERR_R_VAR_M R_REPORT_H 、R_VAR_M にエラーがある場合
-810	ERR_VARSET R_REPORT_H 、未知の変数セットがある場合 (SCVARSET.INI)
-820	ERR_VARSET R_REPORT_H 、変数セットにエラーがある場合
6003	ERR_FILE_NOT_FOUND 1. R_REPORT_H nach T_DATA_M (SFct=10), 名前 1 がおそらく間違っている

Yaskawa Siemens CNC シリーズ

本製品は,外国為替及び外国貿易法で規制する技術に該当します。従って,本資料を輸出する場合または使用に係る技術を非居住者に提供する場合には,同法に基づく許可が必要となりますので,通商産業大臣への役務取引許可申請が必要となります。

製品改良のため、定格、寸法などの一部を予告なしに変更することがあります。
この資料についてのお問い合わせは、当社代理店もしくは、下記の営業部門にお尋ねください。

製造
株式会社 安川電機　　シーメンスAG

販売

シーメンス株式会社
デジタルインダストリーズ
東京都品川区大崎1-11-1　ゲートシティ大崎ウエストタワー
TEL:03-3493-7565

技術的なお問い合わせ相談窓口
シーメンス株式会社
デジタルインダストリーズ　カスタマーサービス事業部
TEL:03-3493-7325　E-MAIL:industry.service.skk@siemens.com

・アフターサービス
シーメンス株式会社
デジタルインダストリーズ　カスタマーサービス事業部
TEL:03-3493-7325　E-MAIL:industry.service.skk@siemens.com