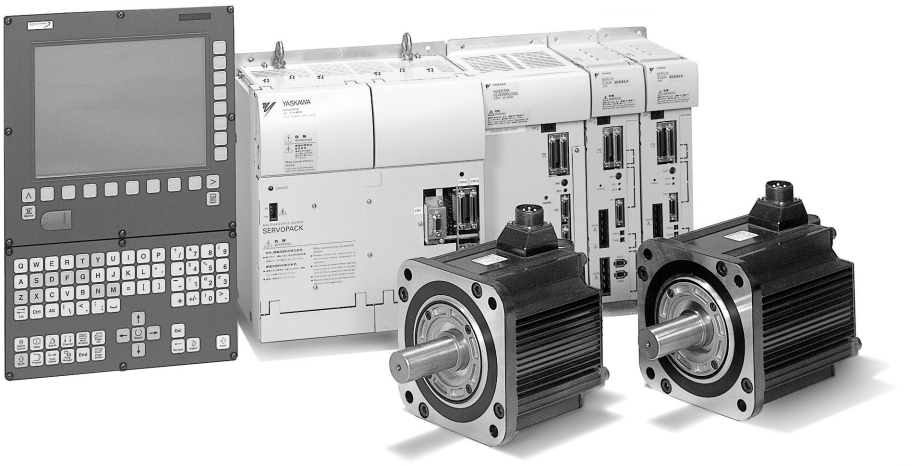
Yaskawa Siemens
numerical controls

Yaskawa Siemens CNC シリーズ

API取扱説明書
HMIプログラミングパッケージ
基礎編



資料番号　NCSI-SP02-13

安川シーメンス NC 株式会社はシーメンス株式会社に統合の後，2010 年 8 月より
シーメンス・ジャパン株式会社へ社名を変更いたしました。本書に記載の「安川
シーメンス NC 株式会社」などの社名に類する名称は「シーメンス・ジャパン株式
会社」へ読み替えをお願いします。

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ください。また，本文中に
840DI と言った表現が出て来ますが，830DI も意味していることがあるとご理解
ください。

1

1.1 オープンシステムとしての 840DI

ハードウェア要件

NC に対する顧客の要望には，以下の 3 つの技術的要素が関係します。

- ハードウェア要件
- オペレータインタフェースのデザイン（MMC）
- NC カーネル機能の拡張

オープンシステム

Yaskawa Siemens 840DI（以降 , 840DI と略します）のオープンシステムアーキテクチャには，上記の要素それぞれに適したプログラミングインタフェースが備えられています。

構成要素

- PLC モジュール：　入出力モジュール，FC モジュールなどを含め，最大で 3 つの SIMATIC-S7 ラックに入れられます。
- MMC 構成要素：　PCI/ISA アダプタ（計測機器，ネットワークボード用に 2 つのスロットを備えています)。フロッピディスクドライブ。PC カードアダプタ。
- NCK 構成要素：　デジタルおよびアナログ入出力の高速 NCK 環境
- シリアルリンク：　V.24 インタフェースまたは MPI（通信，ネットワーキング，PG，PC 用）

MMC デザイン

- PLC モジュール：　ランプ，ボタン，ミニコントロールパネルなどについて，PLC プログラムと連係しています。
- MMC 構成要素：　標準 PC ボード経由の駆動，操作，表示構成要素などについて，PCI/ISA アダプタと連係しています。
- MMC アプリケーション：　OEM パッケージ MMC と連係しています。

NC 機能の拡張

- PLC モジュール：　周辺装置および追加の基軸について，PLC プログラムとと連係しています。
- コンパイルサイクル：　高速な追加の機能を NC カーネルに与えるための最強の方法を提供します。

構成 /OEM ドメインの表

表 1.1 840DI の構成と OEM ドメイン

範囲	MMC	NCK	PLC
アプリケーション	OEM アプリケーション	コンパイルサイクル	FB，FC，OB
開発環境	業務用 PC での WINDOWS の仕様	SUN ワークステーションでの CADUL 社のツール群	PG または業務用 PC での SIMATICSTEP7 ツール
接続	WINDOWS API	イベント	SIMATICSTEP7 ツール
データへのアクセス	DLL	バインディング	DB
ストレージメディア	ハードディスク	PC カード	RAM
プログラミング言語	Visual Basic Visual C++	C++	STEP 7

1.2 ヒューマンマシンインタフェース（MMC）のオープンシステムアーキテクチャ

新機能

ヒューマンマシンインタフェース（MMC）のオープンシステムアーキテクチャは，完全にオープンな数値制御を OEM として工作機械メーカーに供給するための研究成果として生まれた，840DI の新機能です。このことは，ハードウェアだけでなく，ソフトウェアにも当てはまります。

オープンなハードウェア

PCU50 は，いくつかのハードウェアインタフェースを備えた標準的な PC です。

- 大容量記憶装置
- フロッピディスクドライブ
- ΠX カードアダプタ
- PCI/ISA-Box
- 2 つのシリアルインタフェースと 1 つのパラレルインタフェース

オープンなソフトウェア

PCU50 の標準ソフトウェアには，以下のものが含まれます。

- MS Windows 95 オペレーティングシステム
- 840DI 標準ユーザインタフェース

これらの標準的な採用ソフトウェアの下で，アプリケーションを実行できます。

MS-Windows 95 オペレーティングシステムを導入した標準的な PC では，OEM アプリケーションを自在にデザインできます。この構造の詳細については，「MMC 構成要素の基本」の章で説明されています。

1.3 各ユーザに関係する資料の部分

3.2 では，5 章から 10 章で使われている技術用語について説明されています。

資料のどの部分が関係しているかは，それぞれのアプリケーションに応じて異なります。

インテグレーション担当者

MMC 標準ユーザインタフェースにより，それぞれの要求を満たすことができます。しかし，他の目的で PC を使いたいこともあります。その場合，CAD プログラムなどのサードパーティアプリケーションを，未使用のソフトキーを使って任意に統合します。

修正担当者

MMC の動作方法を知りたいとは思わなくても，標準の機能を少し変更したいことがあります。標準のアプリケーションに対して OEM 呼び出しを使用してください。

インタフェース設計担当者

MMC 標準インタフェースには，必要な機能がない場合があります。その場合，独自のアプリケーションを追加すると，ソフトキーや操作パネルを使って，MMC 標準インタフェースのように操作することができます（たとえば，NC ユーザデータの構成や表示など)。

パワーユーザ

読者の顧客は**読者**が販売するマシンを購入します。840DI の標準ユーザインタフェースについて心配する必要はありません。制御のインタフェースは，貴社のアイデンティティによって決まります。

OEM パッケージのインタフェース

一群のインタフェースが提供されており，これらは，以下の要件に応じて，自在に組み合わせることができます。

NC データにアクセスするためのサーバ

NCDDE サーバ
アラームサーバ
データ管理サーバ

Regie

シーケンス制御

標準 NC アプリケーション（たとえば，マシン，パラメータなど）

次の表では，読者に関係するインタフェースをリストしています。太字で印刷されたものは，どの場合でも使う必要のあるインタフェースで，それ以外のものは，アプリケーションの機能拡張に応じて使うインタフェースです。

表 1.2 OEM パッケージのインタフェースを必要とする担当者とインタフェースの種類

統合担当者	修正担当者	インタフェース設計担当者	パワーユーザ
Regie	**Regie**	Regie	
	シーケンス制御	シーケンス制御	
	NCDDE サーバ	NCDDE サーバ	**NCDDE サーバ**
	アラームサーバ	アラームサーバ	アラームサーバ
	データ管理サーバ	データ管理サーバ	データ管理サーバ

1.4 開発環境

概要

顧客固有のアプリケーションを開発し，PCU50 上で実行するには，以下の装備が必要です。

- プログラム開発のための標準的な PC
- WINDOWS アプリケーション用の標準的なソフトウェアツール
- 顧客固有のプログラムを開発するための基本となる OEM パッケージ MMC
- PC ～ MMC 間でデータ転送するためのプログラム
- 顧客のアプリケーションをシステム環境でテストするための数値制御 840DI
- NC システムについての十分なノウハウを持ち，最新の高水準プログラミング言語に関して十分に経験を積んだソフトウェアエンジニア

ハードウェア

MMC アプリケーションは, MS-Windows 95 オペレーティングシステムが導入された標準的な PC 上で開発されます。
システム要件については，2 章の「システムのインストール」でリストされています。

ソフトウェア

.MMC 顧客のアプリケーションは，以下のような標準的なソフトウェア開発ツールを使って作成されます。

- Visual Basic VB 4.0 16 ビット版
- テキストを DLL へ追加するためのツール（たとえば，Visual C++）
- OEM パッケージ MMC

外部ツールで必要とされる拡張と，それらの使用上の推奨事項については，2 章「インストール」にリストされています。

テストツール

OEM パッケージ MMC を使用して WINDOWS の下で開発されたアプリケーションは，以下のように処理されます。

- PC 上でテストされる
- モデル環境でシミュレートされる
- MMC へ転送される

システムテストは，宛先ハードウェア上で，実際の運用条件の下で実行されます。宛先は，必要な構成が設定されている，完全な NC 制御になります。

ソフトウェアエンジニアに対する要件

MMC アプリケーションの開発者は，一般の WINDOWS プログラミングと NC 固有の条件の両方を考慮しながら開発します。そのため，両方の分野に精通し，ソフトウェアプロジェクトの組織で経験を積んでいなければなりません。

PC での経験

特に以下のことが求められます。

- PC ベースでの開発プロジェクトをやり通した経験
- DOS および WINDOWS プログラミングの高度な経験
- オブジェクト指向による問題解決およびプログラミングについての知識 •Visual Basic Professional における高度な経験
- リソースエディタ（たとえば，Visual C++）での経験

NC についての知識

要件は，基本的に問題についての技術な要請に応じて異なってきます。

- NC のソフトウェアアーキテクチャについての知識
- 必要なデータについての精通度
- 特有のテクノロジーを使う，ユーザフレンドリなオペレータインタフェースの設計の経験

1.5 サポート

OEM ユーザは，幾種類ものサポートを受けられます。

- トレーニング
- 電話相談サービスによるソフトウェアサポートサービス
- ハードウェア／ソフトウェア開発サービス

トレーニング

グレード別トレーニングでの 840DI 概念では，特に以下の点を説明しています。

- NC 制御の運用とプログラミング
- 運転と保守の設定
- OEM アプリケーション

これにより，数値制御の構造や，ベースシステムをそれぞれの目的に適応させる際の MTB のさまざまな可能性について，理解を深めることができます。

ソフトウェア更新サービス

ユーザは，ソフトウェア更新サービスを利用して，次のリリースのサポートを購入します（更新およびアップグレード版）。

- **更新**：更新とは，ソフトウェアバージョンがより上位のバージョン番号になることです。これには，バグ修正が含まれます。たとえば，リリース 3.4 から 3.6 への更新のように使います。
- **アップグレード**：アップグレードとは，ソフトウェアにより上位のリリース番号が付けられることです。これには，新機能が備えられます。たとえば，リリース 3.6 から 4.2 へのアップグレードのように使います。
- **電話相談サービス**：OEM パッケージの電話相談サービスは，ソフトウェア更新サービスの一部です。

ソフトウェア更新サービスは，実際の製品リリースだけで終わる場合もあります。有効期間は 12ヶ月ですが，終了 4 週間前までに取り消さなければ，さらに 12ヶ月延長されます。

電話相談サービス

OEM パッケージ NCK を使う場合，顧客は電話相談サービスのサポートを 3ヶ月間受けられます。この期間については，さらに 12ヶ月のソフトウェアサポート契約を結ぶことにより，延長することができます。

ハードウェアおよびソフトウェアサービス

まず，顧客の要件を以下のように分類します。

- ハードウェア補足事項
- MMC オペレータのインタフェースデザイン
- NC 機能の拡張

次に，以下の実現方法について決定します。

- PLC プログラミング
- MMC での WINDOWS 開発
- コンパイルサイクルでの NCK の OEM 拡張

その後，

- 以下のいずれかにより実行します。

自社スタッフ

- 外部業者に発注

当社の開発部門へアウトソーシングすると経済的です。経験のあるプログラマやシステムスペシャリストが，スケジュール厳守かつ定額の料金で開発を行います。

1.6 表記上の規則

このマニュアルには，本文の説明に加えて，プログラムの例やプログラムの抽出が収められています。さらに，特定のキーを押すかキーの組み合わせで入力する，ユーザ用の命令も収められています。

本書のこれらの構成要素を強調するため，それぞれの構成要素は異なるフォントや字体で記されています。

このような印刷上の規則は, プログラミング言語 'Visual Basic' および 'VisualC++' のマニュアルで使われている規則に従っています。これらの規則については，「*Document Conventions*」の「Introduction」でリストされています。

表 1.3 このユーザーズマニュアルにおけるテキスト構成要素の表記

例	意味および説明
Sub，If，ChDir	プログラミング言語の予約済みキーワード。太字で，最初の文字が大文字。
BackColor，Click，Debug	特性，イベント，特殊オブジェクトの名前。最初の文字が大文字。
setup	ユーザによるキーボード入力。太字。
event-driven	テキストの中。用語が最初に出現するときに，テキスト本文をイタリックにする。
eventname	プログラムの中。ユーザが与える情報のプレースホルダ。イタリック。
[expression list]	プログラムの中。角括弧内は任意指定項目。
{ While|Until }	プログラムの中。複数の項目から選択する。大カッコ。
`Syntax error`	ユーザー定義変数およびエラーメッセージのタイプ。Courier。
`Sub HelloButton_Click ()Read-out.Text = "Hello!"End Sub`	プログラムコード。Courier タイプ。
`Readout.Text = "This_ should be typed all in_ one line"`	プログラムコードはすべて，コードウィンドウの 1 行内に入力する。行を連結するときは，_ の識別アイコンを使う。
CONSTANT.TXT	ファイル名。大文字。

ENTER，CTRL+R（同時に押す）	キーの名前およびキーの組み合わせ。小さな大文字。
ALT+F1 ALT+F1	キーの組み合わせ。キー ID の間に + - 符号。

1.7 過去のバージョンからの変更事項

1.7.1 リリース 3.4（1996 年 8 月）

概説

新しい機能および変更された機能 :

- 文書
- デリバリボリューム
- 拡張されたデリバリボリューム（新しいソースファイル）
- 変数のヘルプ機能
- 変更されたインストール手順
- リリース 3.1 から 3.4 への更新に関する情報
- 新しいスタートアップルーチン
- DLL 用の拡張域
- Regie の属性：AccessLevel
- 新しい Regie コントロール
- Regie のテキスト
- Regie の構成用ツール
- Regie のダイナミックリンクライブラリ
- シーケンス制御の一時ファイル
- アクセス権付きのソフトキー
- 拡張されたメニューツリー生成プログラム
- 拡張されたドメイン間のコピー機能
- ドメインサービスに関する追加情報
- 変更されたデータ管理サーバ
- 変数：stopCond
- 新しい変数：stopCondPa
- 変数：safeFctEnable，パラメータの変更
- グローバルユーザ変数，索引範囲の変更
- ローカルユーザ変数，索引範囲の変更
- 追加の PLC 変数
- コントロール：TABELLE.VBX• 変数：load，データ形式の変更
- 変数：axisFeedRateUnit，修正
- 軸の状態データ，値の限定
- PI サービス：_N_F_PROT，パラメータ

文書

文書が改訂され，分かりやすい編成になるよう構成が一新されました。

すべての章：.INI ファイルの節はアルファベット順にソートされています。

5 章： システムの構成が追加されました。

6 章および 7 章： 5 章（Regie）および 6 章（シーケンス制御）はそれぞれ 6 章と 7 章に移りました。キーボードサーバに関する情報（これまでは 7 章に記述されていた）は，対応する節（Regie およびシーケンス制御）に分けて掲載されています。

11 章： 変数の説明が短くなりました。現在では詳細な情報は，/LIS/ の資料 ‘ リスト ’ から入手するか，またはオンラインヘルプを使用して，対応するプロジェクトに直接情報を引き渡すことができます。変数のオンラインヘルプは，8 章で説明されています。

11.12 章： PI サービスの構造が新しくなっており，サービスの種類に従って分類されています。

11.14 章： NCDDE アラームが大幅に更新されました。

11.18 章： ソフトウェアとハードウェアに関する情報を含む新しい章です。

デリバリボリューム

OEM パッケージ MMC のデリバリボリュームは，以下の表に示されている 4 つの構成要素から成ります。

表 1.4 OEM パッケージ MMC リリース 3.4 のデリバリボリューム

ソフトウェア	ボリューム	ディスクの数
MMC 102 基本システム	NC または外部 PC で実行できる MMC 102 基本システム	10
OEM パッケージ	ソース，例，シーケンス制御，変数のオンラインヘルプ	3
WIN 32S	WINDOWS 3.11 用 32 ビットエクステンダ	2
リモート診断	リモート診断用のホストおよびターゲットソフトウェア	2

すべての構成要素を完全にインストールするには，ハードディスク上に約 64 MB の記憶域が必要です。

拡張されたデリバリボリューム

現行のデリバリボリュームには,‘アラーム診断’および‘パラメータ’という,2 つの標準的なアプリケーションのソースファイルが含まれています。

重要:

これらのソースファイルが備えられた理由は，OEM のお客さまが既存のアプリケーションに用いる独自の図やメニューツリーを引き続き使用できるようにするためです。今後のリリースでは，既存のアプリケーションの内部インタフェースの互換性は保証できないので，この種のインタフェースは使用しないでください。

変数のヘルプ機能

デリバリボリュームに変数のオンラインヘルプが追加されました。一般的にこのヘルプは，MMC 102 のプログラミング，MMC 100 の構成，または NC 変数セレクタを使用した PLC のプログラミングに使用されます。変数に関する情報には，いろいろな検索基準でアクセスすることができます（WINDOWS ヘルプの使用法に精通していれば)。このヘルプから，変数の定義をプロジェクトに直接引き渡すことができます。頻繁に使用する変数のブックマークを定義したり，独自のコメントを追加したりすることができます。8 章では，この特長の拡張と用途について説明されています。

インストール

OEM パッケージ MMC リリース 3.4 のインストール手順は，これまでの手順とは違います。2 章には，パッケージのインストールと撤去に関する詳細な指示があります。

リリース 3.1 から 3.4 への更新

リリース 3.1 で開発したアプリケーションは，変更を加えなくてもリリース 3.4 で実行できます。しかし，アプリケーションを編集してから，新しいリリース 3.4 のシーケンス制御を使用してコンパイルした場合は，まず Visual Basic プロジェクトの .MAK ファイルを編集しなければなりません。編集するには，WINDOWS の「メモ帳」エディタなどを使用して .MAK ファイルを開き，

単にエントリ "REGIECTL.VBX" を "RECTLP32.VBX" に置き換えてください。

重要 : これはグローバルユーザ変数に適用されます。

プロジェクトでグローバルユーザ変数を使用している場合は，11 章に記述されているとおり，グローバルユーザデータ GUD（GD1 ～ GD9）の指標の範囲が以下のように変更されていることを考慮してください。

列指標 = 変数の指標

これは先頭を 0 にするために使用されます。リリース 3.2 以降では先頭が 1 になっています。

重要 : これはローカルユーザデータに適用されます。

プロジェクトでローカルユーザ変数を使用している場合は，11 章に記述されているとおり，ローカルユーザデータ LUD の指標の範囲が以下のように変更されていることを考慮してください。

行指標 = 変数の指標

これは先頭を 0 にするために使用されます。リリース 3.2 以降では先頭が 1 になっています。

新しいスタートアップルーチン

リリース 3.1 以降，操作コンポーネント MMC 102 のスタートアップとソフトウェアローダが新しくなっています。詳細については，Installation & Start-up Guide（IA）で説明されています。このルーチンはスタートアップ時に呼び出されます。テキスト

"Starting DOS ..."

が表示されたら，［6］キーを押して，開いたメニューから選択してください。

DLL 用の領域の拡張

現リリースの Regie は 2 倍の数の DLL を管理できるようになりました。

- 32 個の補助アプリケーション
- 32 個の領域アプリケーション
- **64** 個のシステムのダイナミックリンクライブラリ（DLL）
- **64** 個の MMC のダイナミックリンクライブラリ

Regie の属性：AccessLevel

新しい属性 ‘AccessLevel‘ が，セクション [TaskConfiguration] に追加されました。

この属性を使用すると，Regie のタスクのアクセス許可レベルを設定できます。リリース 3.2 以降，Regie を取り扱うソフトキーにアクセス権が割り当てられるようになりました。

表 1.5 Regie の操作用のソフトキー

Machine	Parameter	Program	Services	Diagnosis	Set up

デフォルト値は AccessLevel := 4 です。

表 1.6 には有効なアクセス許可のレベルがリストされています。デフォルト値は背景が網掛けになっています。

表 1.6 レベルのアクセス許可

アクセス	必要なもの	ユーザグループ
S0	システムパスワード	当社
S1	MTB パスワード	工作機械メーカー
S2	保守パスワード	セットアップ／サービス担当員（工作機械メーカー）
S3	ユーザパスワード	特権ユーザ（企業内サービス）
S4	キースイッチ位置 3	プログラマ
S5	キースイッチ位置 2	訓練を積んだオペレータ
S6	キースイッチ位置 1	オペレータ
S7	キースイッチ位置 0	中級程度の技術のオペレータ（NC 開始／ NC 停止，操作パネル）

例 :

[TaskConfiguration]

Task2 = Name := ib, AccessLevel := 2

セットアップ担当員だけが，MTB のパスワードを入力して，領域アプリケーション ‘set-up’ にアクセスできます。

新しい Regie コントロール

Regie 用の新しいコントロール（RECTLP32.VBX）が開発されました。このコントロールにはコマンドインタープリタチャネルが含まれています。6 章に詳細が記述されています。

Regie のテキスト

Regie の言語依存テキストは，言語 DLL に格納されなくなり，<language.INI> というファイルに格納されるようになりました。ASCII エディタを使用して，このファイルを編集できます。6 章にその構造に関する情報が記述されています。

Regie の構成用ツール

構成ツールを使用して，REGIE.INI のセクション [TaskConfiguration] を構成できます。

このツールは，コントロール上のサービスメニュー内，

および MMC-OEM パッケージのインストールディレクトリ内のパス ..¥aconfig¥aconfig.exe にあります。

この構成ツールを使用して，

- 個々のタスク用にソフトキーにラベルを付ける方法を指定できます。対応するファイルのある（...¥MMC2¥LANGUAGE¥RE_XX.INI）すべての言語に，この操作を実行できます。
- ファイル REGIE.INI のパラメータ（name，CmdLine，DosBox，PreLoad，TimeOut，HeaderOnTop，TerminateTasks，および AccessLevel）を編集できま

す。
- 既存のエントリを移動したり除去したりできます。このツールの使用法に関するオンラインヘルプがあります。

注 : この構成ツールは，リリース 3.2 から始まる初期設定ファイル REGIE.INI の編集に適しています。

Regie のダイナミックリンクライブラリ

REGIE.DLL の機能は大幅に拡張されました。現在，REGIE.DLL には以下のことを実行する機能があります。

- タスク転換
- 非表示領域へのタスク転換
- 非表示領域への即時タスク転換
- 画面制御
- ロック
- コマンド行
- その他

6 章には，これらの特長の概要と詳細情報が記述されています。

シーケンス制御の一時ファイル

ソフトウェアパッケージリリース 3.4 をインストールすると，<drive>:¥ALTMP¥*.$$$ に格納されているシーケンス制御の一時ファイルが自動的に削除されます。

アクセス権付きのソフトキー

現在では，個々のソフトキーにアクセス許可レベル（Access Level）を割り当てることができます。

追加属性 Access Level が状態マトリックス PROGNAME.ZUS に追加されました。この属性を使用して，このソフトキーのアクションに関するアクセス権を設定できます。

無効項目のプリセット値は，Access Level := 5 です。項目がない場合（既存のメニュー構成を引き渡す場合），AccessLevel:=7（アクセスレベルなし）が予約済みです。

表 1.7 には有効なアクセス許可のレベルがリストされています。デフォルト値は背景を網掛けにしています。

表 1.7 8 レベルのアクセス許可

アクセスレベル	必要なもの	ユーザグループ
0	システムパスワード	当社
1	MTB パスワード	工作機械メーカー
2	保守パスワード	セットアップ／サービス担当員（工作機械メーカー）
3	ユーザパスワード	特権ユーザ（企業内サービス）
4	キースイッチ位置 3	プログラマ
5	キースイッチ位置 2	訓練を積んだオペレータ

6	キースイッチ位置 1	オペレータ
7	キースイッチ位置 0	中級程度の技術のオペレータ（NC 開始／ NC 停止，操作パネル）

記号名を使用して，アクセスレベルとユーザクラスの定義と管理を実行することもできます。

メニューツリー生成プログラム

メニューツリー生成プログラム（Screen Control Design）が改訂されており，シーケンス制御の新しい機能に適合するようになっています。

追加の機能 :

- 改良されたグラフィカルユーザインターフェース
- store/drop ではなく store/unload による Z フラグのマーク付け
- ソフトキーをロックするための入力フィールド
- ソフトキーを切り替えるための入力フィールド
- ソフトキーのテキストが切り替わる状態のマーク付け

変更された機能 :

- ソフトキーのバーの除去 : 現在では，ソフトキーのバーを削除する機能として［DELETE］特殊ボタンが備えられています。
- ［RECALL］キー : 再呼び出しキーに後続の状態を構成していない場合，これ以前の状態は継続されません。
- 状態ごとに 1 つ以上のソフトキーテキスト（スペース 1 つなど）を挿入する必要はなくなりました。

■ シーケンス制御の新しい機能

現在では，新しい関数を使用してシーケンス制御を十分に実現できます。以下のことを実行するための関数が合計 43 あります。

- 状態制御
- ヘルプ機能
- MDIChild 関数
- ソフトキーの解放／ロック
- ソフトキーテキスト機能
- テキスト表示
- モーダルウィンドウ関数
- アクション関数（状態マトリックスの動的変更）

7 章に詳細が記述されています。

ドメイン間コピー関数の拡張

SW リリース 3.3 以降，拡張されたコピー関数を使用できます。ホットリンクの活動が非常に多い場合に，単一の PI コマンドや転送コマンドよりデータのスループットは大きくなります。8.3.2 に，この新しい関数が説明されています。

ドメインサービスに関する追加情報

8 章で説明されているように，MAP_ACC_MC コマンドが関係しています。パラメータ Win File に拡張子 .NSK のファイルを指定すると，ドメインサービスにより ACC ファイルだけでなく，対応する LINK コマンドを含む NSK ファイルが生成されます。

データ管理サーバ

データ管理サーバが改訂されました。

■ 変更された機能と拡張された機能

- 現在では，データ管理サーバを使用して，DOS ファイルシステム全体のパスを使用できるようになりました。以下のように，このパスの先頭はディスクドライブの指定でなければなりません。
  A:¥TEST.ARCx，C:¥TMP¥TEST.XYZ，K:¥TMP¥NEU.TXT
  これらを create および copy プロシージャに，それぞれソースとして，また宛先として使用することができます。
- 指定したフロッピーディスクにアクセスする際に，データ管理サーバによりディスクスロットが閉じているかどうかチェックされます。エラーの場合は，新しいエラーコード 125 が戻されます。
- copy プロシージャの宛先の先頭がディスクドライブの指定である場合は，データタイプは DOSFILE か DOSDIR のどちらかに設定されます。
- 現在では，関数 copy を使用すると，NC のアクティブファイルシステムのロード／アンロードを実行する際に，そのアクティブファイルシステムをパスにすることができます。
  例 :
  load　　copy ¥NCINI.MDN¥UFR.DIR¥CH_UFR.INI–m¥INITIAL.INI -nf *
  ¥unload copy¥CH_UFR.INI -n¥NCINI.MDN¥UFR.DIR¥CH_UFR.INI –mf
  このプロシージャの間，アクセス権は考慮されません。
- PLC データの取り扱いが以下の点について拡張されました。
  PLC ファイルを DOS ファイルにコピーします。
  copy ¥PLC.DIR¥0800001 -n C:¥TMP¥TEST.TXT -m DOSFILE
  DOS ファイルを PLC ファイルにコピーします。
  copy C:¥TMP¥TEST.TXT -m ¥PLC.DIR¥0800001 -n DOSFILE
  create ¥PLC.DIR 0800001 --- DOSFILE -n C:¥TMP¥QUELLE.TXT
- 現在では，関数 **Best_Datatype** を使用して，以下の呼び出しを実行できます。
  Best_Datatype test mpf ¥wks.dir¥welle2.wpd –n
  この場合，データタイプは，このディレクトリ内で有効なデータタイプに従って設定されます。

新しい関数

- 関数 **activate2**: **activate** と全く同じ働きをしますが，ソースファイルが削除されません。
- 関数 **get_attributes2**: **get_attributes** と同じ呼び出しで，**リスト**から得られる結果と同じになります。例 : topic: get_attributes item: \wks.dir -n#100#WKS WKS work pieces DIR WKS 0 812388942 DM 77770#

変数 stopCond

11 章で説明されているように，**NC の保留状態**を表す変数 **/Channel/State/stopCond** の有効値として，以下の値が追加されました。したがって，現在戻される可能性のある値は 50 になりました。

25 "wait for gear stage change"
26 "wait for position control"
27 "wait for threat cutting"
28 "wait"
29 "wait for punch"
30 "wait for safe operation"

以下の値は予約済みです。

31 "wait 31"
～
50 "wait 50",

対応するテキストは必要な時点で指定されます。この値の範囲が拡張されたことを考慮しなければなりません。

新しい変数 stopCondPar

以下のパラメータは，**NC の保留状態（/Channel/State/stopCond）**変数の追加パラメータです。
/Channel/State/stopCondPar
このパラメータは，4 つの保留状態の考えられる理由を示し，主軸に対するそれぞれの軸を参照します。
以下にリストされていない変数 stopCond の値はすべて，変数 stopCondPar = 0 です。

値変数 stopCond 変数 stopCondPar

12"wait: axis enable missing" 軸番号

15"wait: for spindle" 主軸番号

22"wait: spindle enable missing" 主軸番号

23"wait: axis feed override 0" 軸番号

変数 safeFctEnable

SISITEC 軸の変数 safeFctEnable には，機械データ MD 36901_SAFE_FUNCTION_ENABLE の内容が入ります。

グローバルユーザ変数

11 章に記述されているとおり，グローバルユーザデータ GUD（GD1 ～ GD9）の範囲が以下のように変更されています。

列指標 = 変数の指標

これは先頭を 0 にするために使用されます。リリース以降，先頭が 1 になっています。

ローカルユーザ変数

11 章に記述されているとおり，ローカルユーザデータ GUD（GD1 ～ GD9）の範囲が以下のように変更されています。

行指標 = 変数の指標

これは先頭を 0 にするために使用されます。リリース以降，先頭が 1 になっています。

PLC 変数

11 章の PLC 変数の説明に，以下の項目が追加されました。

/Float 順次格納されている 4 つのバイトに対して，
inter- 浮動小数点数として事前設定されているものとしてアクセスする。
/Directory /Hierarchy1 すべてのモジュール形のモジュールリストにアクセスする。
Directory /Hierarchy2 1 つのモジュール形のモジュールリストにアクセスする。
Directory /Hierarchy3 特殊なモジュールのモジュールリストにアクセスする。

追加パラメータなし : PLC 変数の巡回サービスのサイクルタイムは，500 ミリ秒です。
追加パラメータ : -FAST ▼この変数と，同一クラスタの他のすべての変数のスキャン時間は，両方とも 100 ミリ秒です。

コントロール Tabelle.VBX

コントロール TABELLE.VBX は，今後は使用しないでください。

変数のロード（サービスドライブ）

ドライブのロードを操作する 2 つの変数
driveHSA/State/Load および
DriveVSA/State/Load
のデータタイプが，unsigned から float に変更されました。それに伴い，最小値 = 0.0，最大値 = 100.0 になります。

変数 axisFeedRateUnit

変数の値（モジュール SEGA および SEMA における突っ込み率の単位）
axisFeedRateUnit
には，以下の意味があります。

0 = mm ／ min
1 =° ／ min（mm ／ rev ではない）
2 = inch ／ min
3 = 意味なし（inch ／ rev ではない）

軸の状態データ，限界

以下のモジュール

- SMA 機械座標系による機械軸の状態データ
- SEMA 機械座標系による機械軸の拡張状態データ
- SGA ワーク座標系による形状軸と追加軸の状態データ

- SEGA ワーク座標系の形状軸と追加軸の拡張状態データ

の変数の行数は，0 ～ M_AX_NUM に達するのではなく，0 ～ **numMachAxes**（モジュール CY に格納される）に達します。

PI サービス _N_F_PROT

PI サービスの _N_F_PROT（ファイルに対するアクセス権を割り当てる）関数を**実行**するためのアクセス権には，以下の 2 つのレベルしかありません。

0 : 実行するファイルを選択できません。または
7 : 実行するファイルを選択できます。

1.7.2 リリース 4.2，1997 年 8 月

概説

新しい機能および変更された機能 :

- デリバリボリューム
- インストール
- オペレーティングシステム
- リリース P3.6 から P4.1 への更新に関する情報
- NC データ用の 32 ビットサーバ
- OEM の例
- PC 上での " シミュレーション "
- コントロール Tabelle.VBX
- 文書
- NC データおよび PI サービスのオンラインヘルプ
- 新しい PI サービス
- ソースコード
- メニューツリー生成プログラム
- 新しいディレクトリ構造
- 新しい構成ツール
- OEM ソフトキー : カスタマインタフェース
- シーケンス制御の変更
- シーケンス制御の新しい機能

デリバリボリューム

OEM パッケージ MMC のデリバリボリュームは，下記の表でリストされている 3 つの部分から成り立っています。

ソフトウェア	特長	ディスク
MMC 102 標準 SW	MMC 102 標準 SW	14
OEM ディスク	例，シーケンス制御	2
リモート診断	リモート診断用のホストおよびターゲットソフトウェア	2

インストール

最初に MMC102 標準 SW，次に OEM ディスクをインストールして，OEM パッケージをインストールします。この作業を行うために，Windows で開始されるセットアッププログラムが最初のディスクに含まれています。

必要であれば，リモート診断をインストールできます。

オペレーティングシステム

パッケージ 4.1 以降の MMC102 ソフトウェアでは，オペレーティングシステムとして Microsoft Windows 95 を使用します。

重要:

上記にかかわらず，Regie（タスク管理）は，まだ 16 ビットアプリケーションしか扱えません。つまり，OEMFRAME を使用する場合でも，開始できるのは 16 ビットの操作だけです。

SW 3.6 から SW 4.2 への更新

これは，リリース 3.6 で作成された .EXE ファイルとのバイナリ互換性を保持するためです。つまり，MMC パッケージ 3.6 用に作成されたアプリケーションが，リリース 4 のインストール後も，制御が正常に実行されるようになっています。

残念ながら，これはすべてのアプリケーションで保証されているわけではありません。ご使用のアプリケーションで問題が生じた場合，OEM パッケージ 4.2 のシーケンス制御を再生成する必要があります。これを行うには，Visual Basic 4.0 16 ビット版を使用しなければなりません。

NC データ用の 32 ビットサーバ

標準ソフトウェアの 16 ビット版として，NC データ用のサーバ（NCDDE サーバ，アラームサーバ）が提供されています。32 ビット版が必要な場合（たとえば，SIMATIC プログラミング機器を操作する場合など）は，担当の販売業者に連絡してください。

OEM の例

以下の OEM の例が 2 つ追加されました。

- 例 11: シーケンス制御の機能
- 例 12: いくつかの非同期ジョブの順次実行の例

シミュレーション

原則として，PC 上での標準構成要素のシミュレートにはいくつかの制限があります。これは，必要な変数がすべて使用できるわけではないためと，いくつかの機能が実行されないためです。

コントロール Tabelle.VBX

コントロール "Tabelle.VBX" はなくなりました。代わりに，コントロール "grid.VBX"

を使用します。

文書

文書は完全に改訂され，ソフトウェアのリリース 4 で重要な機能が追加されました。

NC データおよび PI サービスのオンラインヘルプ

新しいパッケージでは，OEM での使用を目的としてリリースされている，操作パネルのインタフェース（BTSS）のすべてのデータに関するオンラインヘルプのほかに，PI サービスに関するオンラインヘルプが提供されています。BTSS 変数に関しては，以下のプログラマ／構成担当者を OEM ユーザとみなすことができます。

- MMC102-OEM のアプリケーションのプログラマ
- OP030 の構成担当者
- NC-Var-Selector を使用する PLC プログラマ

PI サービスのオンラインヘルプには，パラメータの説明および各サービスの例が含まれています。

BTSS 変数のオンラインヘルプには，ユーザ /LS/ の 4 章「変数」にある情報が含まれています。それに加えて，パッケージではすべての BTSS 構成要素の例が提供されており，OEM プログラマがコピーできるようになっています。

以降の本文では，さまざまなプログラミング環境（OEM-MMC，OP030，および NCVarSelector）に関して，BTSS 構成要素という表現を使用しています。これは，それらがすべて操作パネル（BTSS）に接続できるためです。

BTSS 変数のオンラインヘルプの使用

オンラインヘルプのレイアウトは，特に PC 上での使用を想定して設計されています。（以前のバージョンの紙文書のような）リストはありませんが，BTSS 変数ごとに完全な情報が画面に表示されます。

オンラインヘルプでは Windows の標準機能しか使用しないため，OEM ユーザは特別な専門知識を必要としません。PC システムは特別な構成を必要としません。オンラインヘルプは，Windows を使用しているあらゆる PC で実行できます（ドイツ語版および英語版の Windows でテスト済みです）。

オンラインヘルプは，独立した Windows アプリケーションとして，OEM 構成パッケージと並行して実行されます。OEM パッケージとオンラインヘルプの間にリンクは存在しません。

以下に，オンラインヘルプの標準機能をリストします（ほかのオンラインヘルプで OEM ユーザが習熟しているのと同様のものです）。

- 現在表示されているトピックを印刷する。
- 各トピックに対して自分の注釈を追加する。
- 表示されているトピックから一部をコピーし，別のファイルに貼り付ける。これは，プログラミング例を自分の OEM プログラムに使用する場合に便利です。
- 頻繁に使用するトピックをすぐに見つけるために，ブックマークを定義する。

- オンラインヘルプ自体に，機能（ブックマーク，印刷，コピーなど）について説明するオンラインヘルプを含める。
- 説明に関するいくつかのレベルをスキップしていき，各変数に関する情報へジャンプする。

データ域 -> データモジュール -> 変数 -> 例

キーワードで直接検索することもできます。多くの場合，以下のものをキーワードとして使用します。
データモジュールの略称
変数名（検索ウィンドウ内で "variable.." として表示されます）
変数の短い説明

新しい PI サービス

PI サービスのオンラインヘルプが使用できるようになりました。

ソフトウェアリリース 4.2 以来，以下の PI サービスが使用できるようになりました。
_N_CRCEDN ：エッジ番号を指定して，カッティングエッジを作成します
_N_DELECE ：カッティングエッジを削除します
_N_TMCRTO ：ツール管理で，指定した名前，ツール番号，および Duplo 番号により，ツールを作成します
_N_TMFPBP ：指定したマガジン内の空いている場所を検索します。場所のタイプ，検索するマガジン，およびツールが必要とするスペースの量を指定することができます。
_N_TMGETT ：与えられたツール ID および Duplo 番号から，T 番号を判別します

記号アドレッシング

今後は PI サービスの記号アドレッシングを使用してください。

/NC および **/PLC** は，/0d0d などよりも読みやすさが勝ります。

ソースコード

以前のソフトウェアリリースでは，アプリケーションマシン，パラメータ，および診断のソースが提供されており，OEM ユーザがそれらのアプリケーションのソフトキーに，独自の機能を割り当てられるようになっていました。これには，MMC ソフトウェアを更新するたびに，それらのアプリケーションを再コンパイルしなければならないという欠点がありました。

この問題を解決するために，MMC102 SW 4.2 では OEM ソフトキーが導入されました。

メニューツリー生成プログラム

メニューツリー生成プログラムは，SW 3.2 以来変更されていません。

新しいディレクトリ構造

カスタマアプリケーションの扱いを簡単にするために，MMC102 のディレクトリ構造が拡張されました。

これまでは，MMC のファイルはディレクトリ ¥MMC2 にありました。このディレク

トリには OEM アプリケーションも入っており，ini ファイル（regie.ini など）はここで変更されました。

MMC を更新すると，¥MMC2 ディレクトリが削除された後に再インストールが行われるので，OEM 固有の変更がすべて失われました。

この点を改良するために，SW 4.2 以降では以下の 3 つの追加ディレクトリが作成されるようになりました。

- ADDON（Autoturn や SINDNC などの当社提供アプリケーション用）
- OEM（OEM アプリケーション用）
- USER（.ini ファイルやアラームテキストなどの変更用）

¥MMC2 ディレクトリはまだ存在します。**しかし，SW4.2 以降では書き込み保護されています。**更新中に削除される点は変わりませんが，OEM 固有の変更はほかのディレクトリに入れられるようになったので，それらが失われることはありません。

OEM アプリケーションの開発者は，それらのアプリケーションを前述の新しいディレクトリのいずれかにインストールできるようにし，対応する SETUP アプリケーションのデフォルトパスがその OEM ディレクトリを指すように注意しなければなりません。

ユーザ固有の NC データを組み込むために，システムはファイル user.nsk を呼び出します。

新しい構成ツール

MMC102 SW4.2 用に，新しい構成ツールが開発されました。

このツールを使用して，以下のことが行えます。

(1) ファイル REGIE.INI のセクション [TaskConfiguration] の構成
(2) MMC102 の書き込み可能な .ini ファイルの編集

構成ツールを使用して行った変更は，新しいディレクトリである ADDON，OEM，または USER のいずれかに入っている，対応するファイルに直接保管されます。

このツールは，サービスメニューのコントロールや，PC 上の OEM パッケージ MMC のプログラミンググループ内にあります。

これは以下の用途で使用できます。

- タスクごとのソフトキーラベルの指定そこでは，ファイル mmc.ini 内で定義されているすべての言語が選択できるようになっています。
- regie.ini のパラメータ（name，CmdLine，DosBox，Pre-Load，TimeOut，HeaderOnTop，TerminateTasks，および AccessLevel）の編集
- Regie への新しいタスクの追加
- 既存のエントリの移動および削除

構成ツールでは，可能な操作についてオンラインで説明します。

注 : この構成ツールは，SW 4.2 から始まる初期設定ファイルの編集に適しています。

OEM ソフトキー : カスタマインタフェース

840DI で提供されている各アプリケーションに関して，対応する .ini ファイル内でエントリを作成し，追加の状態を構成することができます。これにより，アプリケーション自体を変更することなく，事前に決められている場所でユーザピクチャを挿入することができます。

注 : この機能については，4.4 で説明されています。

.INI ファイルへのアクセス

ディレクトリ構造が新しくなったため，.INI ファイルは別のディレクトリに入れられている場合があります。最も優先順位の高いエントリにアクセスするには，WINDOWS-API の関数を使わずに，ライブラリ **AB16.DLL**（シーケンス制御の一部）の以下の関数を使用する必要があります。

AL*GetPrivateProfileString*

AL*GetprivateProfileInt*

AL*WritePrivateProfileString*

.INI ファイルの基本ディレクトリが基本 MMC ディレクトリ（通常は L:¥MMC2）とは異なる場合，対応する標準 C 関数が呼び出されます。

関数 ALGetPrivateProfileString および ALGetprivateProfileInt で値を読み取る場合，MMC パスで始まる目的のエントリを見つけるために，ディレクトリ ..**¥USER, ..¥OEM，..¥ADD_ON** および **MMC2**（この順番で）が検索されます。エントリが見つからない場合は，以下のようになります。

.INI ファイルが前述のディレクトリのいずれかにある（またはその場所にあると思われる，つまり基本パスが同じかディレクトリが指定されていない）場合にはデフォルト値が戻され，

そうでない場合は，標準 C 関数 *GetPrivateProfileString* または *GetprivatePro-fileInt* が呼び出されます。

関数 *ALWritePri-vateProfileString* を使用して，.INI ファイルへデータを書き込む際に，

- .INI ファイルの基本ディレクトリが ..¥USER，..¥OEM，..¥ADD_ON **または** MMC2 の場合，あるいはファイル名にディレクトリの指定が含まれていない場合，宛先としてディレクトリ **..¥USER** が指定されます。.INI ファイルがなければ，作成されます。
- .INI ファイルの基本ディレクトリが基本 MMC ディレクトリと異なる場合，標準 C 関数 WritePrivateProfileString が呼び出されます。

ディレクトリ **..¥USER** がなければ，作成されます。.INI ファイル名にパスの指定が含まれていない場合，標準ディレクトリ**だけ**が検索されるか，または **..¥USER** だけにデータが書き込まれます。

WinApi 関数 ***GetPrivateProfileString，GetprivateProfileInt，***および ***WritePrivateProfileString*** は，Visual Basic 用に「再定義された」ものなので，OEM アプリケーションの場合，それらの 3 つの関数の宣言が存在すれば，専用ファイルから除去しなければなりません。**シーケンス制御が行うのと同じ引き数にキーワード**

***ByVal* が使用されていないかどうか，宣言を調べてください。**

C ライブラリ関数は，関数 **GetProcAddress** を使用して宛先を入手してからでなければ，使用することはできません（まずライブラリ **AB16.DLL** をロードするか，そのハンドルを決定しなければなりません）。. 以下に例を示します。:

- プロトタイプ定義 :

  typedef int WINAPI PFN_GETPRIVATEPROFILESTRING
  (LPCSTR,LPCSTR,LPCSTR,LPSTR,int, LPCSTR);

  typedef UINT WINAPI PFN_GETPRIVATEPROFILEINT
  (LPCSTR, LPCSTR, int, LPCSTR);

  typedef int WINAPI PFN_WRITEPRIVATEPROFILESTRING
  (LPCSTR, LPCSTR, LPCSTR, LPCSTR);.

- 関数ポインタおよび AB16-Dll ハンドルの宣言 :

  HINSTANCE hAB16DLL;
  PFN_GETPRIVATEPROFILESTRING
  *lpfnGetPrivateProfileString;

  PFN_GETPRIVATEPROFILEINT *lpfnGetPrivateProfileInt;

- 関数ポインタの設定で，そのポインタが元の関数を指しているためにエラーになっている場合の，AB16.DLL のロードおよび関数ポインタの設定。

  lpfnGetPrivateProfileString = NULL;

  lpfnGetPrivateProfileInt = NULL;

  hAB16DLL = LoadLibrary ("ab16.dll");
  if (hAB16DLL<0 || hAB16DLL>HINSTANCE_ERROR) {
  lpfnGetPrivateProfileString =
  (PFN_GETPRIVATEPROFILESTRING *)
  GetProcAddress (hAB16DLL, "ALGetPrivateProfileString");
  lpfnGetPrivateProfileInt = (PFN_GETPRIVATEPROFILEINT *)
  GetProcAddress (hAB16DLL, "ALGetPrivateProfileInt");
  }
  if (!lpfnGetPrivateProfileString)
  lpfnGetPrivateProfileString = GetPrivateProfileString;
  if (!lpfnGetPrivateProfileInt)
  lpfnGetPrivateProfileInt = GetPrivateProfileInt;

- AB16.Dll ハンドル（通常は **DLL** 用の関数 **WEP**）の使用可能化

  if (hAB16DLL<0 || hAB16DLL>HINSTANCE_ERROR)FreeLibrary (hAB16DLL);

- 外部としての関数ポインタの宣言（ほかのモジュールで使用するため）

  extern PFN_GETPRIVATEPROFILESTRING
  *lpfnGetPrivateProfileString;
  extern PFN_GETPRIVATEPROFILEINT *lpfnGetPrivateProfileInt;

- プログラムテキスト内での関数ポインタの使用

  (*lpfnGetPrivateProfileString) ("HSoft-
  keyText","Entry","Dflt",szHSoftKey,sizeof(szHSoftKey),szIniFilePath);

シーケンス制御の新しい関数

Function AL_GetSkState(sk As Integer) As Integer

この関数は，sk で指定されているソフトキーがロックされている場合，値 FALSE を戻します。それ以外の場合は TRUE を戻します。

Function ALGetSuccessorBySK(ByVal State As Integer, ByVal SKIn-dexAs Integer) As Integer

この関数は，引き数 State および Softkey-Index で指定されている状態を継承するものを戻します。

LOCK_ETC および *UNLOCK_ETC*

プロシージャ *Lock_ ETC* は，ETC キーをロックします（ETC 画面はグレーになります)。プロシージャ *UnLock_ETC* は，そのロック解除を行います。このプロシージャには引き数はありません。

LOCK_RECALL および *UNLOCK_RECALLT*

プロシージャ *Lock_Recall* は，Recall キーをロックします（Recall 画面はグレーになります)。プロシージャ *UnLock_Recall* は，そのロック解除を行います。このプロシージャには引き数はありません。

Sub Show_A_Hidden_Child(ByVal fname As Form)

このプロシージャは，引き数 *FormName* で指定されている MDI の子を表示します。それまでは，*Hide_A_Child* または *byHide_Childs* で隠されています。

新しい計算式タイプ 5

アプリケーションは，可能であればメモリーに保持する特定の形式を選択できます。これを行うために，シーケンス制御でストラテジが定義されています。シーケンス制御はメモリー内の形式の保持やアンロードを行います。したがってアプリケーションは，その形式をいつでもメモリから除去できるようになっていなければなりません。

注 : この機能については，7 章で説明されています。

1.7.3 リリース 4.3，97 年 12 月

概説

ソフトウェアリリース 4.3 には，リリース 4.2 と比べて追加の機能はありません。

SW 3.6 から SW 4.3 への更新

その意図は，リリース 3.6 で作成された .EXE ファイルとのバイナリ互換性を保持することです。つまり，MMC パッケージ 3.6 用に作成されたアプリケーションが，リリース 4 のインストール後も，制御が正常に実行される必要があるということです。残念ながら，このことはすべてのアプリケーションで保証されているわけではありません。ご使用のアプリケーションで問題が生じた場合，OEM パッケージ 4.3 のシーケンス制御を再生成する必要があります。これを行うには，Visual Basic 4.0 16 ビット版を使用しなければなりません。

アプリケーションに関する注記

OEM パッケージ MMC を使用する際には，以下の注意点が重要になることがあります。

注 : ソフトウェアリリース 4 以降，ファイル "REGIE.INI"，"MMC.INI"，および "RE_XX.INI" は，以下のディレクトリを順序どおりにサーチして評価されるようになっています。
1.MMC2
2.ADD_ON
3.OEM
4.USER
1 章の 22 ページにある，新しいディレクトリ構造も参照してください。

注 : 上記の理由のため，OEM ユーザは "REGIE.INI" と "LANGUAGE¥RE_XX.INI" の違いを OEM パスだけに入力しなければなりません。つまり，OEM パス内のエントリは，標準の MMC2 パス内のエントリを上書きします。

OEMFRAME

現在では，OEMFRAME を使用して 32 ビットのアプリケーションを Regie に組み込めるようになりました。

フォーム形式 5: MAYBE_1OR2

フォーム形式 5 は，現時点ではフォーム形式 2 と同じ働きをします。

以下の特性をこのフォーム形式に割り当てることが計画されています。

それは，リソースが低レベルの場合は，フォーム形式 5 はフォーム形式 1 に変換されてから，メモリから除去されるという特性です。

外部生成されたメッセージテキスト

DOS エディタを使用してメッセージやアラームテキストを作成すると，ä，ö，または ü のような特殊文字に関する問題が生じることがあります。OEM と ANSI では文字セットが違うことが原因です。

Windows では通常，ANSI が使用されます。自動的に認識することはできません。

それでも DOS 生成ファイルをインポートする場合は，MBDDE.INI 内のテキストファイルの名前の後にブランク 1 個とストリング "DOS" を追加してください（大文字も小文字も使用できます）。すると，アラームサーバ MBDDE により，OEM から ANSI への変換が自動的に実行されます。

新規 :

パラメータ DOS の設定後やリセット後には，その影響を受けるテキストファイルの日付を更新しなければなりません（開いて保管するだけでよい）。更新しないと，システムによってパラメータの変更が通知されません（9.4.1 を参照してください）。

開発環境

MS Visual Basic 3/4.0_16 を使用することをお勧めします。

アプリケーション DCTL コントロール

先行するトランザクションが終了しないと，DCTL コントロールによりその次のトラ

ンザクションを実行することはできません。呼び出し側が別の場所で DCTL コントロールを使用して同期トランザクションを処理している際に，その呼び出し側がトランザクション（Read/Write/Execute または Advise）を同じ DCTL コントロールについてパラメータ化しようとすると，エラー : LastError = 07 xx 01 12 が起こります。

シーケンス制御のグローバル変数

他にも，新しくインストールされたディレクトリ ADD_ON，OEM，および USER のパスを備えるグローバル変数が，既存のグローバル変数 g_chMMCPATH（c:¥mmc2 などシステムの MMC2 パスが含まれている）に追加されました。

g_chAdd_OnPath には，ディレクトリ ADD_ON のパスが含まれています。

g_chOemPath には，ディレクトリ OEM のパスが含まれています。

g_chUserPath には，ディレクトリ USER のパスが含まれています。

電源遮断時のデータ管理の問題

問題の説明 :

COPY コマンドか CREATE コマンドを使用する際に，パラメータ -f を指定してデータ管理サーバのソースファイルを指定すると，以下のエラーが起こることがあります。

COPY または CREATE アクションがエラーで打ち切られても，パラメータ -f により宛先ファイルはいかなる場合でも上書きされるので，既存のファイルは削除され，新しいファイルは作成／コピーされません。

処置 :

COPY コマンドを実際の宛先ファイルに直接採用する代わりに，宛先ファイルを MMC から NC 内の一時ファイルにコピーしてください。COPY コマンドが正常に実行されたら（"#100#..."），NC 上の既存の宛先ファイルを削除してから，RENAME コマンドを使用して一時ファイルをリネームしてください。

CREATE コマンドの場合は，まず CREATE コマンドを使用して空ファイルを作成してから，COPY コマンドの場合と同じステップに従ってください。

CREATE に関する問題

問題の説明 :

ソースファイルを指定して CREATE コマンドを使用する場合に，このファイルが非常に大きい（1MB 以上）と，MMC 102 がブロックされます。

処置 :

これが起こる可能性があるのは MMC だけです。ソースファイルを指定しないで CREATE コマンドを使用してください。次に VisualBasic のコマンド “FILECOPY” を使用して，CREATE で作成したファイルを上書きしてください。

1.7.4 リリース 4.4，98 年 8 月

SW 3.6 から SW 4.4 への更新

OEM パッケージのディレクトリ ”SAMPLES.OEM” に組み込まれているファイル "Releas_e.wri" で，"Updating from SW 3.6 to SW 4.4" を参照してください。

構成ツール

ソフトウェアリリース 4.4 以降，構成ツールは OEM パッケージの一部ではなくなりました。

BTSS 変数

以下の BTSS 変数が追加されました。

/Channel/GeometricAxis/DisplayAxis[u<area index>,c<column index>,<row index>]

/Channel/MachineAxis/DisplayAxis[u<area index>,c<column index>,<row index>]

/Channel/ProgramInfo/circleRadiusS[u<area index>,<column index>]

詳しくは，OEM オンラインヘルプ（BTSS_GR.HLP）を使用してください。START-PROGRAM - SINUMERIK MMC V4.4

新しい PI サービス

_N_EXTMOD 外部からのプログラム実行の選択
_N_SETUDT 現行のユーザデータを活動化します。

オンラインヘルプ（PI_GR.HLP）も参照してください。

使用できなくなった PI サービス

_N_ACTDEF Definition 受動ファイルシステムでアクティブな定義 (GUD, Makro)

オンラインヘルプ（PI_GR.HLP）も参照してください。

オンラインヘルプ

BTSS 変数（BTSS_GR.HLP）および PI サービス（PI_GR.HLP）のトピックに関するオンラインヘルプが改作されました。

1.7.5 リリース 5.1 1998 年 12 月

SW 5.1 への更新

ファイル : "Releas_d.wri" の 'Updating to SW 5.1' を参照してください。

新しい OEM の例

新しい OEM の例 :

Oembsp13: コントロール DCTL.OCX と言語切り替えを使用します。
Oembsp14: ソフトキーを使用して C++ アプリケーションを制御します。
Oembsp15: エディタベースのパラメータ化を OEM アプリケーションに組み込みます。

ファイル : "contents.wri" も参照してください。

OEM アプリケーションのインストールガイド

OEM アプリケーションのインストールの手引きが追加されました。ファイル : "Oem-inst.doc" です。

NCDDE サーバ

NCDDE サーバの変数サービスでは，読み取りアクセス用の形式の仕様を，複数の戻り値がある場合に "|" シンボルで区切るように要求できます。旧リリースでは，明示的な形式仕様では区切り文字は生成されませんでした。

05.01 以降では，形式を明示的に指定する場合にも，この区切り文字が生成されます。変数 NCDDE_OPERATION_MODES のビット 0 をセットすると，エラー修正をオフに切り替えることができます。下位互換性のためにビット値 0 になるよう定義されていました。したがって，旧リリースではいずれも，既存でなかった変数には値 0 が使用されます。

BTSS 変数

以下の BTSS 変数が新しく追加されました。

AaEgActive	電子ギア : 指定されている先行軸の結合はアクティブです。つまりオンに切り替えです。
AaEgAx	電子ギア : n 番目の先行軸の軸番号（1 ～ n)。(軸の指標 = 軸番号 - 1)
AaEgBc	電子ギア : ブロック変更基準。EGON,EGONSYN と関係があります。
AaEgDenom	電子ギア : 指定された先行軸の結合係数の分母
AaEgNumera	電子ギア : 指定された先行軸の結合係数の分子
AaEgNumLa	電子ギア : EGDEF により指定された先行軸の数 指定された先行軸の同期座標
AaEgSynFa	電子ギア : 後続軸の同期座標
AaEgType	電子ギア : 指定された先行軸の結合形式
AaEsrEnable	(軸ごとに固有) 関数 "Extended stop and retract" の応答の使用可能化
AaEsrStat	(軸ごとに固有) 関数 "Extended stop and retract" の状態戻りメッセージ
AcAlarmStat	!=0: アラームがアクティブになります。対応するコード化アラーム応答を関数 "Extended stop and retract" のソースとして使用できます。
AcAxCtSwA	ャネルを参照する軸コンテナの状態
AnAxCtAS	コンテナの実アドレス。つまり，軸コンテナ回転時のスロット数。
AnAxCtSwA	軸コンテナの回転がアクティブになります。
AnAxEsrTrigger	(グローバル) 制御フラグ "Stop and retract by drive"

AaEgActive	電子ギア :
APbbIn PLC	入力／出力域 IN のデータバイト（8 ビット）
ApbbOut	PLC 入力／出力域 OUT のデータバイト（8 ビット）
APbdIn	PLC 入力／出力域 IN のデータダブルワード（32 ビット）
ApbdOut	PLC 入力／出力域 OUT のデータダブルワード（32 ビット）
APbrIn	PLC 入力／出力域 IN の実データ（32 ビット）
APbrOut	PLC 入力／出力域 OUT の実データ（32 ビット）
APbwIn	PLC 入力／出力域 IN のデータワード（16 ビット）
ApbwOut	PLC 入力／出力域 OUT のデータワード（16 ビット）
AxisActivInChan	チャネル内の軸がアクティブかどうかを示します。
BadMemFfs	フラッシュファイルシステム（FFS）内の損傷を受けたバイトの数
BasisFrameMask	アクティブなチャネル固有の基本枠を示します。
DiagnoseDataFfs	フラッシュファイルシステム（FFS）の診断データ
FreeMemFfs	フラッシュファイルシステム（FFS）内の空きバイトの数
NcuLinkActive	機械データによって NCU リンクを使用できるかどうかを示します。
NettoMemFfs	フラッシュファイルシステム（FFS）用に使用できる正味のバイト数
NumBasisFrames	基本枠の数
NumUserFrames	チャネルから独立しているユーザ枠の数
ProtAreaCounter	保護域の変更時に加算されるカウンタ（データモジュール PA）
TotalMemFfs	フラッシュファイルシステム（FFS）用の PCMCIA カード上の予約済みバイト数
UsedMemFfs	フラッシュファイルシステム（FFS）内の使用済みバイト数
VaEgSyncDiff	電子ギア : 同期差（実際の値）

オンラインヘルプ（BTSS_GR.HLP）も参照してください。

PI サービス

新規追加 :

_N_SCALE_ 測定単位の設定（メートル <-> インチ）

オンラインヘルプ（PI_GR.HLP）も参照してください。

オンラインヘルプ

オンラインヘルプの BTSS 変数と PI サービスのセクションが更新されました。

1.7.6 バージョン 5.2，1999 年 11 月

SW 5.2 への更新

現行のシーケンス制御を使用して EXE ファイルを新たに作成してください。そうすれば，既存の OEM アプリケーションが安全かつ正しく稼動するよう保証されます。

インストール（2 章）

プログラム Install Shield を使用して MMC-OEM パッケージを CD-ROM からインス

トールする方法が説明されています。

コンポーネント PCU50 の機能（2 章）

新しい章である 2.3 には，コンポーネント PCU50 の機能の一部が詳細に説明されています。

概説

この章には，コンポーネント PCU50 の以下の機能が記載されています。

- Windows 95 の特性
- レジストリ処理の向上
- MMC のシャットダウン
- キーの組み合わせのロック

Windows 95 の特性

Windows 95 の自動ハードウェア識別は使用できなくなり，これを開始するとアイドルループが発生します。
画面の解像度は 256 色に変更されました。
他の設定を使用することもできます。

レジストリ処理の向上

標準的な Windows 環境と MMC Windows 環境の Windows システム環境は，SW 05.01.26 以前のように区別され別々に管理されることはなくなりました。両方のシステム環境が 1 つになります。したがって，ユーザにとってシステム環境が扱いやすくなります（REGISTRY, SYSTEM.INI)。新しい処理方法では，システムのスタートアップ時の面倒なプロンプト（システム設定の変更を保管するかオペレータに尋ねたり，バックアップにも転送するか尋ねたりするもの）は不要になります。システム環境は自動的に管理され，次回のシステムのスタートアップが正常に実行されることが保証されます。システム環境に変更を加えてある場合は，新しい管理方法に自動的に引き継がれます。

以下のようなシステム環境の管理手順が実現しました。

- 作業用コピー，安全コピー，バックアップコピー，および確認用の当社オリジナル環境が（依然として）1 つずつあります。
- 既存の Windows（保守モードまたは MMC モード）では，通常は安全コピーがバックアップコピーになり，作業用コピーが安全コピーになります。もちろん作業用コピーは依然として作業用コピーとしても残ります。このようなしくみにより，システム環境に変更を加えても次回のスタートアップ時に使用できることが保証され，また追加保管されます。す
- スタートアップ時に，作業用コピーを使用してスタートアップを実行できるかどうかチェックされます。実行できる場合，システムは作業用コピーを使用して起動されるので，前回の Windows 終了時の環境を使用できます。実行できない場合，安全コピーを作業用コピーとして使用して，システムが起動されます。最新のシステム設定はシステム保全性に違反するので失われます。安全コピーを使用してスタートアップを実行することもできない場合は，バックアップコピーが作業用コピーになり，システムはこのコピーを使用して起動されま

す。この場合も最新の変更内容は失われます。
- スタートアップを正常に実行するための最後の手段として当社オリジナル環境があります。これには変更を加えることができず，どんな場合でもスタートアップが正常に実行されることが保証されています。スタートアップ後，この環境が作業コピーになります。つまり，配布時の元の状態が再度設定されます。
- 対応するファイルは，C:¥Tools¥Siemens.org，C:¥Tools¥User.sav，C:¥Tools¥User.act，および Widows ディレクトリ（C:¥win.95）にあります。

注：これらのファイルはソフトウェアの更新時に上書きされます。

MMC のシャットダウン

MMC シャットダウン時の機能に変更が加えられました。現リリースでは，［EXIT］ボタンで MMC を抜けると，MMC アプリケーションは閉じ，Windows はシャットダウンされます。"Safe to power off. Press any key to reboot" というメッセージが表示され，システムが停止します。このプロセスでは，自動再始動は停止します。MMC のスタートアップと Windows 保守モードでのスタートアップとではこの点が違います。MMC のスタートアップ時には上記のように Windows がシャットダウンされますが，保守モードのスタートアップ時には MMC だけがシャットダウンされ，Windows はアクティブのままです。

このように機能が改善されたことにより，MMC のシャットダウン時に，デフォルトでは領域メニューの第 2 層の HSK8 に［EXIT］ソフトキーが設けられます。MMC のシャットダウン時に以下の手順が実行されます。

MMC のシャットダウン指示（[EXIT］ボタン）に従って，REGIE と MMC のアプリケーション（OEM-Frame を使用して開始しなかったアプリケーション）により，プロトコル QueryForShutDown が開始されます。次に（REGIE.INI 中のタスク索引に対応するシーケンスに従って），QueryForShutDown メッセージがアプリケーションに送信されます。アプリケーションから最初の否定応答がある（シャットダウンが拒否される）と，直ちに再呼び出しアラーム “...area xxx cannot beshut down” が表示されます。RECALL キーを使用してアラームに応答すると，対応する領域が暗黙的に選択されます。その領域で，必要な操作アクションを実行できます。しかし MMC シャットダウンは中断されるので，後で再始動する必要があります。DOS ボックス内のアプリケーションがアクティブになっている場合も，MMC のシャットダウンは停止し，

対応するメッセージが表示され，最初に DOS ボックスのアプリケーションを終了するように求められます。MMC アプリケーションをすべてシャットダウンできる状態になると，終了プロトコルがアプリケーションごとに 1 つずつ実行されます。OEM 枠から開始されたアプリケーションは WM_CLOSE メッセージを受け取るだけです。アプリケーションがすべてこのように処理されると，Windows は終了しますが MMC はスタートアップします。

注：ソフトキー "Exit" は，ファイル REGIE.INI 内のエントリ "ExitButton=False" によってロックできます。

キーの組み合わせのロック

現リリースでは，Alt+Ctrl+Del，Alt+Tab，Alt+F4 などのキーの組み合わせは隠されています。OEM や保守のためにこれらの組み合わせが必要な場合は，対応するファイ

ル SYSTEM.INI のセクション [MMC103keyb] のエントリ SeqAct に変更を加えることができます。この処理を実行できるのは PCU50 ハードウェア上だけです。当社固有のキーボードドライバが採用されているのはこのハードウェアだけだからです。保守モードではこのエントリは無効です。つまり，すべてのキーの組み合わせを使用できます。
以下のコードが実装されています。
ビット 0: CTRL-ALT-DEL
ビット 1: ALT-F4
ビット 2: ALT-TAB
ビット 3: 左 SHIFT-ALT-TAB
ビット 4: 右 SHIFT-ALT-TAB
ビット 5: CTRL-ESC
ビット 6: ALT-ESC
ビット 7: ALT-SPACE

■ キーボードドライバのカスタマイズ（3 章）

ソフトウェアバージョン 5.1 以降，領域アプリケーションから MMC キーを処理できるようになりました。詳細については，6.3.1.5 と 11.10.4 を参照してください。

■ ファイル REGIE.INI 内の，OEM カスタマ用のエントリ（6 章）

セクション StartupConfiguration の OEM 用のエントリ

OEM アプリケーションを，ファイル 'Re-gie.ini' のセクション [StartupConfiguration] の領域 'Startup12' ～ 'Startup24' に入れる必要があります。他の場所に入れると，PCU50 標準システムと競合する可能性があります。

セクション TaskConfiguration の OEM 用のエントリ

ファイル 'Regie.ini' のセクション [TaskConfiguration] の範囲 0 ～ 23 は，領域メニューによってアクティブにできるタスク用に予約されています。それ以外の番号（24 ～ 63）は，いわゆる子アプリケーション用に予約されています。そのうち 51 ～ 63 は OEM カスタマ用の子アプリケーションに使用できます。

■ 領域アプリケーションからの MMC キーの処理（6 章）

Regie の TaskConfiguration の新しい 2 つの属性を使用すると，領域アプリケーションから絎 MC キー " を処理できます。

属性	意味
GIMMEKEYS	REGIE のキーを使用できるようにするマスクであり，このキーが領域アプリケーションによって処理されるようにします。
ShowAppMenu-Key	領域転換のキーを使用できるようにするマスクであり，このキーが領域アプリケーションによって処理されるようにします。

キーボードフィルタのカスタマイズ

下記の拡張機能を使用して，'MMC キー '（領域転換キーやチャネル切り替えキーなど）をさまざまな領域アプリケーション用にカスタマイズしてください。

カスタマイズを実行すると，たとえばOEMアプリケーションでF10キーを独自の処理に使用し，別のキー（ShowAppMenuKeyによりパラメータ化したもの）を使用して領域転換を開始することができます。

必須の設定と，新しい設定項目 GIMMEKEYS および ShowAppMenuKey は，ファイル REGIE.INI のセクション [TaskConfiguration] に以下のように入力されています。

...
[TaskConfiguration]
TaskX = name := oemframe, ..., GIMMEKEYS := n,
ShowAppMenuKey := m

パラメータ n と m はビットマスクです。これらの意味は属性を使用して記述されます。

属性 GIMMEKEYS:

これは REGIE のキーを使用できるようにするマスクであり，このキーが領域アプリケーションによって処理されるようにします。
GIMMEKEYS:=n
n は 32 ビットのビットマスクで，アプリケーションで独自に処理される REGIE キーを定義します。

ビット 0: 領域転換
1=OEMApp で F10 が処理されます。0=OEMApp で F10 に標準的な処理（領域転換）が実行されます。
ビット 1: チャネル転換
1=OEMApp で F11 が処理されます。0=OEMApp で F11 に標準的な処理が実行されます。
ビット 2: Cancel キー（BigMac）
1=OEMApp で ESC が処理されます。0=OEMApp で ESC に標準的な処理が実行されます。
ビット 3: 機械領域キー
1=OEMApp で SH-F10 が処理されます。0=OEMApp で SH-F10 に標準的な処理が実行されます。
ビット 4: Tab キーの代わりに End キー
End キーを押すと，1=End キーが OEMApp に渡されます。0=Tab が OEMApp に渡されます。

例：

GIMMEKEYS:=15 OEMApp で独自に F10，F11，ESC，SH-F10 が処理されます。

GIMMEKEYS:=1 OEMApp によって独自に F10 が処理されます。

属性 Show-AppMenuKey:

領域転換のキーを使用できるようにするマスクで，このキーがアプリケーションによって処理されるようにします。

ShowAppMenuKey:=m

ここで，m は 32 ビットのビットマスクで，このアプリケーション内で領域転換（F10 置換）をアクティブにするキーを定義します。

ビット 0 ～ 7 は，定義される領域転換キーの仮想キーコードです（seewinuser.h の VK_xxx エントリ）

ビット 16 1=Shift を押さなければなりません。0=Shift を押す必要はありません。

ビット 17 1=Ctrl を押さなければなりません。0=Ctrl を押す必要はありません。

ビット 18 1=Alt を押さなければなりません。0=Alt を押す必要はありません。

例 :

ShowAppMenuKey := 65659

65659 = 0x1007BVK_F12 = 0x7B

→ Shift-F12 を押すと，このアプリケーションに対して領域転換関数を起動できます。

VK_F1 = 0x70，VK_F1 = 0x71，...，VK_F24 = 0x87

注 :ShowAppMenuKey' を使用してアプリケーションに別のキーを領域転換用に割り当てても，キー F10 は依然として作動します。作動しないようにするには，'GimmeKey' を明示的に指定してください。

■ OEMFRAME に関するヒント（6 章）

Windows アプリケーションを領域アプリケーションとして扱うには，以下のステップに従う必要があります。

(1) プログラムファイルをディレクトリ OEM にコピーします。
(2) プレースホルダーアプリケーション OEMFRAME.EXE を使用して，ディレクトリ OEM 内のファイル REGIE.INI のセクション TaskConfiguration にエントリを作成します。
(3) 必要なドライバ（ポインティングデバイス（マウス）用ドライバなど）をインストールします。
(4) ソフトキーテキストをディレクトリ C:¥OEM¥LANGUAGE 内の softkeytext ファイルに入力します。
(5) 特殊値をディレクトリ OEM 内のファイル OEMFRAME.INI のセクション [PROGRAM NAME] に入力します。

注 : ファイル REGIE.INI 内でプログラムを特徴付けるには，Class Name または Window Name を入力するだけで十分です。

注 : パラメータ „CmdLine“ を指定する際には，ディレクトリ名やファイル名にスペース文字を使用することはできません。

ファイル OEMFRAME.INI 内の新しい属性

ファイル OEMFRAME.INI 内で以下の新しい 2 つの属性を使用すると，アプリケーションの統合性が向上します。

nDelayInitComplete=xx

nSecondsToFindWindows=30

属性 nDelayInitComplete:

ファイル 'OEMFrame.ini' 内のエントリにより，WM_INITCOMPLETE の送信を遅延さ

せたり抑止したりできます。この場合，'nDelayInitComplet=xx' を，ファイル 'OEMFrame.ini' の対応するセクションに入力しなければなりません（xx はマイクロ秒単位の時間）。

属性 nSecondsToFindWindows:

OEM アプリケーションによっては，画面表示に長い時間がかかる場合があります（ステップ 7 など）。OEM 枠は，アプリケーションウィンドウのために特定の時間（従来は 20 秒）だけ待機します。その後，アプリケーションは開始されなかったと想定し，終了します。アプリケーションを再度選択すると，もう一度再始動されます。そのため，以下の変更が加えられました。

(1) 最大待ち時間が 20 秒から 40 秒に拡張されました。

(2) 現リリースでは，OEMFrame.INI を使用して待ち時間を構成できます。
[<ApplName>]
nSecondsToFindWindow = ...

注 : この値は，ファイル REGIE.INI 内のタイムアウト値と対応していなければなりません。

例 :
Regie.ini, Section
[TaskConfiguration]TaskX=name:=oemframe, ..., TimeOut:=30000
OEMFrame.ini, Sektion [Application name]
nSecondsToFindWindow = 30

■ 言語の選択とアジアの言語

アジアの言語

アジアの言語には追加のツールが必要です。このツールを使用すると，対応するフォントが自動的にインストールされます。対応するフォントは個々の言語パッケージを使用して自動的にインストールされ，mmc.ini およびレジストリ内の必要なエントリがすべて実行されます。

- DLL のテキストファイル（機械データファイル）と言語依存の INI ファイルは，ディレクトリ "¥mmc2¥language" に格納されます。
- アラームテキストは，ディレクトリ "¥dh¥mb.dir" に格納されます。

言語選択

他の言語のコントロールを構成するには，ソフトキー 'select language' を使用してください。
文献 :
/IAD/ Installation and Start-Up Guide 840D,
13 章「MMC, Configure 'Language selection' softkey」

■ ソフトキーのピクトグラム（7 章）

言語 DLL（先頭が ¥¥）の中でソフトキーのファイル名が指定されている場合は，このファイル名はビットマップの名前として解釈されます（形式は BMP）。末尾は，アプリケーション固有の INI ファイルからの基本パス **SKPICTO** になります。

ディレクトリ **USER，OEM，ADD_ON，**および **MMC2** でこのファイルがサーチさ

れ，検出されると，ファイルに含まれているビットマップが画面に表示されます（SW P4）。

P5 の新規事項 : 絶対パスを使用してビットマップファイルを指定すると（L:¥... など），既存の場合はこのビットマップが表示されます。重要 : DLL の中では '¥' の前にもう 1 つ '¥' を付け加えなければなりません（L:¥¥... など）。

■ シーケンス制御の新しい関数（7 章）

シーケンス制御の 4 つの新しいプロシージャは，ソフトキーテキスト関数を補足し，画面の動的なグラフィック解像度をサポートします。

概説

名前	F/P	意味
ソフトキーテキスト関数		
SK_Highlight	P	ソフトキーを強調表示します。
SK_HighlightUn	P	ソフトキーを即時強調表示します。
動的グラフィック解像度の属性関数		
subSetTFrmAttr	P	フォームの属性を設定します。
subSetTCtrlAttr	P	コントロールの属性を設定します。

■ 追加のソフトキーテキスト関数

■ SK_Highlight

説明

プロシージャ *SK_Highlight* は，パラメータ指標によってアドレッシングされたソフトキーを強調表示します（一方，プロシージャ SK_HighlightUn は対応するソフトキーを即時に強調表示します）。

構文

Sub SK_HighLight(ByVal Index As Integer)

■ SK_HighlightUn

説明

プロシージャ *SK_HighlightUn* は，パラメータ指標でアドレッシングされたソフトキーを，背景色を青くして即時に強調表示します（一方，関数 SK_Highlight はアクションが終了するまでは対応するソフトキーを強調表示しません）。

構文

Sub SK_HighlightUn(ByVal Index As Integer)

■ 動的グラフィック解像度用の関数

概説

PC バージョンの MMC アプリケーションを，設定された画面サイズで表示するために，P5 のシーケンス制御はフォームとフォーム上に表示されるコントロールやテキストのサイズをスケール調整します。しかしこの処理は，シーケンス制御（SC）に認識されサイズ変更されるフォームとコントロールだけに実行できます。

SC でスケール調整できるのはコントロールのサイズだけです。内容に変更を加えることはできません（グリッドやリストボックスなどには SC で認識されない依存関係があります。リストボックスの行のスペーシングなどが該当します)。

この機能をアプリケーションに対して使用できるようにするには，グローバル INI ファイルである MMC.INI の中でこの機能をアクティブにする必要があり，このファイルのプログラムコードに以下のような多少の変更を加える必要があるかもしれません。

グローバル INI ファイル **MMC.INI** のセクション **[CONTROL]** 内のエントリ：

;==Resolution: 0=fixed (640x480), 1=variable, default:0
Resolution=1
;==BaseScreen: 640x480, 800x600, ... default: 640x480
BaseScreen=640x480

- **Resolution=1**（可変）は，PC に実際に設定されている画面解像度が使用されることを示します。
- **BaseScreen=640x480** は，アプリケーションが開発された際の解像度を指定します。当社の場合は，これまで常に（!!!）640x480（デフォルト設定）でした。つまり，普通はこのエントリに変更を加える必要はありません。

Resolution=1 の場合は，INI エントリの **ScreenTwips**（アプリケーション固有の INI ファイルにある）は無視されます。(ScreenTwips=1 は，コントロールで設定されている値に基づく実際の条件とは無関係に，ピクセル当たりの twips の数を 15 に設定します。

シーケンス制御にはグローバルデータ構造が含まれており，アプリケーションに関係のある以下のデータがすべてここに保持されています。

Type **AppRes_Info**
HSize As Integer 水平解像度
VSize As Integer 垂直解像度（タスクバーを除く）
HFact As Single BaseScreen-X を参照する X 係数
VFact As Single (BaseScreen-Y-Taskbar) を参照する Y 係数
End Type
Global **g_tAppRes** As AppRes_Info

コントロールまたはフォームの幅（高さ）に，**g_tAppRes. HFact**（幅の場合），**g_tAppRes. Vfact**（高さの場合）の数量が乗算されます。

プロシージャ **subSetTCtrlAttr** および **subSetTFrmAttr** には，オプションパラメータがあり，このパラメータは対応するコントロールのフォントサイズに係数 **g_tAppRes. Hfactor** を乗算するかどうかを指定します。パラメータを指定しなかったり FALSE にした場合は係数は 1 になり，それ以外は **g_tAppRes. Hfact** になります。

■ SubSetTFrmAttr

説明

プロシージャ **subSetTCtrlAttr** は，パラメータ **Ctrl** で指定された，形式 **wType** のコントロールの属性 FontName，Font-Size，および FontBold を設定します。

アプリケーションにより，FormLoad 中に，すべてのコントロールについてこのプロシージャを呼び出す必要があります。こうすると，アプリケーションのフォームの表示が一様になり，言語を転換しても正しい FontAttributes が得られるようにすることができます。

プロシージャ **subSetTCtrlAttr**(Ctrl As Control, wType As Integer,Optional ByVal size As Variant) には，2 つまたは 3 つのパラメータを指定できます（3 つめはオプション）。3 つめのパラメータを指定した場合，FontSize の設定値が再度計算されないことを示します。これは，高解像度（640x480 ～ 1024x768 など）の場合に特に重要です。高解像度の場合，フォームのロード時に計算が実行されるからです。

■ SubSetTCtrlAttr

説明

プロシージャ subSetTFrmlAttr は，パラメータ frm で指定されたフォームのヘッダ／ダイアログ行（typewType によって定義される）の属性 FontName，Font-Size，および FontBold を設定します。アプリケーションにより，FormLoad 中に，すべてのコントロールについてこのプロシージャを呼び出す必要があります。こうすると，アプリケーションのフォームの表示が一様になり，言語を転換しても正しい FontAttributes が得られるようにすることができます。

プロシージャ subSetTFrmAttr(frm As Form, wType As Integer,Optional ByVal size As Variant) には，2 つまたは 3 つのパラメータを指定できます（3 つめはオプション）。3 つめのパラメータを指定した場合，FontSize の設定値が再度計算されないことを示します。これは，高解像度（640x480 ～ 1024x768 など）の場合に特に重要です。高解像度の場合，フォームのロード時に計算が実行されるからです。

■ NCDDE サーバの構成（8 章）

インストールの設定に応じて，NCDDE サーバの構成には以下の 4 つの方法があります。

- 1 つの NC に対する接続を確立する（デフォルト）
- 1 つまたは複数の NC に対する接続を確立する（M:N - 機能については 8.3.3 を参照）。
- PC 上のローカル操作モード：開発者がこれを使用すると，NC に接続しなくても自分の PC 上で自分のアプリケーションをローカルにテストできます。この場合のために NCDDE サーバには置換値が備えられています。この値はコマンド "NEW"（8.8）を使用して定義し，コマンド "ANIMATE"（8.8）を使用して変更を加えて，アクティブな NC をシミュレートできます。
- PC シミュレータを使用した PC 上のローカル操作モード：開発者がこれを使用すると，NC に接続しなくても自分の PC 上で自分のアプリケーションをローカルにテストできます。NC シミュレータを使用すると，本物の CNC に極めて近い動作を実現できます。

NcddeMachineNamesAdd1

この属性は，ファイル MMC.INI のセクション [GLOBAL] に指定し，インストールされている NC シミュレータを特徴付けます。NC シミュレータがインストールされていない場合は，このエントリには意味がありません。

ネームスペース

LOCAL モードでは，NCDDE サーバでは変数の ' ネームスペース ' は識別されません。' ネームスペース ' は，TOPIC 間の区別に使用されるものです。TOPIC LOCAL 用の変数が作成され，TOPIC Sim0 にも同じ変数が作成された場合，NCDDE では両者を区別しません。このことにより，領域アプリケーション MACHINE で現行のブロック表示による画面が選択されていると，シミュレーションモードでは現行のブロック表示が使用できないなどの影響が生じることがあります。この場合，プログラムにより ' シミュレーション変数 ‘ を上書き定義するローカル変数が作成されます。

■ 新しい DCTL コントロール（32 ビット）（8 章）

このコントロールは，OEMFRAME を介して 32 ビットのアプリケーションを結合し，Siemens コントロールを使用する場合に特に適しています。
このコントロールを使用すると，32 ビットのアプリケーションから PI サービス，Variable サービス，および Domain サービスにアクセスできます。
このコントロールは，VisualBasic®6 だけでテストされています。他の開発ツールを使用すると，正しく機能しない危険性があります。必要なファイルはすべて ¥oem¥dctl にあります。このコントロールを使用するには，対応するディレクトリ（¥oem¥dctl）内のファイル readme.txt に記述されているステップに従ってください。

■ NCDDE サーバの拡張（8 章）

■ 複数変数サービス

概説

複数変数サービスを使用すると，1 つの NCDDE ジョブ内の複数の変数にアクセスできます。そのため，複数ある特定の変数のアクセス速度が上がります。これは読み取りアクセスおよび書き込みアクセス専用です（ホットリンクは不可です）。

項目を指定する際には，対応するアクセス先の特定変数／配列の項目と同様に，'|' で区切ります。読み取りアクセスして得られるデータは，配列にアクセスする際に密に圧縮されます。形式を指定して配列にアクセスする場合や，新しいアクセス変更を指定して（後述）アクセスする場合などは，その前に区切り文字をパラメータ化する必要があります。書き込みアクセスの際に，書き込まれるデータの先頭文字は，さまざまなデータブロックの区切り文字として解釈されます。

制限

- 個々のジョブには，**最大 8 つ**の密に圧縮された PDU が含まれます。これにより，普通は 1 つのジョブ内の 100 を超える変数にアクセスできます（正確な数は試行して判別できます）。
- PDU は宛先アドレスに送信されます。したがって，PLC アクセスと NC アクセスが 1 つのジョブに混在することはありません。さらに，別々のチャネル内のチャネル固有変数へのアクセスも混在**しません**（NC の要件）。ドライブ固有の変数に対するアクセスにも同じことが当てはまります。

- 複数変数サービスでは，**実変数だけ**をアドレッシングできます（BTSS インタフェース／ PLC-BUB）。日付，時刻，システム状態のリスト，ディレクトリ情報などはアドレッシングできません。
- DDE 項目のサイズは **255 文字に限定されていること**に注意してください。項目のストリングがこの限度を超えている場合は，間接的に指定する必要があります（下記参照）。

複数変数サービスを使用した書き込みと読み取りの**例**

項目 :
/channel/parameter/r[1,2](|"!l%ld")|/channel/parameter/r[10](|)
データ例 : |1|2|10.000000

■ 間接的に項目を指定する

間接的に項目を指定すると，255 文字より長い項目を使用できます（最大 4KB）。
NCDDE ローカル変数の内容は，DDE アクセス用の項目として使用されます。この場合，ローカル変数の名前は，'>' 文字を前に付けた項目として指定しなければなりません。
R10 アクセスの場合の例 :
Exec: NEW(x,"/channel/parameter/r[10]")
項目 : >x
データ例 10. 000000

注 : NCDDE サーバでの変数書き込み用やコマンド実行用のデータ長は，4KB に限定されています。この値を超えると，エラー 0X01050414 になります。

■ 新しいアクセス変更

以下の文字を括弧で囲んで項目ストリングに追加できます。

'|' CF_TEXT 読み取りアクセスに関する個々の単一項目の前に '|' シンボルを挿入します。これは書き込みアクセスの場合は評価されません。8.12.1 にある複数変数サービスの例を参照してください。

'^' このタグが付いた変数には，ホットリンクの非活動化は無効です（DEBA/DEBR）。

■ アラームプロトコルの拡張

アラームサーバの初期設定ファイル MBDDE.INI のセクション [PROTOCOL] が拡張され，新しい属性 DiskCare が備えられました。

DiskCare

メッセージモジュール MBDDE の場合，現リリースではアラームプロトコルをハードディスク（ファイル mmc2¥proto.txt）に書き込むかどうか，およびいつ書き込むかを構成できます。旧リリースでは，アラームが生じたり消えたりするたびに，アラームプロトコルがハードディスク上に書き込まれていました。現リリースでは，ファイル MBDDE.INI のセクション [PROTOCOL] 内のエントリ "DiskCare" により，プロトコルファイルがいつ書き込まれるかが制御されます。

以下のパラメータを設定できます。

DiskCare = -1（デフォルト）MBDDE サーバにより，メインメモリでアラームプロトコルが実行されます。診断モードでソフトキーにより命令されている場合に限り，プロトコルはハードディスクに書き込まれます。また，制御がオフになると，ハードディスクに書き込み済みでない限り，アラームプロトコルは使用不可になります。

DiskCare = 0 プロトコルファイルに変更を加えると，即時に格納されます（旧リリースの動作と同じ）。

DiskCare = n アラーム状態の変更内容は，変更が加えられなくなってから n 秒間経過するとプロトコルファイルに書き込まれます。

エントリ DiskCare はスタートアップ時だけ評価されます。

■ 新しい PI サービス（11 章）

PI サービスの関数グループ " 工具関数 " が拡張され, 5 つの新しい関数が追加されました。

関数グループ	意味	PI 名
工具 関数	固有の D 番号のチェックを開始します。	_N_CHEKDM
	姉妹工具グループの工具をアクティブにします。	_N_SETTST
	マガジン内の磨耗グループをアクティブに設定します。	_N_TMAWCO
	定義された数の切削端のある工具を作成します。	_N_TMCRTC
	アクティブ状態にリセットします。	_N_TMRASS

詳細については，オンラインヘルプファイル（PI_UK.HLP）を参照してください。

■ 新しく追加，変更，および除去された変数（11 章）

以下の変数が変更，新規追加，または除去されました。

データ域 A ／データモジュール M（軸固有の機械データ）
MDCA_CTRLOUT_MODULE_NR → _CTRLOUT_MODULE_NR

MDCA_CTRLOUT_TYPE → _CTRLOUT_ TYPE
MDCA_ENC_MODULE_NR → _ENC_MODULE_NR
MDCA_ENC_TYPE → _ENC_TYPE

データ域 A ／データモジュール SE（軸固有の設定データ）
MDB_WORKAREA_MINUS_ENABLE → _WORKAREA_MINUS_ENABLE
MDB_WORKAREA_PLUS_ENABLE → _WORKAREA_PLUS_ENABLE
MDD_SPIND_MAX_VELO_G26 → _ SPIND_MAX_VELO_G26
MDD_SPIND_MAX_VELO_LIMS → _ SPIND_MAX_VELO_LIMS
MDD_SPIND_MIN_VELO_G25 → _ SPIND_MIN_VELO_G25
MDD_WORKAREA_LIMIT_MINUS → _ WORKAREA_LIMIT_MINUS
MDD_WORKAREA_LIMIT_PLUS → _ WORKAREA_LIMIT_PLUS

データ域 C ／データモジュール ETP（イベント形式）
asciiMode（新規追加）

データ域 C ／データモジュール FB（基本枠）
asciiMode（データベースエントリを訂正）

データ域 C ／データモジュール FU（設定可能なゼロオフセット）
asciiMode（データベースエントリを訂正）

データ域 C ／データモジュール S（チャネル固有の状態データ）
aaEgBc（除去）
acwStat（新規追加）
acwTu（新規追加）
acPtpSup（新規追加）
actOnToolLength1（新規追加）
actOnToolLength2（新規追加）
actOnToolLength3（新規追加）
acVactB（新規追加）
axisActivInChan（新規追加）
chanAxisNoGap（新規追加）
pEgBc（新規追加）
pTcAng（新規追加）
pTcDiff（新規追加）

データ域 C ／データモジュール SE（チャネル固有の設定データ）
MDD_DRY_RUN_FEED _DRY_RUN_FEED
MDD_THREAD_START_ANGLE _THREAD_START_ANGLE

データ域 C ／データモジュール SEGA（状態データ : WKS 内のチャネル軸（SGA の拡張機能））
AaVactW（新規追加）
ActProgPosBKS（新規追加）

データ域 C ／データモジュール SEMA（状態データ : MKS 内のチャネル軸（SMA の拡張機能））
aaOffVal（新規追加）
aaVactB（新規追加）
aaVactM（新規追加）
chanAxisNoGap（新規追加）
focStat（新規追加）
saveActPosDiff（新規追加）
saveActVeloDiff（新規追加）
saveMaxVeloDiff（新規追加）

データ域 C ／データモジュール SPARP（パートプログラム情報）
circleTurn（新規追加）
circle TurnS（新規追加）

データ域 C ／データモジュール SSP（状態データ : 主軸)
channelNo（新規追加）

データ域 C ／データモジュール SSP2（状態データ : 主軸）
channelNo（新規追加）

データ域 C ／データモジュール Y（チャネル固有の主軸）
numActAxes（新規追加）
NumMagPlaceParams（除去）
numOriAxes（新規追加）
progProtText（新規追加）

データ域 M ／データモジュール S（内部状態データ MMC）（新規追加）

/Nck/Nck/ActApplication（新規追加）
/Nck/Nck/ActBag（新規追加）
/Nck/Nck/Channel（新規追加）
/Nck/Nck/CoordSystem（新規追加）

データ域 N ／データモジュール DIAGN（グローバル診断データ）（新規追加）
actCycleTimeBrut（新規追加）
actCycleTimeNet（新規追加）
maxCycleTimeBrut（新規追加）
maxCycleTimeNet（新規追加）
minCycleTimeBrut（新規追加）
minCycleTimeNet（新規追加）
pcmciaDataShotAct（新規追加）
pcmciaDataShotSum（新規追加）
pcmciaFfsLength（新規追加）
pcmciaShotStatus（新規追加）
pcmciaStartFfsOffset（新規追加）
pcmciaStartShotOffset（新規追加）

データ域 N ／データモジュール FA（チャネルから独立したアクティブなゼロオフセット）（新規追加）
linShift（新規追加）
mirrorImgActive（新規追加）
rotation（新規追加）
scaleFact（新規追加）

データ域 N ／データモジュール FB（チャネルから独立した基本枠）（新規追加）

linShift（新規追加）
linShiftFine（新規追加）
mirrorImgActive（新規追加）
scaleFact（新規追加）

データ域 N ／データモジュール FB（チャネルから独立したユーザ枠）（新規追加）

linShift（新規追加）
linShiftFine（新規追加）
mirrorImgActive（新規追加）
scaleFact（新規追加）
データ域 N ／データモジュール FU（グローバル機械データ）
MDBA_DRIVE_IS_ACTIVE → _DRIVE_IS_ACTIVE
MDCA_DRIVE_LOGIC_NR → _DRIVE_LOGIC_NR
MDCA_DRIVE_MODULE_TYPE → _DRIVE_MODULE_TYPE
MDCA_DRIVE_TYPE → _DRIVE_TYPE
MDD_INT_INCR_PER_DEG → _INT_INCR_PER_DEG
MDD_INT_INCR_PER_MM → _INT_INCR_PER_MM
MDD_SYSCLOCK_CYCLE_TIME → _ SYSCLOCK_CYCLE_TIME
MDL_POSTCTRL_SYSCLOCK_TIME_RATIO _ POSTCTRL_SYSCLOCK_TIME_RATIO
MDLA_DRIVE_INVERTER_CODE → _ DRIVE_INVERTER_CODE
MDSA_AXCONF_MACHAX_NAME_TAB → _ AXCONF_MACHAX_NAME_TAB

データ域 N ／データモジュール SEMA（状態データ : MKS 内のチャネル軸（SMA の

拡張機能))
aaCoupAct（新規追加）
aaCoupOffs（新規追加）
aaCurr（新規追加）
aaDtbb（新規追加）
aaDteb（新規追加）
aaDtepb（新規追加）
aaLeadP（新規追加）
aaLeadSp（新規追加）
aaLeadSv（新規追加）
aaLeadTyp（新規追加）
aaLeadV（新規追加）
aaLoad（新規追加）
aaMm（新規追加）
aaMm1（新規追加）
aaMm2（新規追加）
aaMm3（新規追加）
aaMm4（新規追加）
aaOff（新規追加）
aaOffLimit（新規追加）
aaOffVal（新規追加）
aaOscillReversePos1（新規追加）
aaOscillReversePos2（新規追加）
aaOvr（新規追加）
aaPower（新規追加）
aaSoftendn（新規追加）
aaSoftendp（新規追加）
aaStat（新規追加）
aaSync（新規追加）
aaTorque（新規追加）
aaTyp（新規追加）
aaVactB（新規追加）
aaVactM（新規追加）
aaVc（新規追加）
ackSafeMeasPos（新規追加）
actCouppPosOffset（新規追加）
actFeedRate（新規追加）
actIndexAxPosNo（新規追加）
actSpeedRel（新規追加）
actValResol（新規追加）
amSetupState（新規追加）
axComp（新規追加）
axisActiveInChan（新規追加）
axisFeedRateUnit（新規追加）
chanAxisNoGap（新規追加）
chanNoAxisIsActive（新規追加）
cmdContrPos（新規追加）
cmdCouppPosOffset（新規追加）

cmdFeedRate（新規追加）
cmdSpeedRel（新規追加）
contrConfirmActive（新規追加）
contrMode（新規追加）
isplayAxis（新規追加）
distPerDriveRevol（新規追加）
drive2ndTorqueLimit（新規追加）
driveActMotorSwitch（新規追加）
driveActParamSet（新規追加）
driveClass1Alarm（新規追加）
driveContrMode（新規追加）
driveCoolerTempWarn（新規追加）
driveDesMotorSwitch（新規追加）
driveDesParamSet（新規追加）
driveFastStop（新規追加）
driveFreqMode（新規追加）
driveImpulseEnabled（新規追加）
driveIndex（新規追加）
driveIntegDisable（新規追加）
driveLinkVoltageOk（新規追加）
driveMotorTempWarn（新規追加）
driveNumCrcErrors（新規追加）
driveParked（新規追加）
drivePowerOn（新規追加）
driveProgMessages（新規追加）
driveReady（新規追加）
driveRunLevel（新規追加）
driveSetupMode（新規追加）
driveSpeedSmoothing（新規追加）
effComp1（新規追加）
effComp2（新規追加）
encChoice（新規追加）
fctGenState（新規追加）
feedRateOvr（新規追加）
focStat（新規追加）
fxsStat（新規追加）
handwheelAss（新規追加）
impulseEnable（新規追加）
index（新規追加）
kVFactor（新規追加）
lag（新規追加）
logDriveNo（新規追加）
measFctState（新規追加）
measPos1（新規追加）
measPos2（新規追加）
measPosDev（新規追加）
measUnit（新規追加）
paramSetNo（新規追加）

preContrFactTorque（新規追加）
preContrFactVel（新規追加）
preContrMode（新規追加）
PRESETActive（新規追加）
PRESETVal（新規追加）
progIndexAxPosNo（新規追加）
qecLrnIsOn（新規追加）
refPtBusy（新規追加）
refPtCamNo（新規追加）
refPtStatus（新規追加）
safeActPosDiff（新規追加）
safeActVeloDiff（新規追加）
safeActVeloLimit（新規追加）
safeDesVeloLimit（新規追加）
safeFctEnable（新規追加）
safeInputSig（新規追加）
safeInputSig2（新規追加）
safeInputSigDrive（新規追加）
safeInputSigDrive2（新規追加）
safeMaxVeloDiffmax（新規追加）
safeMeasPos（新規追加）
safeMeasPosDrive（新規追加）
safeOutputSig（新規追加）
safeOutputSig2（新規追加）
safeOutputSigDrive（新規追加）
safeOutputSigDrive2（新規追加）
spec（新規追加）
subSpec（新規追加）
torqLimit（新規追加）
traceState1（新規追加）
traceState2（新規追加）
traceState3（新規追加）
traceState4（新規追加）
trackErrContr（新規追加）
trackErrDiff（新規追加）
type（新規追加）
vaVactm（新規追加）

データ域 N ／データモジュール SMA（状態データ : MKS 内のチャネル軸)（新規追加）
actIncrVal（新規追加）
actToolBasePos（新規追加）
cmdToolBasePos（新規追加）
extUnit（新規追加）
name（新規追加）
status（新規追加）
toolBaseDistToGo（新規追加）
toolBaseREPOS（新規追加）

varIncrVal（新規追加）

データ域 N ／データモジュール SSP（状態データ : 主軸）
actGearStage（新規追加）
actSpeed（新規追加）
channelNo（新規追加）
cmdAngPos（新規追加）
cmdConstCutSpeed（新規追加）
cmdGearStage（新規追加）
cmdGwps（新規追加）
cmdSpeed（新規追加）
driveLoad（新規追加）
gwpsActive（新規追加）
index（新規追加）
name（新規追加）
opode（新規追加）
speedLimit（新規追加）
speedOvr（新規追加）
spindleType（新規追加）
status（新規追加）
turnStatus（新規追加）

データ域 N ／データモジュール SP2（状態データ : 主軸）
channelNo（新規追加）

データ域 N ／データモジュール Y（グローバルシステムデータ）
maxnumContainerPlaces（除去）
maxnumContainerSlots（新規追加）
numContainerPlaces（除去）
numContainerSlots（新規追加）

データ域 N ／データモジュール YFAFL（NCK の指示によるグループ Fanuc）（新規追加）
Gruppe（新規追加）
gruppe_NUM（新規追加）

データ域 T ／データモジュール TC（工具キャリアのパラメータ）（新規追加）
tcCarr1（新規追加）

tcCarr2（新規追加）
tcCarr3（新規追加）
tcCarr4（新規追加）
tcCarr5（新規追加）
tcCarr6（新規追加）
tcCarr7（新規追加）
tcCarr8（新規追加）
tcCarr9（新規追加）
tcCarr10（新規追加）
tcCarr11（新規追加）
tcCarr12（新規追加）
tcCarr13（新規追加）

tcCarr14（新規追加）
tcCarr15（新規追加）
tcCarr16（新規追加）
tcCarr17（新規追加）
tcCarr18（新規追加）
tcCarr19（新規追加）
tcCarr20（新規追加）
tcCarr21（新規追加）
tcCarr22（新規追加）
tcCarr23（新規追加）

データ域 T ／データモジュール TD（工具データ : 汎用データ）
adaptNo（新規追加）

データ域 T ／データモジュール TUM（工具データ : マガジンユーザデータ）
data _ userData

データ域 T ／データモジュール TUP（工具データ : マガジンの場所のユーザデータ）
data _ userPlaceData

データ域 T ／データモジュール TUS（工具データ : モニタユーザデータ）
data _ userData

詳細については，オンラインヘルプ（BTSS_UK.HLP）を参照してください。

■ キーボードドライバの変更（11 章）

キーボードドライバによる OP 031 の走査コードの変更

一部のキー（VSK0 ～ 7，M キー，ETC キー，'('，')'，単一引用符）については，OP031 により走査コードが生成され，PC 領域に予約されます。これらのコードは標準的なキーコードではなく，今後のリリースでは OP によりこれらのキーコードが生成されなくなるので，これらのキーコードをできるだけ早く除去する必要があります。P5 に含まれているキーボードドライバにはマッピング機能が備えられており，この機能を使用すると，OP031 によって生成された特殊な走査コードを再定義できます。

P5 では，以下の非互換性が生じます。

P5.1 までは，OP031 により以下のコードが作成されました。

表 1.8 SW バージョン 5.1 以前のコード

キー	走査コード	仮想キーコード
VSK0:	0x5E	0xE0
VSK1:	0x5F	0xE1
VSK2:	0x62	0xE2
VSK3:	0x63	0xE3
VSK4:	0x64	0xE4
VSK5:	0x65	0xE5
VSK6:	0x66	0xE6
VSK7:	0x67	0xE7
単一引用符 :	0x68	0xBF（PC など）

キー	走査コード	仮想キーコード
ETC:	0x69	0xE8
MACHINE:	0x6A	0xE9
(	0x6B	0xEA
)	0x6C	0xEB

P5.1UPD 以降は，KeyboardDriver により以下のマッピングが備えられます。

表 1.9 SW バージョン 5.1UPD 以降のコード

キー	走査コード	仮想キーコード
VSK0:	Shift-F1 と同様	Shift-F1 と同様
VSK1:	Shift-F2 と同様	Shift-F2 と同様
VSK2:	Shift-F3 と同様	Shift-F3 と同様
VSK3:	Shift-F4 と同様	Shift-F4 と同様
VSK4:	Shift-F5 と同様	Shift-F5 と同様
VSK5:	Shift-F6 と同様	Shift-F6 と同様
VSK6:	Shift-F7 と同様	Shift-F7 と同様
VSK7:	Shift-F8 と同様	Shift-F8 と同様
単一引用符 :	SingleQuote と同様	SingleQuote と同様
ETC:	Shift-F9 と同様	Shift-F9 と同様
MACHINE:	Shift-F10 と同様	Shift-F10 と同様
(	(と同様	(と同様
)	) と同様	) と同様

マッピングには，ファイル SYSTEM.INI に含まれているテーブルが使用されます。このテーブルで，仮想キーコードが仮想キーコードのシーケンスに置換されます（P5.1UPD 以降）。P5.1 の SYSTEM.INI には，前述のマッピングを実行できるテーブルが備えられます。したがって，P5.1UPD の KeyboardDriver では，PC 互換の走査コードだけが生成されます。' 古い ‘OP031 コードだけを処理する OEM アプリケーションが PCU50 上にインストールされている場合は，SYSTEM.INI にあるマッピング機能が使用できなくなる可能性があります。このことは PCU50 アプリケーションには当てはまりません。

複数の OEM アプリケーションがインストールされており，キーコードに関する要件が異なる場合（つまり，一方の OEM アプリケーションでは ' 古い ' コードだけが処理され，もう一方では ' 新しい ' コードだけが処理される場合など）に問題が生じます。ただし，このような事例は今のところ起こりません。現在までのところ，' 新しい ' キーコードだけに反応する OEM アプリケーションはないからです。このようなアプリケーションは，P5.1UPD より前のリリースの PCU50-SW では実行できません。

すべてのアプリケーションが両方の種類（' 古い ' ものと ' 新しい ' もの）のキーボードコードに反応するのであれば，すべての SW リリース上で実行できるので障害は生じません。OP031 でしか操作できないアプリケーションの場合，以前の走査コードしか処理できないため，問題が生じることがあります。

■ サポートされている言語（11 章）

表 1.10 言語および適用される ANSI テーブル／コードページ

言語	略称	コードページ（DOS）	ANSI テーブル（Windows）
ドイツ語	GR	850	1252
英語	UK	850	1252
スペイン語	SP	850	1252
イタリア語	IT	850	1252
フランス語	FR	850	1252
中国語（簡体字）	CH	936	-
中国語（繁体字）	TW	950	-
韓国語	KO	949	-
日本語	JA	932	-
スウェーデン語	SW	850	1252.
ハンガリー語	HU	852	1250
ポルトガル語	PO	850	1252
チェコ語	CZ	852	1250
トルコ語	TR	857	1254
ロシア語	RU	866	1251
ポーランド語	PL	852	1250
オランダ語	NL	850	1252
フィンランド語	FI	850	1252

■ ANSI のテーブルとフォントの割り当て

表 1.11 ANSI テーブル 1250（中央ヨーロッパ）

フォント	ファイル	文字
Arial	Cearial.ttf	Arial CE（True Type）
Arial	bold Cearіabd.ttf	Arial CE Bold (True Type)
Arial italic	Cearіali.ttf	Arial CE Italic (True Type)
Arial bold italic	Caeriabi.ttf	Arial CE Bold Italic (True Type)

表 1.12 ANSI テーブル 1251（キリル文字）

フォント	ファイル	文字
Arial	Aricyr.ttf	Arial Cyr (True Type)
Arial bold	Aricyb.ttf	Arial Cyr Bold (True Type)
Arial italic	Aricyri.ttf	Arial Cyr Italic (True Type)
Arial bold italic	Aricyrbi.ttf	Arial Cyr Bold Italic (True Type)

表 1.13 ANSI テーブル 1252（西ヨーロッパ）

フォント	ファイル	文字
Arial	Windows 標準	Arial (True Type)
Arial bold	Windows 標準	Arial Bold (True Type)
Arial italic	Windows 標準	Arial Italic (True Type)
Arial bold italic	Windows 標準	Arial Bold Italic (True Type)

上記の言語を使用するためには，Windows で個々の言語をインストールしなければなりません。

■ オンラインヘルプ

オンラインヘルプの BTSS 変数と PI サービスのセクションが更新されました。

■ 今後のソフトウェアリリースの予告

ソフトウェアリリース 6.0（Windows NT 4.0 英語版に基づく）以降，コントロール ddectl，Regie コントロール（rectlp32）および DOS-Box はサポートされなくなります。

2 システムのインストール

この章では，OEM パッケージ MMC のインストールの仕方と，インストールするための条件について説明します。

ソフトウェアライセンスに関する重要事項

OEM パッケージ MMC には，3 つの構成要素があります。

- MMC103 標準ソフトウェア
- OEM ツール，サンプル，ソースファイル
- リモート診断機能

これら 3 つのソフトウェア構成要素の中で，MMC103 標準ソフトウェアだけが，MMC 上にずっと格納されることになります。

MMC 構成要素を開発ツールとして使うだけであれば，これをお客様へ提供する前に，他の 2 つの構成要素をハードディスクから除いてください。これらの構成要素のソフトウェアライセンスは，それぞれ別個のものとなります。

2.1 PCU50 の特長

概説

この章では，構成要素 PCU50 のいくつかの特長がまとめられています。

- レジストリ処理の向上
- MMC のシャットダウン
- キーの組み合わせのロック

レジストリ処理の向上

標準的な Windows 環境と MMC Windows 環境の Windows システム環境は，SW 05.01.26 以前のように区別され別々に管理されることはなくなりました。両方のシステム環境が 1 つになります。したがって，ユーザにとってシステム環境が扱いやすくなります（REGISTRY, SYSTEM.INI)。新しい処理方法では，システムのスタートアップ時の面倒なプロンプト（システム設定の変更を保管するかオペレータに尋ねたり，バックアップにも転送するか尋ねたりするもの）は不要になります。システム環境は自動的に管理され，次回のシステムのスタートアップが正常に実行されることが保証されます。システム環境に変更を加えていても，この新しい管理システムへ自動的に引き継がれます。

以下のようなシステム環境の管理手順が実現しました。

- 作業用コピー，安全コピー，バックアップコピー，および確認用の当社オリジナル環境が（依然として）1 つずつあります。
- 既存の Windows（保守モードまたは MMC モード）では，通常は安全コピーがバックアップコピーになり，作業用コピーが安全コピーになります。もちろん作業用コピーは依然として作業用コピーとしても残ります。このような仕組みになっているので，システム環境に変更を加え，次回の起動時に有効にしたり，さらに保管することができます。
- スタートアップ時に，作業用コピーを使用してスタートアップを実行できるかどうかチェックされます。実行できる場合，システムは作業用コピーを使用して起動されるので，前回の Windows 終了時の環境を使用できます。実行できない場合，安全コピーを作業用コピーとして使用して，システムが起動されます。最新のシステム設定はシステム保全性に違反するので失われます。安全コピーを使用してスタートアップを実行することもできない場合は，バックアップコピーが作業用コピーになり，システムはこのコピーを使用して起動されます。この場合も最新の変更内容は失われます。
- スタートアップを正常に実行するための最後の手段として SIEMENS オリジナル環境があります。これには変更を加えることができず，どんな場合でもスタートアップが正常に実行されることが保証されています。スタートアップ後，この環境が作業コピーになります。つまり，配布時の元の状態が再度設定されます。
- 対応するファイルは，C:¥Tools¥Siemens.org，C:¥Tools¥User.sav，C:¥Tools¥User.act，および Windows ディレクトリ（C:¥win.95）にあります。

注： これらのファイルはソフトウェアの更新時に上書きされます。

MMC のシャットダウン

MMC シャットダウン時の機能に変更が加えられました。現リリースでは，［EXIT］ボタンで MMC を抜けると，MMC アプリケーションは閉じ，Windows はシャットダウンされます。"Safe to power off. Press any key to reboot" というメッセージが表示され，システムが停止します。このプロセスでは，自動再始動は停止します。MMC での起動なのか，Windows サービスモードでの起動なのかが区別されます。MMC での起動時には，Windows は上記のようにシャットダウンされますが，サービスモードでの起動では，MMC だけがシャットダウンされ，Windows は活動状態のままになります。

このように機能が改善されたことにより，MMC のシャットダウン時に，デフォルトでは領域メニューの第 2 層の HSK8 に［EXIT］ソフトキーが設けられます。MMC のシャットダウン時に以下の手順が実行されます。

MMC のシャットダウン指示（［EXIT］ボタン）に従って，REGIE と MMC のアプリケーション（OEM-Frame を使用して開始しなかったアプリケーション）により，プロトコル QueryForShutDown が実行されます。次に（REGIE.INI 内のタスク索引に対応するシーケンスに従って），QueryForShutDown メッセージがアプリケーションに送信されます。アプリケーションから最初の否定応答がある（シャットダウンが拒否される）と，直ちに再呼び出しアラーム “...area xxx cannot beshut down” が表示されます。RECALL キーを使用してアラームに応答すると，対応する領域が暗黙的に選択されます。その領域で，必要な操作アクションを実行できます。しかし MMC シャットダウンは中断されるので，後で再始動する必要があります。DOS ボックスでアプリケー

ションが活動状態である場合も，MMC シャットダウンは停止し，対応するメッセージが出され，まず DOS ボックスのアプリケーションを終了するよう求めてきます。

すべての MMC アプリケーションをシャットダウンする準備ができたら，終了プロトコルがアプリケーションごとに 1 つずつ実行されます。OEM 枠から開始されたアプリケーションは WM_CLOSE メッセージを受け取るだけです。アプリケーションがすべてこのように処理されると，Windows は終了しますが MMC はスタートアップします。

注：ソフトキー "Exit" は，ファイル REGIE.INI 内のエントリ "ExitButton=False" によってロックできます。

キーの組み合わせのロック

Alt+Ctrl+Del，Alt+Tab，Alt+F4 のようなキーの組み合わせは，現在のところ隠されています。OEM や保守のためにこれらの組み合わせが必要な場合は，ファイル SYSTEM.INI のセクション [MMC103keyb] のエントリ SeqAct に適切な変更を加えることができます。この処理を実行できるのは MMC103 ハードウェア上だけです。Siemens 固有のキーボードドライバが採用されているのはこのハードウェアだけだからです。保守モードではこのエントリは無効です。つまり，すべてのキーの組み合わせを使用できます。以下のコードが実装されています。

ビット 0: CTRL-ALT-DEL

ビット 1: ALT-F4

ビット 2: ALT-TAB

ビット 3: 左 SHIFT-ALT-TAB

ビット 4: 右 SHIFT-ALT-TAB

ビット 5: CTRL-ESC

ビット 6: ALT-ESC

ビット 7: ALT-SPACE

2.2 開発環境の最適化

概説

この節では，作業をやりやすくするための秘訣とヒントを紹介します。

リソースの節約

WINDOWS プログラムを開発し実行するときには，システムリソースが決定的なボトルネックになります。開発が終了してしまってから，リソースを最適化することは困難です。プログラミングのリソース節約の方法に熟達し，コードの設計および実装の段階で制限事項を考慮に入れるようにすると良いでしょう。

Visual Basic 3 プログラム言語の資料と，次に示す記事がこの作業に役立ちます。

- Visual Basic Version 3.0 for Windows"Programmer's Guide" の 641 ～ 646 ページの

付録 D「Specifications and Limitations」。

- Visual Basic Version 3.0 for Windows"Programmer's Guide" の 255 ～ 263 ページの 11 章「Optimizing your Application for Size and Speed」。
- Microsoft TechNet CD, PSS ID Number: Q112860, "How to Opti-mizeMemory Management in VB3.0 for Windows"

Visual Basic ファンクションの使用制限

ここでは，MMC 103 環境で使用を避けるべき Visual Basic のファンクションをいくつか紹介します。

■ MessageBox および InputBox（ファンクション）:

Regie 自体が，どのウィンドウを前面に表示するかを決定します。したがって，ダイアログボックスへの入力を待っている状態でも，そのダイアログボックスが隠れてしまう可能性があり，MMC 操作が妨げられます。代わりのファンクションとして，ModalDialog が備えられています。

■ 1 つのフォームの複数のインスタンス:

コントロールを実行するときに，Regie は UNLOAD を呼び出します。暗黙的なフォーム変数（フォーム形式と同じ名前）が使われる場合に限り，Visual Basic 側で（必要ならば）新しいインスタンスを作ります。

■ DoEvents:

再帰呼び出しを迂回することは難しいため，これは使わないでください。

2.3 運用時の問題と解決方法

概説

続くセクションでは，運用時に生じる可能性のある問題と，ホットラインで連絡せずにその問題を解決する方法を紹介します。

LoadLibrary の初めての呼び出し

LoadLibrary によって初めて DLL（ここでは REGIE.DLL）が呼び出される場合，REGIE.DLL ファイルは，実際のディレクトリか WINDOWS 検索パスに存在していなければなりません。開発の状況によっては（たとえば，デバッガでテスト中である場合），LoadLibrary が REGIE.DLL ファイルを見つけられない場合もあります。

対策 **:**
WPS（‘Windows Process Status’。Visual Basic の開発環境に存在する）を使い，DLL をいったん始動します。これ以降に LoadLibrary を呼び出すと，DLL の参照カウンタが増加するだけ（ロード処理は行われない）なので，正常に動作します。

Child 名のつづり

Child 名は必ず正確につづり，ASCII ファイルで使われているときには，" で囲む必要

があります。Child 名では，小文字と大文字の区別があります。Visual Basic リリース 3.0 でこのような指示を無視すると，システムが破損してしまいます。
Child 名は 8 文字を超過しないようにします。

MDIchild のコピー

次のような手順を進めてゆき，すでに存在している同じような MDIchild から，新しい MDI の子を生成します。

- プロジェクトを格納します。
- 既存の MDIchild の特性名とタグを変更します。
- [名前を付けて保存 ...］を使い，変更した child を新しい名前で格納します。これで，元の child はプロジェクトの一部ではなくなります。
- [Add File］を使い，元の MDIchild をプロジェクトへ追加します。

START-MDIFORM での ‘Lock’ ファンクション

START-MDIFORM の FormLoad で Set_State が呼び出されている場合に，PRIVATE.BAS ファイルの private_ini() の下で Set_State（START-MDIFORM）ファンクションを使うと，‘lock’ ファンクションが誤動作します。

SetState() がシーケンス制御によって 2 度呼び出されるので，‘lock’ ファンクションは取り消されます。

対策 **:**
START-MDIFORM の ‘lock’ ファンクションを，State_Reached へ移動します。

シーケンス制御からのエラーメッセージ

エラーメッセージとして

"File AL_UTIL.DLL (or similar) not found"

が表示される場合，VB において，オプション／プロジェクト／コマンド行引数の下のパスが正確に指定されていない可能性があります。これは，次のように指定する必要があります。

L:¥mmc2.

Alcommon.bas:State_init

.

.

->Check FormNames

FormNames が指定ファイルと一致したら，altmp ディレクトリからご使用のアプリケーションの一時ファイルを除去するようにします。

ソフトキーの連続クリック

ソフトキーを複数回クリックすると，システムが破損することがあります。

説明 :
ソフトキーを複数回クリックすると，エラーメッセージとして

“ALCOMMON.BAS Set_Action, Error: Foreign application won’t per-formDDE-method or

operation” が表示されます。続いて，"Error: The opera-torsystem has reached a critical point ---> reboot the system" が表示されます。

理由 :
ソフトキーアクションによって確立された DDE 接続がまだ終了しておらず，この接続が存在するのに同じソフトキーアクションで DDE 接続を再び確立しようとしています。

対策 :
DDE 接続を確立する前に AL_STOPSKS（ソフトキーのロック）を使い，DDE 接続の確立後に AL_RESUMESKS（ソフトキーの解放）を使います。

テスト PC での表示サイズ

説明 :
テスト環境で，VGA 解像度の OP031 で表示される枠に，すべてのフォームが適合しているわけではありません。ソフトキーが 2 行（外側の行と内側の行）で表示されてしまいます。

対策 :
ご使用のアプリケーションの .INI ファイルの［CONTROL］セクションに，ScreenTwips=1 という行を追加します。

モノクロディスプレイ（白黒）でのコントラストが足りない

対策 :
DOS では，CTRL-ALT-U キーの組み合わせでコントラストを上げたり，CTRL-ALT-D キーの組み合わせでコントラストを下げたりすることができます。

プログラムを使って同じ効果を得るには，REGIE.DLL ファイルの ContrastUp または ContrastDown ファンクションを使います。

電源遮断時のデータ管理の問題

問題の説明 :
COPY コマンドか CREATE コマンドを使用する際に，パラメータ -f を指定してデータ管理サーバのソースファイルを指定すると，以下のエラーが起こることがあります。

COPY または CREATE アクションがエラーで打ち切られても，パラメータ -f により宛先ファイルはいかなる場合でも上書きされるので，既存のファイルは削除され，新しいファイルは作成／コピーされません。

対策 :
COPY コマンドを実際の宛先ファイルに直接採用する代わりに，宛先ファイルを MMC から NC 内の一時ファイルにコピーしてください。COPY コマンドが正常に実行されたら（"#100#..."），NC 上にすでに存在している宛先ファイルを削除し，RENAME コマンドで一時ファイルをリネームします。

CREATE コマンドの場合は，まず CREATE コマンドを使用して空ファイルを作成してから，COPY コマンドの場合と同じステップに従ってください。

CREATE に関する問題

問題の説明 :
ソースファイルを指定して CREATE コマンドを使用する場合に，このファイルが非常に大きい（1MB 以上）と，MMC 103 がブロックされます。

対策 **:**
これが起こる可能性があるのは MMC だけです。ソースファイルを指定しないで CREATE コマンドを使用してください。次に VisualBasic のコマンド “FILECOPY” を使用して，CREATE で作成したファイルを上書きしてください。

2.4 OEM ユーザーへの注意点 - " ハードディスクのバックアップ／復元での Ghost の使用について "

ハードディスクの内容を GHOST を使って保存することについては，HMI Advanced Startup Manual で詳しく説明されています。

そこでは，イメージのインポートについても説明されています。

インポート中には次のダイアログが表示されます :

What kind of partitioning is the selected image of?

1 Standard partitioning (default)
2 User-defined partitioning

ここでは，イメージの基になるパーティション設定を（オペレーティングシステム固有の）標準的なパーティション設定にするか，それともパーティション設定を自由に決定するかを選択できます。

標準のパーティション設定を選択すると GHOST は自動的にイメージをインポートしますが，パーティション設定の自由決定を選択すると，インポートは対話式に実行されます。ただし，オペレータがパーティションサイズを入力する必要があるわけではありません。

標準のパーティション設定を選択した場合，インポートされるイメージのオペレーティングシステムとハードディスクの実際の容量に応じて，パーティションサイズが自動的に決まります。パーティションサイズは（テキストエディタを使って）Ghost.ini ファイルで確認することができます。パーティションサイズは，パーティション設定の可能な範囲（Range）の中で設定します。つまり，

[DOS]
Ranges=....
....
....
[Win95]
Ranges=....
....

```
....
[WinNT]
Ranges=Min_450MB UpTo_2100MB UpTo_4800MB
UpTo_2100MB=19P 19P 28P V
UpTo_4800MB=10P 20P 25P V
Default=2048M 2048M 25P V
```

可能な入力値 :

Min _yMB: ハードディスクには少なくとも y MB の容量が必要です。これに満たない場合はイメージがインポートされません。

UpTo_yMB: ハードディスク容量が x MB と ≤ y MB の間（前の UpTo_xMB または Min_xMB の次の範囲に入る値）であれば，UpTo_yMB の指定内容でパーティションが決定されます。

Max_yMB: ハードディスク容量は y MB 以下でなければなりません。これを超える場合はイメージがインポートされません。

ハードディスク容量が x MB と ≤ y MB の間（前の UpTo_yMB または Min_xMB の次の範囲に入る値）であれば，パーティションは Max_yMB の指定内容で決定されます。

サイズ範囲に上限がない場合（サイズ範囲の昇順の最後の項目は，Min_yMB または UpTo_yMB です。そうでない場合，'Ranges（範囲）' にはサイズ範囲が指定されません），パーティションは 'Default（デフォルト）' 設定に従って決定されます。

実際のオペレーティングシステムで有効なパーティション設定は，サービスメニューの開始時に決定されます。

詳細なステップは，HMI Advanced Startup Manual を参照してください。

3 MMC コンポーネントの基本

840DI で OEM アプリケーションを開発するには，システム全体に通じていなければなりません。つまり，標準操作構成要素がユーザの観点からどのように機能するか，また構成要素がどのように協働しなければならないかを知っておく必要があります。
この章では関係する構成要素の概要を説明します。この章に加えて 5 章から 10 章（構成要素の説明）を読めば，問題をすばやく解決できるようになります。

MMC 構成要素の機能を詳しく知るためには，以下の資料をお読みになるようお勧めします。

- /BA/ Operator’s Manual
- /BH/ Operator Components Manual
- /FB1/, /FB2/ Description of Functions
- /IAD/ Installation and Start-up Guide

その他に MMC アプリケーションのプログラマは，次のことに精通している必要があります。

- MS-Windows 95 オペレーティングシステム。
- Windows リソースの作成（Visual C++ など）。
- テクノロジ固有の Windows オペレータインタフェースの設計。

3.1 システムの概説

機能構成要素

CNC コントロール 840DI の機能は，次の 5 つの構成要素に分けられます。

- ヒューマンマシンインタフェース（MMC）
- プログラマブルロジックコントローラ（PLC）
- 数値制御カーネル（NCK）
- ドライブ制御

その他に，領域間の通信，外部構成要素との通信が可能になります。

3.1.1 オペレータ構成要素

操作ユニット

操作ユニットは次のような幾つかのハードウェア構成要素から成ります。

- NC キーパッドを含むディスプレイ装置
- マシン操作パネル
- MF2 標準キーボード
- MMC モジュール
- オプション

ディスプレイ装置

CRT モニタまたはフラットスクリーンモニタ（組み立てフレーム，電源ユニット，ソフトキー，内蔵 NC キーパッド付き）。

- GUI の設計についてのヒントは 4 章にあります。
- ソフトキーの割り当てについては 7 章で説明されています。
- NC キーパッドでの Windows アプリケーションの操作については 7 章で扱われています。
- データの表示については 8 章で説明されています。

マシン操作パネル

- マシン操作パネルには，EMERGENCY STOP（非常停止）ボタンと操作モード用のキーがあります。さらに，方向，主軸，給紙制御，キーロックスイッチ，カスタマーキーについてプログラム制御が可能です。分岐接続インタフェース（MPI）経由で MMC 構成要素に接続します。
- 8 章には，マシン操作パネルから受信した信号を評価するためのヒントが記されています。
- キーロックスイッチ経由でのアクセス許可の処理については，10 章で説明されています。
- MPI に関する情報は 3 章に記されています。

標準キーボード

MF2 規格に準拠した US 標準レイアウトの QUERTY キーボード(ファンクションキー，カーソル付き)。

- OEM ユーザが特に留意すべきことはありません。

MMC モジュール

OEM ユーザはいくつかのプロセッサタイプとメモリ拡張から選択することにより，目的に応じてモジュール 840DI のパフォーマンスをハードディスクに適合させることができます。

これは通常のインタフェースを持つ業界標準 PC です。オペレーティングシステムとして MS-Windows 95 がインストールされています。

- OEM パッケージ MMC を完全にインストールするためには，2 章を参照してください。
- ハードディスクの残りの内容についても，2 章でリストされています。
- OEM アプリケーションをユーザのハードディスクにコピーする方法を知りたい場合は，引き続きこの章をお読みください。

3.1.2 オプション

基本システムの機能を拡張するために，いくつかのインタフェースや追加構成要素が備えられています。

- PC 拡張ボード用の PCI/ISA ボックス
- フロッピーディスクドライブ用のインタフェース

PCI/ISA ボックス

PCI/ISA ボックスはハードウェア拡張ボックスで，ISA または PCI 規格の 2 つの標準

PC カードまで設置することができます。

これらの構成要素を使用するときに考慮すべきシステムデータについては，11 章で説明されています。

インストール時に一般的に発生する問題の解決方法については，2 章で扱われています。

適切に使用するための機械的な要件と電力上の要件は，Operating Components /HB/ マニュアルで説明されています。

3.2 MMC のソフトウェアアーキテクチャ

3.2.1 MMC ソフトウェアの層

概観

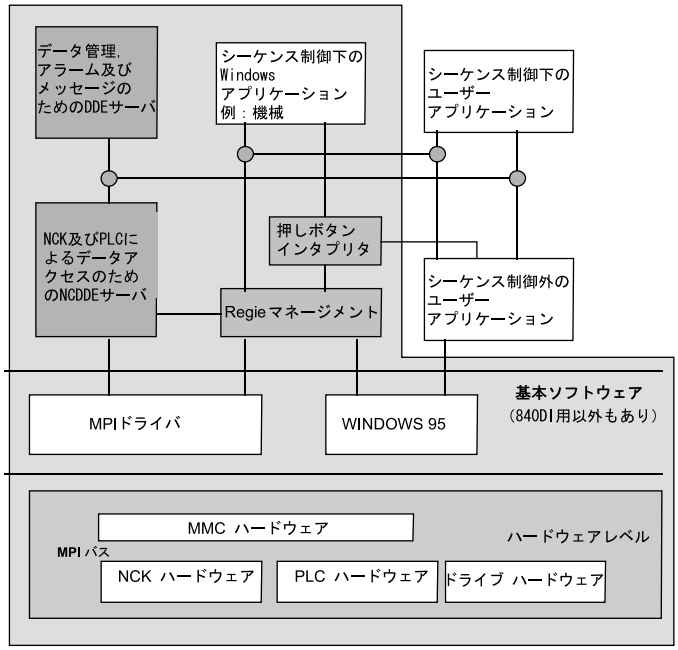


図 3.1 MMC ソフトウェアの編成

解釈の助け

図 3.1 は，アプリケーションの視点から見た MMC のソフトウェアの各層の図式を示しています。

グレーで強調表示されている領域は，標準デリバリボリュームを表しています。ここには，3 つのレベルがあります。

- ハードウェア
- オペレーティングシステムとドライバ
- アプリケーション

標準アプリケーションの他に，コントロール上でカスタマー固有のアプリケーションを実行することができます。

既製のプログラム（例えば CAD システム）に加えて，840DI シーケンス制御を含む独自のアプリケーションが使用可能です。Visual Basic のインタフェースが提供されています。これによってプログラムは，840DI グローバル変数にアクセスすることができ，また Visual Basic がなければ応答できなかったいくつかのイベントについて通知されるという利点があります。

ドライバとオペレーティングシステムを使用することによって，アプリケーション開発者として，ハードウェア層から独立して作業できます。

通常，シーケンス制御のないアプリケーションは，840DI での使用を目的として開発

されたのではないプログラム（CAD プログラムなど）です。ユーザインタフェースをそれに適合させた場合，それらのアプリケーションを 840DI コントロール上で実行することができます。起きる可能性のある問題については，4 章で説明されています。

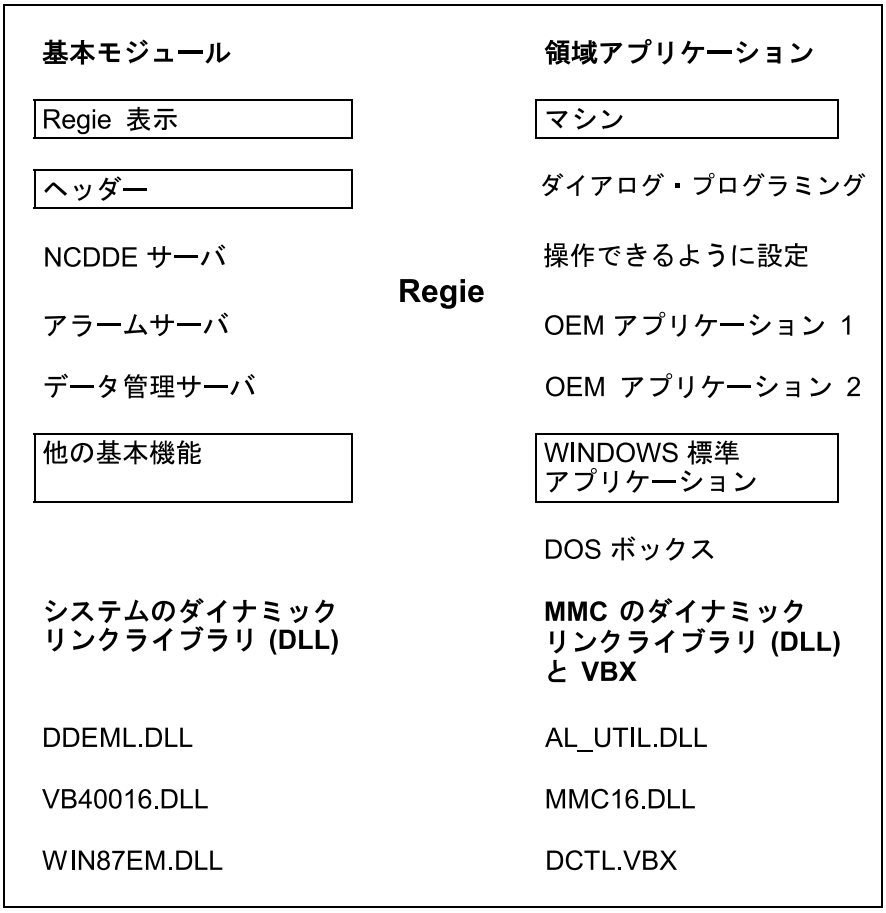
3.2.2 Regie

特長

Regie とは，次のものを柔軟に管理するための調整プログラムです。

- 補助プログラム
- 領域アプリケーション
- ダイナミックリンクライブラリ
- VBX ファイル

以下の図にリストされている通りです。このように Regie は Windows のプログラムマネージャに相当します。



特に，‘Regie’ アプリケーションは以下の処理を行います。

- システムの初期化
- システムのスタートアップ
- システムのダイナミックリンクライブラリと MMC のダイナミックリンクライブラリのロード
- アプリケーションの正しい順序での開始

- システムの構成
- 領域の変更

シーケンスの開始

Regie は，まず補助プログラムを開始し，続いて領域アプリケーションを開始します。Regie は初期化が正常に行われたことを示すそれぞれのプログラムからの戻りメッセージを待ちます。

この処置によって，要求され次第，できるだけ早く必要な補助機能が提供されます。

Regie により開始されたアプリケーションを閉じるときにも，同様の手順がとられます。

OEM 用の Regie

Regie は OEM アプリケーションの管理も行います。OEM ユーザはアプリケーションをシステムに知らせるために，Regie の特定のファイルを編集することができます。

Regie が管理可能な限度は次のとおりです。

- 32 個の補助アプリケーション
- 32 個の領域アプリケーション
- 64 個のシステムのダイナミックリンクライブラリ（DLL）
- 64 個の MMC のダイナミックリンクライブラリ

キー割り当て

OEM ユーザに渡すことのできる NC 操作パネルと PC 標準キーボード両方の可能なキー割り当てが，次の表にリストされています。

表 3.1 NC キーボードと PC（MF2 キーボードのキーの割り当て）

NC キーボード	MF2 キーボード
水平ソフトキー 1 ～ 8	F1 ～ F8
再呼び出しキー	F9
領域変更キー	F10
チャネル変更キー	F11
通知キー	F12
アラーム確認	Escape
ウィンドウ変更キー	Home
終了	End

タビュレーター

Windows アプリケーションの場合，END キーを TAB キーに変換する必要があります。なぜなら NC 操作パネルには，使用可能な TAB キーがないからです。

キー割り当て

Regie はすべてのキーストロークを受け取り，それをアクティブなアプリケーションに応じて渡します。

[F10]（領域変更）は常に Regie で評価されます。

アクティブアプリケーションが DOS ボックスで実行されている場合，Regie はその他のすべてのキーストロークを受け取ります。

アクティブアプリケーションが OEM フレームの場合，領域変更キー，チャネル変更キー，マシンキーの［cancel］，［end］は Regie で評価され，その他すべてはアプリケーションで評価されます。

シーケンス制御のあるアプリケーションがアクティブの場合，Regie は［END］，［HOME］を評価し，それに加えてすべてのファンクションキーを評価します。シーケンス制御はソフトキー（F1 ～ F8 と SHIFT F1 ～ SHIFT F8）をアプリケーションに渡します。

OEM フレーム

特別な領域アプリケーション OEM フレーム（OEMFRAME.EXE）は，特殊コードからソフトキーコードをフィルタに掛け，それらのキーコードを Windows アプリケーション用の標準キーコード（F1 ～ F8 と Shift F1 ～ Shift F8）に変換します。

キーボードドライバのカスタマイズ

ソフトウェアバージョン 5.1 以降，領域アプリケーションからの MMC キーの処理が可能です。詳細については，本書の 6.3.1.5，および 11.10.4 を参照してください。

3.2.3 シーケンス制御

特長

シーケンス構造は，SIEMENS 標準アプリケーションおよび互換 OEM アプリケーションのためのフレームを備えています。これには次のような特長があります。

- シーケンス制御（状態からなるメニューツリー）の管理
- ソフトキーの照会（縦方向と横方向）
- NC 特殊キーの照会
- ソフトキーテキストの表示
- 対話式プロンプト行の管理

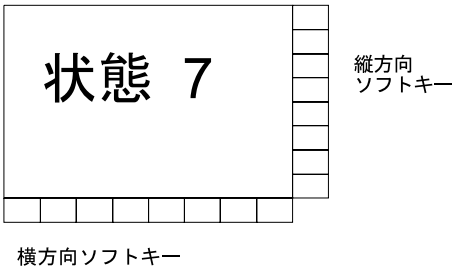
状態

シーケンス制御を記述するための中心的な要素は状態です。状態には次のような特長があります。

- 固有の状態番号
- その状態で表示される MDIchild のリスト
- ソフトキー設定の定義 : テキスト索引と割り当てられた機能
- z フラグ : 保管または終了の設定

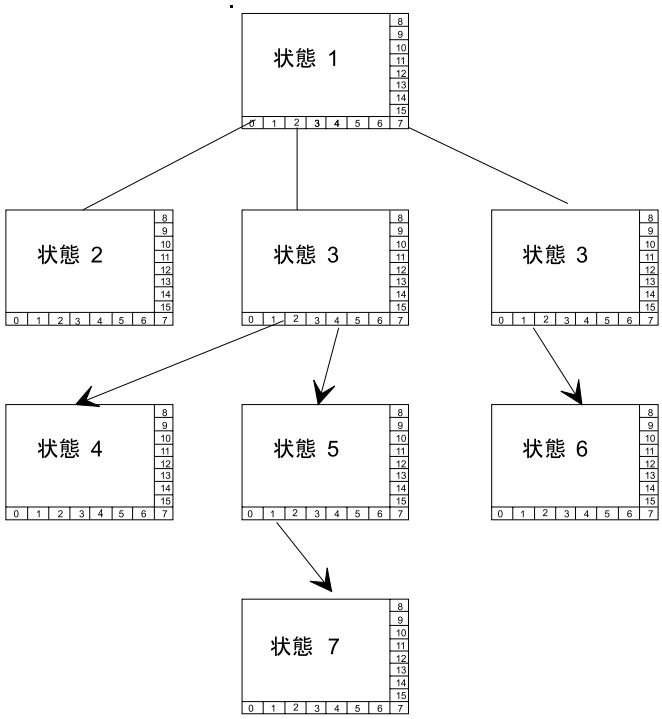
アプリケーションの .INI ファイルに開始状態の番号を保管することができます。

構造図では，状態は次の図のようにシンボルによって表されます。



メニューツリー

これらのシンボルをメニューツリーに結合させることができます。次の図はその例を示しています。



状態変換

状態変換は以下の事象により発生します。

- ソフトキーアクション（縦方向，横方向）または再呼び出し。これらの状態変換は状態マトリックス（.ZUS という拡張子のファイル）に記述されます。
- 外部からの影響に対するプログラムされた反応。たとえば，アプリケーション Machine は操作モードの変更に対して反応します。

以下は状態変換の例です。

- 別のウィンドウをオープンすることによりウィンドウを隠す。
- ソフトキー機能を変更する。

キーボード入力の処理

キーボードパネル，つまりショート NC パネルのキーは，標準 PC キーボードのキーとは異なります。これらのキーには，たとえばアラーム確認，領域変更のように，NC 固有の特別の意味を持つものがあります。

これらのキーは，通常はエントリフォーカスのない（すなわち，バックグラウンドで実行されている）タスクによって基本システムで処理されます。

3.2.4 NCDDE サーバ

概要

NC-DDE サーバはデータ転送に関する次の 3 つのジョブを実行します。

- 変数サービス：NC データ，PLC データ，ドライブデータにアクセスします。
- ドメインサービス：ファイルを MMC から NCK，また NCK から MMC へコピーします。
- PI サービス : NC のプログラム呼び出しサービスを開始します。

変数サービス

DDE サーバは，一例として，表 3.2 に従って NC データへのアクセスを可能にします。DDE（動的データ交換）は，Windows アプリケーション間の動的データ転送に特有の機能です。

表 3.2 NC-DDE サーバで提供されるデータの概要

データグループ	データタイプ
マシンデータ	グローバルマシンデータ チャネル固有のマシンデータ 軸固有のマシンデータ
設定データ	グローバル設定データ チャネル固有の設定データ 軸固有の設定データ
プログラムパラメータ	フレーム ツール修正 ユーザパラメータ
実値	軸座標の実値 送り速度の実値 主軸値 オーバライド設定値
PLC データ	入力 データブロック 出力 データブロックリスト フラグ クロック タイマー システム状態リスト カウンター メッセージ／アラーム

データグループ	データタイプ
ファイル	パートプログラム サブプログラム ツールデータ

データ記述

変数はデータ記述ファイルにまとめられています。すべての変数についての概要を，リストブック /LIS/，またはオンラインヘルプ機能で参照することができます。

テスト用のデータ表示

開発時やテスト時に DDE 接続を使用可能にするために，特殊変数を定義することができます。

- NEW：データの定義
- ANIMATE：スクリーンデモンストレーション値の変更

画面上のデータ表示

Visual Basic 標準コントロールを使用して，画面にデータを表示します。速さを求める特別な要求に応じて，新しく開発された Vusual Basic の補足（VBX. ファイル）が OEM パッケージに備えられています。

テスト用の DDE ツール

DDE 接続のテスト用に, OEM パッケージ MMC のデリバリーボリュームに DDE テストツールが含まれています。

■ *ドメインサービス*

NC-DDE サーバのドメインサービスを使用して，MMC と NCK の別々のドメインに含まれるデータを，両方向に転送することができます。

ドメインサービスのコマンド

オンラインヘルプで有効なコマンドを調べることができます。コマンドは次のようなパラメータと共に表示されます。

- MMC Windows 環境のファイル名
- NC 環境のファイル名
- 状態を返す変数の名前

転送の状態

状態はモニタされ，次の 5 つの状態で表現されます。

- 転送を開始した
- 転送を実行中
- 転送を終了した
- 転送は正常に行われた
- 転送はエラーコードをもって停止した

表 3.3 ドメインサービスのコマンド

コマンド	説明
COPY_TO_NC	追加情報付きの MMC から NCK へのコピー
COPY_TO_NC_BINARY	追加情報なしの MMC から NCK へのコピー
COPY_FROM_NC	追加情報付きの NCK から MMC へのコピー
COPY_FROM_NC_BINARY	追加情報なしの NCK から MMC へのコピー

■ *PI サービス*

PI サービスを使用して，コマンドを NC と PLC に転送することができます。

備えられている実行コマンドを使用します。

PI_START（引数）PI サービスを開始する

PI_STOP（引数）PI サービスを停止する

PI_RESUME（引数）停止していた PI サービスを再開する

有効なパラメータは次にリストしたタイプの関数です。詳細は 8 章で説明されています。オンラインヘルプにも，使用する可能性のあるパラメータについて説明があります。

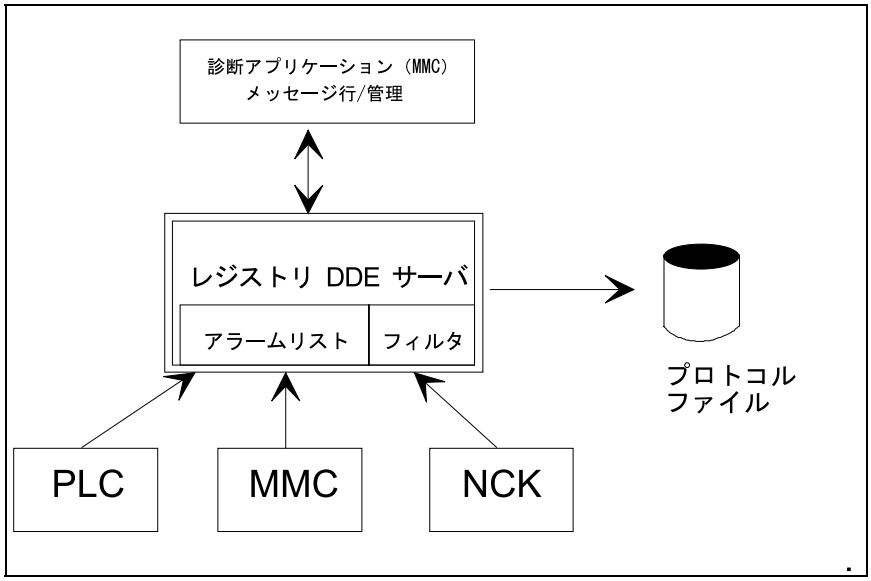
表 3.4 : PI サービスの例

関数グループ	例	意味
NC 関数	_N_DIGION	ディジタル化をオンにする
NC プログラム関数	_N_FINDBL	サーチをアクティブにする
ファイル関数	_N_F_DELE	ファイルを削除する
保護レベル関数	_N_F_PROT	ファイルに保護レベルを割り当てる
ツール関数	_N_CREATO	ツールを作成する
PLC 関数	_INSE	モジュールをアクティブにする

3.2.5 アラームサーバ

アラームサーバ

DDE アラームサーバは，電流アラームとメッセージを MMC に送ります。次の図で概要が説明されています。



システムアラーム／メッセージ

以下のアラーム／メッセージが，システムで発生することがあります。

- NCK アラーム
- ドライブアラーム
- サイクルプログラムアラーム
- PLC アラーム
- PLC メッセージ
- コンパイルサイクルアラーム
- MMC アラーム／メッセージ

特長

DDE アラームサーバは，メッセージに対して次のようなレジストリ機能を備えています。

- 登録アラーム
- 確認アラーム
- 照会機能 :

  最優先アラーム

  2 番目に優先されるアラーム

  アクティブアラームの数

  発生したアラームの数

  アクティブアラームのリスト
- ログファイルの記録

9 章では，アプリケーション固有のアラーム処理を構成する方法について説明されています。

アラームの表示

DDE アラームサーバには，オペレータインタフェースがありません。備えられているのは DDE プロトコル経由のアラーム／メッセージだけです。アラーム／メッセージ用のウィンドウを構成することによって，アラームの表示をアクティブにすることができます。このウィンドウには，たとえば最優先アラームを含む DDE 変数が表示されます。

アラームのログファイル

DDE アラームサーバは現在アクティブになっているアラーム／メッセージのリストを管理します。オプションで，フィルタを通して渡されたアラーム／メッセージの記録も行います。これらのログファイルは診断のために使用することができます。

ログファイルの生成

ログファイルは，次の特長を使用して柔軟に生成することができます。

- Name：ログファイルの名前
- Filter：ログ記録されるアラームの選択特性
- Record：ログファイル内のエントリの数
- RecLen：1 エントリの長さ（バイト数）
- FlushTime：バッファをログファイルにフラッシュする時間枠

3.2.6 データ管理

概説

840DI はハードディスクを備えています。DOS ファイルシステムがこのディスク上のファイルを管理します。

これは，現在 NCK で必要とされていないすべてのデータとプログラムの保管場所です。

Windows システムが提供する機能だけでは NC の要件を満たせないので, MMC ドメインがデータ管理に追加されました。これはデータ管理サーバの機能を使用して実現されます。

これにより，NCK ファイルシステムの機能が MMC 領域に転送されます。

データ管理の利点

MMC アプリケーションがデータ管理から得られる利点には次のようなものがあります。

- 論理関係に応じたデータスキームでデータを構造的に保管できる
- スキームの構造化規則を自動検査できる
- NCK ファイルシステムのデータとハードディスクに保管されたデータに対して共通のビューを持てる
- NCK ファイルシステムを完全システムに透過的に組み込める
- アクセス許可の機密管理
- ファイル名の長さを，25 の使用可能文字に拡張できる
- DOS/Windows 標準アプリケーションからのデータを組み込める

データスキーム

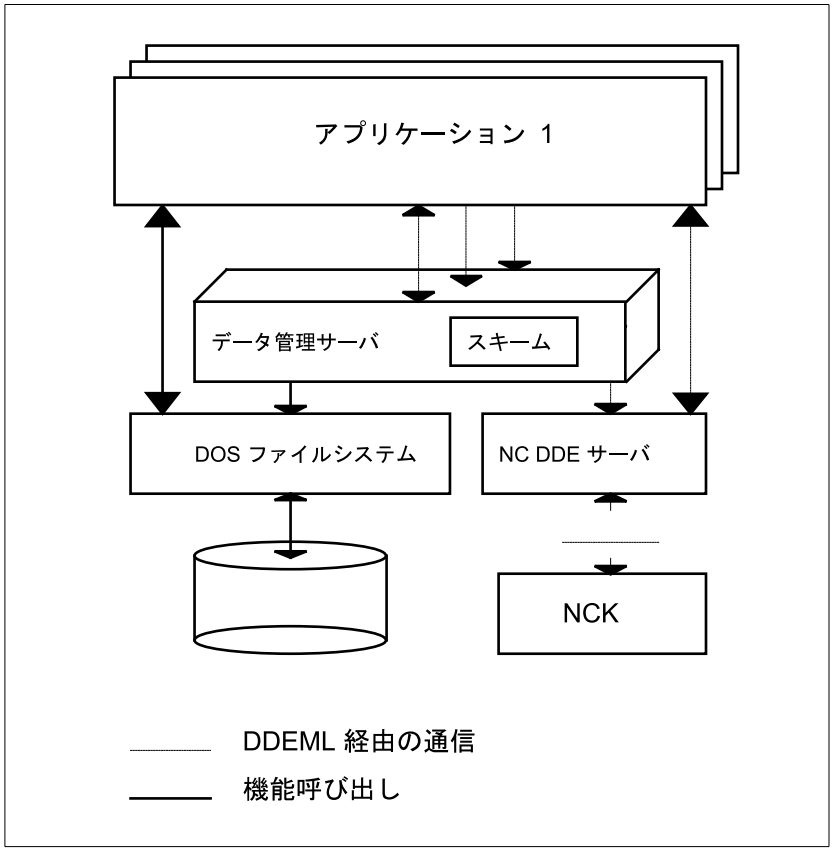
データ管理は，以下のようなデータ構造の輪郭を記述するデータスキームに基づいています。

- データオブジェクトの属性
- データタイプ
- 結び目のタイプ（データタイプの割り当て）
- ディレクトリ内の保管場所

管理される全データの必要情報がデータスキームに保管されます。その他に，追加の管理情報がハードディスクに保管されます。

完全システムのデータ管理

次の図では，データ管理がどのように完全システムに組み込まれるかが示されています。



データに対するデータ管理ビュー

データ管理サーバには，アプリケーションからハードディスク上のユーザデータへのインタフェースが備えられています。サーバから，DOS ファイルシステムにあるデータへの直接アクセスが可能です。また，NC-DDE インタフェース経由で NCK ファイルシステムに含まれるデータにアクセスすることもできます。

このように，完全システムにあるすべてのデータに対して同一ビューが備えられています。

NCK データへの直接アクセス

NC-DDE サーバを経由して，独占的に（MMC から開始して）NCK ファイルシステムにアクセスすることができます。

このデータインタフェースのことが 8 章に記述されています。その章では NC-DDE サーバについて説明されています。

管理情報

データ管理は，DOS ファイルシステムのファイルとディレクトリに対して追加機能を提供しています。例えば次のような機能があります。

- ロングネーム
- アクセス許可
- ユーザクラス

この管理情報は，それぞれのディレクトリで隠しファイルに保管されます。その管理情報には，各ディレクトリとそのディレクトリに含まれる各ファイルに関する次の情報のエントリが含まれます。

- アクセス許可　　読み取り，書き込み，表示，実行，削除。
- 名前　　データ管理の名前。最大 25 文字。
- DOS 名。　　データ管理が作成した DOS ファイル名
- 拡張子　　ファイルの拡張子。3 文字。
- データタイプ ID　　NCK での定義済みの名前，または任意の選択可能な名前として '*'

追加情報の使用

さらに，アプリケーションが，データ管理により提供されるファイルとディレクトリの特性を使用することもできます。機能リストには次のような情報があります。

情報	例
ファイル名	testmac
データタイプ	SMAC
拡張子	SPF
完全名	standard macro
ファイルシステム名	-1testmac
保管場所	活動（NCK），受動（MMC）
サイズ	12345
日付	26.07.1999
アクセス許可	12234

追加情報のないファイル

追加情報が与えられていないファイルまたはディレクトリは，標準 DOS ファイル，標準 DOS ディレクトリとして扱われます。これらのファイルではアクセス許可の管理はなされず，ファイル名は最大で 8 文字です。

初期設定ファイル

中でもファイル DH.INI には，以下のセクションが含まれています（表 3.5）。

表 3.5 ファイル DH.INI のセクション

セクション	意味
DHSTART	ハードディスク上の MMC データ管理のルートディレクトリ
SCEME	データスキームのバイナリ部分の名前

データ管理の特長

データ管理はファイルやディレクトリを処理するためのインタフェースを備えています。

- 作成
- 削除
- コピー
- リスト表示
- ダウンロード（NCK から MMC へのコピー）
- アップロード（MMC から NCK へのコピー），さらに
- アクセス許可の設定
- ヘルプの取得

データ管理サーバは，データ構造をハードディスク（データスキーム）に記述することから開始します。それはシステムに保管され，実行時に読み取ることができます。

データ管理は次の目的のためにこのスキームを使用することができます。

- データスキームで定義した記述に従ってデータを保管する
- デフォルトのアクセス許可を管理し，評価する
- オブジェクトの一般特性を指定する

データ管理のサービスは DDE サーバとして実装されています。

3.3 通信

3.3.1 MPI インタフェース

MPI インタフェースは物理的転送メディアで，以下の構成要素に接続します。

- MMC：オペレータパネル
- MSTT：マシン操作パネル
- NCK：数値制御
- PLC：プログラマブルロジックコントローラ
- PG：プログラミング装置

データ転送率の範囲は，PLC，PG の 187.5 KBit/s から，コントロール 840DI の 1.5 MBit/s までです。

MPI 接続

840DI には, 9 ピン D-Sub メス型コネクタ X4 の MPI インタフェースが備えられています。

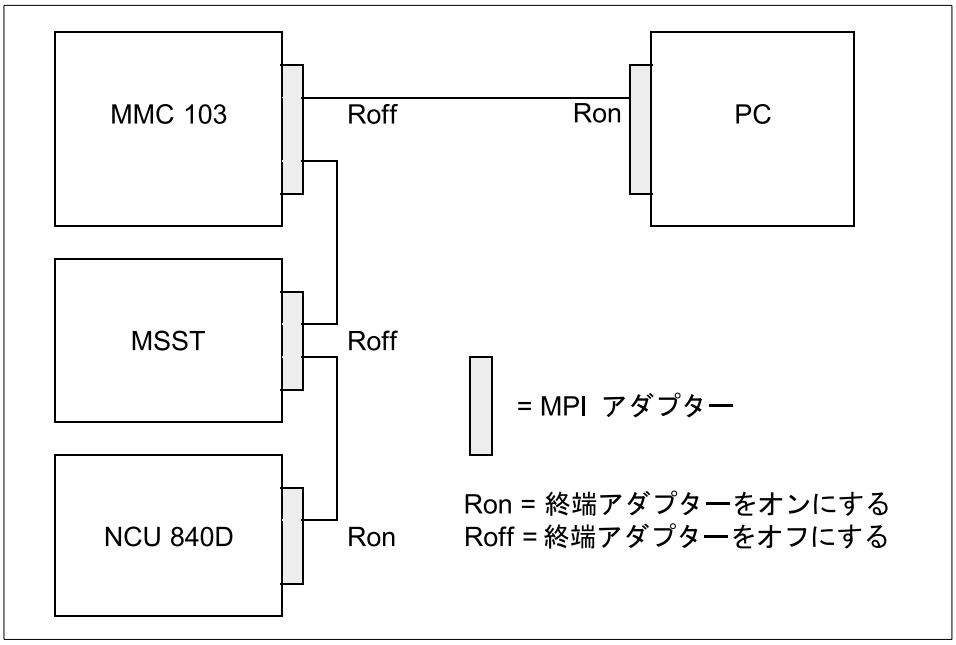
物理的転送メディアは，SINEC L2 に応じた 2 本の保護付きワイヤーケーブルで，最大 200m まで可能です。

配線

ここで配線に関するいくつか重要な支援情報を示します。

バスの両端は信号反射を避けるためターミネイトされている必要があります。少なくとも終端装置のいずれかに 5 ボルトが供給されるようにしてください。（つまり対応するステーションがオンになっていなければなりません）そうしないと，バスの中立電圧が不正確になるからです。

バスセグメント内に有効となっている終端抵抗器が全くない場合があります。その場合，反射を引き起こしてバスの電圧レベルを低めることがあります。



分岐（つまり参加者とバスとを接続するケーブル）はできるだけ短く（2m 未満）するべきであり，それが避けられない場合（つまり PG を接続する場合）にのみ使用すべきです。分岐は，バスコネクタ（バスを通して送られ，アクティブにされる終端抵抗を提供するコネクタ）のみで構成されるようにするのが理想です。使用していない分岐をバスから取り外してください。

ケーブルの構成要素

オリジナルの 840DI ケーブルを使用するようお勧めします。

以下の構成要素を使用してケーブルを自分で作ることができます。

SINEC L2 バスケーブル （長さを指定してください）	注文番号 **6XV1 830 - 0AH10**
バスコネクタ （ケーブルごとに 2 つ）	注文番号 6ES5 762 - 2AA12

構成

NCDDE サーバの開始時には，まだ MPI ドライバがアクティブになっていなければ，それも開始されます。

注 : MPI インタフェースは割り込み要求 IRQ 10 とメモリ領域 CC00H ～ CCFFH を使用します。追加ネットワークボードを使用するときには，このことを考慮に入れなければなりません。
ヒント : 正確に配線し，正しく終端抵抗を設定したにもかかわらず PC に接続する際に問題が生じる場合は，PC が AWARD-BIOS を使用しているかどうかをチェックしてください。要求されたメモリ領域のシャドーイングを行うことにより，AMI-BIOS がうまく機能する場合があります。

プログラミング機器

SIMATIC PG 740 タイプのプログラミング機器には，MPI インタフェースが備えられています。

3.3.2 コントロールへの OEM データの転送

概説

この章では，コントロールへの OEM データの転送についてのみ説明します。

この転送のために使用されるインタフェースは，技術的な装備と，OEM ユーザの個人的な好みに依存しています。原則として備えられているインタフェースは表 3.6 にある通りです。

表 3.6 MMC 103 のインタフェース

インタフェース	ID	コメント
パラレル	X8	外部ネットカードまたは外部の大容量メモリを使用（ストリーマ，JAZ ドライブ，ZIP ドライブなど）
シリアル	X6，X7	ターミナルシミュレーション，PC IN 4，または Interlnk-Intersrv を使用
ネットワーク		PCI/ISA ボックスを使用
フロッピディスク		任意選択のディスクドライブを使用

外部機器のインストール

ユーザの便宜のために新規スタートアップと，オペレータ構成要素 840DI 用のソフトウェアロードルーチンがあります。これについての詳細は，Installation & Start-up Guide （IAD）に記されています。これはシステムの初期化の際に呼び出されます。

構成のすべての変更を，MMC としての操作か，または Windows での操作のいずれかに適用します。両方を使用したい場合は，各構成要素を 2 度インストールする必要があります。

インストール済みソフトウェア

備えられている MMC ソフトウェアは，パラレルインタフェースの VALITEC ストリーマの操作を目的としています。その接続，パラメータの使用，および操作については，Installation and Start-up Guide /IAD/ で説明されています。

シリアルインタフェースは，PCINlight という統合データ転送プログラムでパラメータ化することができます。このプログラムについての詳細は，Operator's Guide /BA/ で説明されています。

シリアルインタフェースを使用するために，Windows95 の［スタート］メニューから，［プログラム］->［アクセサリ］の［ハイパーターミナル］を選択することもできます。

フロッピディスクドライブが X9 に接続されている場合，自動的に認識されます。このディスクドライブの電源はインタフェース X9 によって供給されないことに注意してください。

MMC と他の PC との接続

PCI/ISA ボックス（6FC5 247-0AA02）が，コントロールの内部接続用に必要です。

ISA アダプタに挿入されたネットワークボードと，適合するネットワークソフトウェ

アを使用して，840DI をコンピュータネットワークに接続することが可能です。ネットワークボードとネットワークソフトウェアのインストール，そしてネットワークでの操作についての詳細は，それぞれ対応するマニュアルのインストール指示を参照してください。

3.4 OEM アプリケーション

定義

MMC の機能は, 1 つ以上の OEM アプリケーションによって, カスタム固有に拡張することができます。

OEM アプリケーションは，Regie のダイナミックリンクライブラリ（REGIE.DLL）と関連してのみ実行できるプログラムとして定義されます。

OEM アプリケーションが Regie によって開始されると, OEM アプリケーションはシステムの初期設定ファイル（MMC.INI）に含まれる初期設定を使用することができます。それには MMC のタスクを調整するために必要なシーケンス制御のエレメントが含まれます。

ボリューム

たとえば，OTTO という OEM アプリケーションにいくつかのファイルが含まれています。その意味と内容が表 3.7 にリストされています。

システムへの組み込み

表 3.7 OEM アプリケーションのファイル

ファイル	意味，内容	例
初期設定ファイル	デフォルト値	OTTO.INI
プログラムファイル	実行可能プログラム	OTTO.EXE
テキストファイル	言語に依存するテキスト。たとえば，ソフトキーテキストや表示テキスト。	OTTO_GR.DLL
ダイアグラムファイル	Regie ダイアグラムの管理	OTTO.MDI
状態管理	Regie メニューの管理	OTTO.ZUS

これらのファイルは，表 3.8 にリストする ¥OEM ディレクトリ内のファイルのエントリによって基本システムに組み込まれます。

表 3.8 標準システム内の OEM アプリケーションのエントリ

ファイル	セクション内のエントリ	例
REGIE.INI	TaskConfiguration	Task7=name:=otto OEM アプリケーションがシステムに通知される
REGIE.INI	Version	Version = 5.12 / Date = 26.07.99 OEM 開発者によるバージョン管理
Regie の language.INI	HSoftkeyTexts	HSK7="OTTO" ソフトキー 7 を介してアクセス可能な OEM アプリケーション

構成

OEM アプリケーションは Visual Basic または Visual C++ を使用して構成します。その構成方法については後の章に示されています。

4 グラフィカルユーザインタフェースの設計

この章では，NC の標準インタフェースについて概説します。これは，SINUMERIK 標準インタフェースに倣った操作インタフェースを開発する OEM プログラマにとって役立つはずです。

4.1 標準 NC のユーザインタフェース

概説

840DI のグラフィカルユーザインタフェースは，NC 固有のアプリケーションの以下の 4 つのフィールドで構成されています。

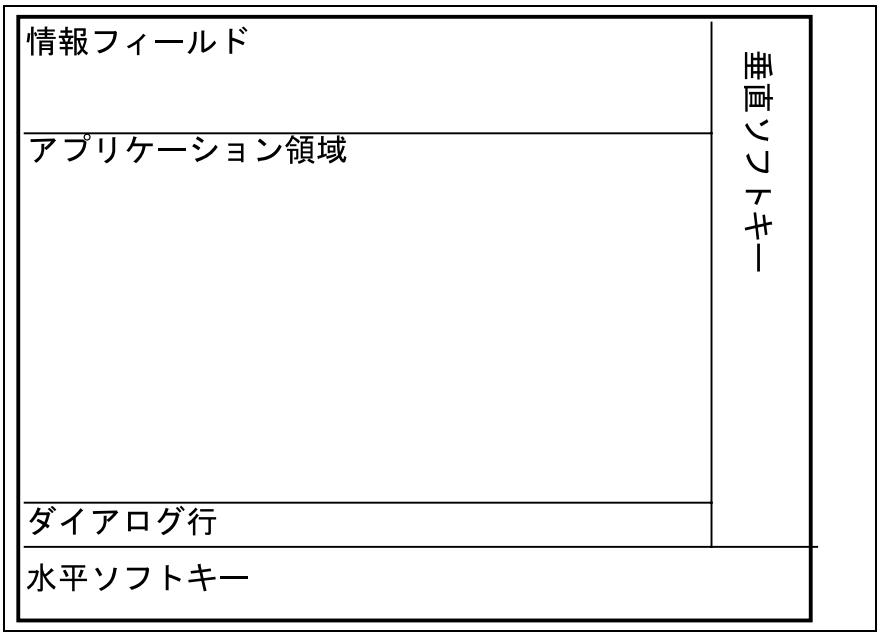


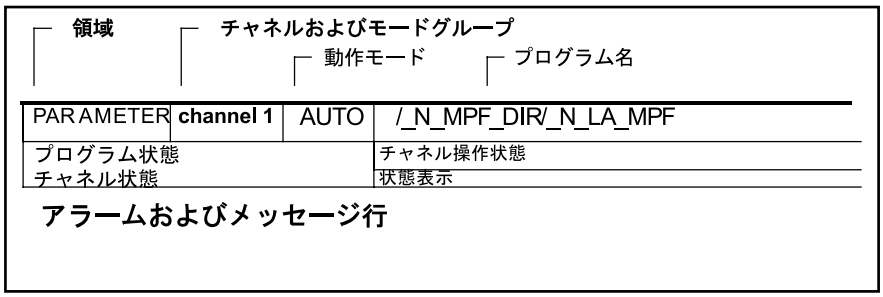
表 4.1 表示画面の構造

フィールド名	内容	例
情報フィールド（ヘッダ）	動作セクション チャネル状態 プログラム状態 チャネルおよびモードのグループの名前 アラームテキスト 動作モード プログラム名 チャネル動作メッセージ チャネル状態の表示	Machine channel reset program is active channel 1 channel 1 block N5 error at circle end position JOG PROGRAM1 Stop: EMERGENCY-STOP active SKP skip block
アプリケーション領域	5 つの作業ウィンドウおよび NC 表示	positions，auxiliary functions，feed
対話式フィールド	オペレータ用のメモの入ったダイアログ行，メニューバーに対応する情報	entered value exceeds upper limit ^ recall is possible
水平ソフトキー	8 個の水平ソフトキーのラベル	PARAMETER
垂直ソフトキーバー	8 個の垂直ソフトキーのラベル	REPOS

4.1.1 情報フィールド（ヘッダ）

配置

表 4.1 で概説されているヘッダの要素は，下記の図のように配置されています。



内容

含められる内容については，/BE/ Operator's Manual を参照してください。

OEM アラーム

このセクションに OEM ユーザアラームを表示できます。詳細な情報は，9 章「アラーム」で扱われています。

情報フィールドの隠蔽

Windows のフルスクリーンアプリケーションの場合など，OEM ユーザはこのフィールド（ヘッダとも呼ばれる）を隠すことにする場合もあります（6 章「Regie」を参照）。

警告：

情報フィールドを隠していると，ユーザインタフェースにアラームが表示されなくなります。
それでもアラームが表示させるようにするには，アラームサーバを使用する別個の OEM 定義アプリケーションを使用して行います。

4.1.2 アプリケーション領域

概説

アプリケーション領域には，アプリケーション固有のウィンドウを導入できます。このフィールドは，制限なしで OEM の処理に自由に配置できます。

サイズ

画面の解像度は，640 × 480 ピクセル（標準 VGA）です。560 × 325 ピクセルは作業フィールド用に予約されています。

表示域の開始位置とサイズは，定数を使用して設定します。

BeArttop	上限

BeArtleft	左限
BeArtwidth	幅
BeArtheight	高さ

これらはファイル ALSTART.FRM で定義されます。

警告：

このファイルは変更しないでください！

ウィンドウの数

アプリケーション領域では，最大 8 個までのウィンドウを定義できます。

ウィンドウの構造

原則として，アプリケーションウィンドウの設計はユーザ次第ですが，840DI 環境内で見た目を統一するために，以下の設計をお勧めします。

ウィンドウはタイトルバーと表示セクションから構成されています。通常 Windows アプリケーションで使用できる，サイジングコントロール（最大化，最小化，クローズボタン）は提供しません。

タイトルバー

タイトルバー用に推奨されているパラメータを，以下に示します。

表 4.2 タイトルバー用のパラメータ

要素	パラメータ	VB の ID
フォーカスが置かれていない背景色ウィンドウ	gray	&H00808080
フォーカスが置かれている背景色ウィンドウ	yellow	&H0000FFFF
テキストカラー	white	&H00000000
テキストフォント		MS Sans Serif

表示セクション

表示セクション用として推奨されているパラメータ：

表 4.3 表示セクションのパラメータ

要素	パラメータ	VB の ID
背景色	light gray	&H00C0C0C0
固定テキスト	background gray text black	VB Label

テキスト枠（入力ウィンドウ用）	background white text black	VB Textbox
テキスト出力	left aligned text black	VB Label
切り替えフィールド	options to be selected left aligned text black	VB SSOption
ON/OFF または YES/NO フィールド	check box	VB SSCheck
線	line style	3-D なし
フォーカス（アクティブウィンドウ）	window border yellow	&H0000FFFF

4.1.3 対話式フィールド

概説

対話式フィールドでは，以下の 2 つのタイプの情報を提供します。

- メニューバーに適用される情報
- オペレータ用のメモが入る対話式の行

メニューバーに適用される情報には，3 つのシンボルが含まれています。

それらの意味については，下記の表 4.4 でリストされています。

表 4.4 メニューバーに適用される情報

シンボル	意味
^	再呼び出しが可能です。上のメニューレベルへ戻ります。
i	このメニューに関する追加情報を，INFO キー i を使用して入手することができます。

その機能が実際に選択できる場合のみ，それぞれのシンボルが表示されます。

追加情報

これらのシンボルのほかに，OEM ユーザは Visual Basic を使用して，独自のアプリケーション固有のシンボルを作成することもできます。

ダイアログ行

ダイアログ行は，オペレータを誘導するのに使用されます。これは，ソフトキー機能と密接に関連しています。モーダルダイアログは，指示されている操作に関する別の方法について説明します。

その内容は，シーケンス制御のコンテキストに応じてプログラムされます。詳細については，7 章「シーケンス制御」を参照してください。

4.1.4 ソフトキーバー

それぞれ 8 個ずつの水平と垂直のソフトキーが，アプリケーションで使用できます。

ただし，水平ソフトキーは 8 個以上構成することができます。8 個目の水平ソフトキーの右側に矢印が表示されます。ラベル，機能，およびアクセス権限を指定できます。詳細については，7 章の「メニューツリー生成プログラム」を参照してください。

4.2 追加アプリケーションの組み込み

Regie のフレームにアプリケーションを組み込む際に，構成ツールを活用することができます。これは，サービスメニューから開始します。これは，ファイル regie.ini および re_XX.ini のセクション TaskConfiguration を変更します。6 章「Regie」では，さらに詳細な説明が行われています。

どのアプリケーションを組み込めるか？

以下の種類のアプリケーションを組み込むことができます。

- シーケンス制御を伴う独自のアプリケーション
- シーケンス制御を使用しない Windows アプリケーション
- DOS アプリケーション

シーケンス制御を使用しないアプリケーションに関する注意事項

シーケンス制御を使用しないアプリケーションを統合する場合，以下の制限を考慮しなければなりません。

- コントロールには，通常マウスおよび MF2 キーボードは含めません。それでもアプリケーションは操作できますか？この点で，ファンクションキー [F9] ～ [F12] （[Shift]，[Alt] キーを押す場合と押さない場合）が特に重要です。
- 組み込まれたアプリケーションを，ユーザコマンドで終了できる場合，黒い画面が表示されます。この場合，まず領域選択を再度アクティブにしなければなりません。
- 特に高速の DOS アプリケーションは，表示メモリに直接書き込むことにより，オペレーティングシステムをバイパスします。840DI はこの動作をサポートしていません。アプリケーションが以下のように動作するかどうかは容易に判別できます。
  - アプリケーションを組み込む
  - Regie を開始する
  - アプリケーションを選択する
  - 別の領域へ変更する
  - 元に戻す
  - 表示は正しいか？
- Regie はメインウィンドウを使用して，アプリケーションの識別を行います。変更はありませんか？タイトルはどうですか？タイトルバーにパラメータやファイル名を表示するプログラムには，特に注意してください。
- 840DI のオペレータインタフェースを変更すると，コントロールは当社サービスの対象ではなくなってしまう場合があります。このサービスの対象から外れないようにするには，コントロールが特定の基準を満たしていなければなりません。それらの基準については，SERVICE- Guideline で要約されています。担当の販売員から最新のリリースを入手することができます。また，対応する契約についてもご説明いたします。

4.3 OEM パッケージを使用したアプリケーションの作成方法

概説

OEM パッケージを使用することにより，840DI の標準アプリケーションに適合しており，マシンオペレータが直観的に操作できるような独自のアプリケーションを OEM で作成することができます。これを行うために，いわゆるシーケンス制御がプロジェクトに組み込まれます。Regie-DLL と接続するだけで実行可能なプログラムのことを，OEM アプリケーションと呼びます。Regie から始めた場合，MMC.INI の初期設定を使用することができ，OP031 のソフトキーを評価することができます。これには，MMC でタスクを調整するのに必要な，シーケンス制御の要素が含まれています。

ボリューム

OEM アプリケーションには以下のファイルが含まれています。

- プログラムファイル（例，OTTO.EXE）
- 初期設定ファイル（例，OTTO.INI）
- 言語依存テキスト（例，OTTO_GR.DLL）

  このファイルには，少なくともアプリケーション固有のソフトキー用のテキストが含まれています。

  それに加えて，アプリケーションが表示するあらゆるテキストを含めることができます。それらには，*LoadString*（VB ディレクトリツリーの winapi31.hlp で説明されています）がアクセスします。
- ピクチャ管理（例，OTTO.MDI）
- 状態管理（例，OTTO.ZUS）
- シーケンス制御のファイル

ファイル拡張子さえ固定されていれば，ファイルにはどのような名前も使用できます。それでも，わかりやすくするために，拡張子との区切りになるドットの前の名前の部分は，すべてのコンポーネントで同じにするようにお勧めします。

アプリケーション用のフレームワークの作成

この手順の詳細については，7.2 章で説明されています。

VB デバッガによるテスト

開発 PC のテストを行う際，以下の点を調べてください。

OEM パッケージがインストールされているディレクトリに，ドライブ文字 L を割り当てます（例，subst l: d:¥oempaket）。

MMC OEM パッケージのインストール中に，ファイル *mmc.ini local* 内の変数 *NcddeMachineName* がローカルに設定されていなければ，その設定を行います。

ファイル *mmc.ini* 内の変数 *NCDDEStartupFile* を，NEW 命令を含んでいる NSK ファイルに設定します。MMC 操作中に，LINK 命令がこの場所に置かれます。

アプリケーションを開始する前に，プログラムを検査するために必要なサーバ（NCDDE サーバ，DH サーバ，MBDDE サーバなど）を開始してください。

NCK が接続されていない場合，サーバの機能が制限されるので注意してください。制限の詳細は以下のとおりです。

- NCDDE は，ユーザーが提供した値しか戻せません（DDE テストにより動的に，または NSK ファイル内で静的に）。
- データ管理では，ローカルファイルしか認識されません。NCK ファイルを宛先指定するコマンドはエラーを戻します。

（regie16.dll からの）Regie 機能は，タスク固有でしか処理されません。たとえば，領域転換についての情報は与えられません。

デバッグにかける労力には，それに見合うだけの価値があります。1 つ 1 つ徹底的にテストされたアプリケーションだけが，Regie に有効にロードできることに気付かれるでしょう。

4.4 必要に応じた標準アプリケーションの改造

840DI で提供されている各アプリケーションに関して，対応している .ini ファイル内でエントリを作成し，追加の状態を構成することができます。したがって，アプリケーション自体を変更せずに，ユーザピクチャを付け足すことができます。

この点に関して，アプリケーション固有の INI ファイルに，新しく 3 つのセクション（[MATRIX]，[CHILDS]，および変更された状態ごとに追加セクションを 1 つ）を追加しなければなりません。

オリジナルのアプリケーションに戻すには，名前が **SwitchToParent** で始まる Regie 機能（説明については，6.4 を参照）を使用してください。

Regie.ini で開始するアプリケーションを，23 より大きいタスク番号に割り当ててください。

セクション MATRIX

ここでは，アプリケーションのどの状態が変更されるかを指定します。行 UPS0 から UPS15 までを使用して，最大 16 個の状態を変更できます。

構文 :

[MATRIX]
UPSx=state:newSection{Softkeys}

ここで,

State	変更する状態の番号 これは，アプリケーションの .ZUS ファイルを調べれば分かります。
NewSection	新しい状態を記述するために, .INI ファイルに新しいセクションを追加しなければなりません。 そこには，新しいセクションの名前を入れます。
Softkeys	機能を変更したいソフトキー

例 :

```
[MATRIX]
USP1=10:otto{0,3,4,9}
```

状態 10 の，1 番目，4 番目，および 5 番目の水平ソフトキー，および 2 番目の垂直ソフトキーが上書きされます。新しい値がセクション [Otto] で定義されています。

セクション CHILDS

ここでは，ファイル regie.ini にあるセクション [Task-Configuration] のエントリの記号名を指定します。

構文 :

```
[CHILDS]
symbol=number
```

ここで,

Symbol	この .INI ファイルだけで使用されている記号名
Number	ファイル *regie.ini*, セクション *TaskConfiguration* で宛先指定されているアプリケーションのタスク番号。タスク番号には，23 より大きい番号を指定しなければなりません。そうしなければ，言語ファイルでソフトキーテキストが入力されていなかった場合に，Regie はエラーを示します。タスク番号に 23 以下を使用する場合，PreLoad :=False を指定して，タスクを構成してください。

例 :

```
[CHILDS]
HINZ=8
KUNZ=17
```

HINZ は領域アプリケーション 8 に，KUNZ は領域アプリケーション 17 に宛先指定します。

新しい状態のための新しいセクション

ここでは，変更を定義します。ソフトキーテキストの上書きや regie.ini からのタスクの呼び出しを行ったり，状態自体によって状態をさらに定義することができます。

構文 :

```
[newsection]
softkey=text>function
```

ここで,

softkey	ソフトキー ID	
text	引用符で囲まれているテキスト	ソフトキーに直接書き込まれるテキスト
	#<*nr*>h	番号 *nr* の水平ソフトキーテキストのテキスト
	#<*nr*>v	番号 *nr* の垂直ソフトキーテキストのテキスト
	#<*nr*>a	番号 *nr* の一般テキストのテキスト
function	{Child}	アプリケーションの記号名 （セクション CHILDS で定義されているもの）

{Child}:'cmdline'	cmdline では, アプリケーション Child に使用する引き数を追加できます。引き数の前にはコロンを付けて，引用符で囲みます。パラメータの形式および番号は，呼び出されるアプリケーションに完全に依存しています。

例 :

```
[OTTO]
0="blabla">{HANS}
3=#5h>{EMIL}
4=#7h>{HINZ}
9=#11a>{KUNZ}:'clock'
```

左側の水平ソフトキーには，*blabla* というラベルが付けられます。*hans* という記号名のアプリケーションを開始できるようになります。どのタスク索引に宛先指定されるかは，セクション [CHILDS] に記述されています。

ソフトキー 3 も同様の働きをしますが，水平ソフトキーのブロックから 5 番目のテキストを使用して，ラベルが付けられます。

ソフトキー 4 は，7 番目の水平ソフトキーのテキストを使用します。これは，記号名 *hinz* が割り当てられているアプリケーションを開始します。対応するタスク索引は，セクション [CHILDS] に記録されています。

ソフトキー 9（2 番目の垂直ソフトキー）には，11 番目の一般テキストが割り当てられています。*kunz* という記号名のアプリケーションが開始されます。引き数 *clock* が渡されます。

4.4.1 領域 MASCHINE への OEM ソフトキーの組み込み

ステップ 1

以下の内容のファイル "Maschine.ini" を，ディレクトリ C:¥OEM に作成します。

```
[MATRIX]
UPS1=1:JUMP{0}

[JUMP]
0="OEM_Test">{RUN_TEST}

[CHILDS]
RUN_TEST=24
```

これらの設定は，OEM ディレクトリに保管され，ソフトウェアの更新時にも失われません。

ステップ 2

以下の内容のファイル "Regie.ini" を，ディレクトリ C:¥OEM に作成します。

```
[TaskConfiguration]
Task24=name := oemframe, cmdline := "C:¥¥OEM¥¥CUSTOMER.EXE",
TimeOut := 60000, PreLoad := False
```

MMC を再起動すると，OEM ソフトキーに割り当てられているアプリケーションが開始されます。

要約

ラベルテキスト "OEM_Test" の OEM ソフトキーが，領域アプリケーション ”Machine” の基本画面の最初のソフトキーとして使用されるように構成されました（*.ZUS ファイル内で状態 1)。

これで，アプリケーション CUSTOMER.EXE は，この OEM ソフトキーを使用して開始できるようになりました。

4.5 標準インターフェース（regie.ini，mmc.ini）の改造

標準システム（ディレクトリ MMC2 にある）を参照する，ファイル MMC.INI，REGIE.INI，および RE_xx.INI（ディレクトリ ADD_ON，OEM，および USER にある）の OEM 拡張機能は，以下の優先順位で Regie の起動時に集められます。

MMC2 (固定システム設定)
ADD_ON
OEM
USER

注 : ファイル MMC.INI，REGIE.INI，および RE_xx.INI は，対応する既存のものに対する差分として取られます（つまり，その情報が集められます）。*.DLL ファイルの場合，検出された最新のものが有効になります。*.NSK ファイル内のエントリは，ディレクトリ ¥USER 内の USER.NSK でしか作成できません。

注 : 上記の理由で，ユーザは自分の異なるエントリを，OEM パス "REGIE.INI" および "LANGUAGE¥RE_XX.INI" に入力しなければなりません。OEM パス内のエントリは，標準 MMC2 パス内のエントリを上書きします。

フランス語

フランス語を使用する場合，以下の内容のファイル MMC.INI を OEM パスに作成します。

例 4-1 OEM パス内の MMC.INI

```
[LANGUAGE]
Language=FR
Language2=GR
```

MPI 接続なしの MMC103 インターフェースの構成

NC を使用せずに MMC103 を使う場合, OEM パス内のファイル MMC.INI に, 以下の行を追加してください。

例 4-2 OEM パス内の MMC.INI

```
[GLOBAL]
NcddeMachineName=local
NcddeDefaultMachineName=local
NcddeMachineNames=
NcddeStartupFile= ncdde202.nsk
```

標準 Windows アプリケーションの組み込み

以下の例では，標準 Windows アプリケーションの電卓が組み込まれます。これを行うには，"regie.ini" というファイルを作成し，以下の行を入力します。

例 4-3 OEM パス内の REGIE:INI ファイル

```
[TaskConfiguration]
Task6=name := oemframe, cmdline := "calc", Timeout := 0,
```

```
ClassName := "SciCalc", HeaderOnTop := False,
PreLoad := False
```

さらに，ソフトキーテキストの構成も行わなければなりません。"re_gr.ini" というファイルを作成し，以下の行を入力します。

例 4-4 OEM¥LANGUAGE パス内の RE_GR.INI

```
[HSoftkeyTexts]
HSK6="Calcu- lator" //20
```

4.6 アプリケーション aeditor

MMC ソフトウェアパッケージには，他のアプリケーションに子アプリケーションとして組み込むことのできる，テキストエディタが含まれています。

これを行うには，以下の 3 つのステップを実行しなければなりません。

regie.ini への追加

Regie に子アプリケーションを認識させなければなりません。*構成ツール*（スタートメニューにある）を使用して，以下のエントリを作成します。

```
[TaskConfiguration]
...
Task35 = name := aeditor, Timeout := 5000
```

開始されるまでエディターがロードされないようにする場合，以下のように追加します。

```
PreLoad := False
```

終了時にエディターが閉じられるようにする場合，

```
TerminateTasks := aeditor:35
```

を，呼び出し側のアプリケーションのエントリーに追加します。

複数のアプリケーションからエディターを呼び出す場合，セクション [TaskConfiguration] にそれぞれ別個のエントリを作成しなければなりません。

```
; Task1 using aeditor:35, Task2 using aeditor:36
Task35 = name := aeditor, Timeout := 5000
Task36 = name := aeditor, Timeout := 5000
```

エディタに関するコマンド行の作成

呼び出しの構文を以下に示します。

<cmd_line> ::= <full_path>,<display_text>,<type>,<access>

<full_path> : 任意の有効な DOS パス。既存のディレクトリでなければなりません。ファイルが存在しなければ，作成されます。

<display_text> : 最大 40 文字の任意の文字列

<type> ::= DOM | FILE

DOM = ドメインパス（P3.1 では実装されていません）

FILE = ファイルパス

<access> ::= RW | RO

RW = 読み取り書き込み

RO = 読み取り専用

Visual Basic での例 :

```
Dim sCmdLine As String
```

```
Dim sFileName As String
' get file name from file list box flbBox
sFileName = flbBox.Path + "¥" + flbBox.List(flbBox.ListIndex)
' open file with read/write access, display full name
sCmdLine = sFileName + "," + sFileName + ",FILE,RW"
-or –
'open file with read only access, title "scratch file"
sCmdLine = sFileName + ",scratch file,FILE,RO"
```

このコマンド行でのエディターの開始

Visual Basic では，コマンドのシーケンスは以下のようになります。

```
Const CHILD_INDEX_EDITOR = 35
Dim nRet As Integer
nRet = WriteCmdLineVB(CHILD_INDEX_EDITOR, sCmdLine)
Call SwitchToChild(CHILD_INDEX_EDITOR)_
```

5　システムの構成

この章では，.INI ファイルについて要約します。これらのファイルは，標準的な MMC のいくつかのアプリケーションによって評価されます。シーケンス制御では，エントリの一部はグローバル変数として備えられています。

MMC 間で設定が異なるアプリケーションでエントリを評価する場合は，この章を参照する必要があります。

標準装備のアプリケーションの専用初期設定ファイルについては取り上げません。詳細については，標準的な資料を参照してください。

以下のファイルは 840DI でいくつかのアプリケーションに使用されます。これらのファイルはディレクトリ **¥mmc2** にあり，変更バージョンはディレクトリ **¥oem** および **¥user** に入れられます。

実行時には，シーケンス制御の章に記述されている以下の関数

AL*GetPrivateProfileString*

AL*GetprivateProfileInt*

AL*WritePrivateProfileString*

を使用して，この種のファイルにアクセスする必要があります。

5.1 DH.INI

データ管理では，構成依存パラメータとして開始時のディレクトリの 1 つだけが使用されます。さまざまな NCU のデータセットを保持する場合や，開発システム上でシナリオをテストする場合などは，このエントリに変更を加えると役立つことがあります。

セクション	エントリ	意味
DHSTART	mmchome	MMC データ管理の ROOT ディレクトリ
SCEME	sceme	データスキームのバイナリ部分の名前

5.2 MBDDE.INI

このファイルは，アラーム処理をパラメータ設定するのに使用してください。

セクション	意味
Alarms	アラームのリストに関する一般情報。 例 : レジストリ項目の日時形式
TextFiles	アラームテキストのリストのパスとファイル名。 例 : ディレクトリ mb（レジストリモジュール）内の MMC アラームテキストの場合，MMC=..¥dh¥mb.dir¥alm_
Helpcontext	ヘルプファイルのパスと名前。 例 : File0=hlp¥alarm_
DEFAULTPRIO	各種のアラームタイプの優先順位のデフォルト定義。 例 : POWERON=100
PROTOCOL	ログファイルの特性。 ログファイルのパスと名前。 例 :file=.¥proto.txt
KEYS	アラームをクリアするのに使用できるキーに関する情報。 例 : Cancel=+F10。 この場合，Shift+F10 の組み合わせでアラームをクリアできます。

これらのエントリについては，9 章で説明されています。

5.3 MMC.INI

このファイルには，ユーザ固有の設定がすべて含まれます。カラーと言語依存の詳細情報を設定し，ハードウェア（V24 インタフェースの設定）を構成できます。NCU に対する割り当てが固有の場合は，仕様はすべてこのファイルにあります。固有でない場合は，詳細情報を含むファイル名があります。

セクション	意味
BTSS settings	操作パネルのマシンデータ。 例 : データの表示精度 例 :R パラメータのアクセスレベルの変更
colors	VGA ドライバのカラーのセットアップ
Control	画面の解像度
DateTime	各種言語の時刻形式
Services	追加情報。 例 : FloppyDisk=a:（フロッピーディスクドライブの名前）
DIRECTORIES	カタログの定義。 例 : AlDir=D:ALTMP（シーケンス制御の一時ディレクトリ）
Europe	セクション LANGUAGE に含まれているフォントからのフォント選択
Function	インタフェースの記述
GLOBAL	NcddeDefaultMachineName NcddeMachineName NcddeMachineNames NcddeMmcName NcddeServiceName NcddeStartupFile NCServerName による，特定の NC の特性
LANGUAGE	2 つの言語とそれらのフォントの選択。 例 : GR Europe
LOCAL	開発コンピュータの CNC コンポーネントのアドレス。 例 : ADDRESS1=/PLC,0
mbdde	アラームサーバの名前
NCU840D	CNC コンポーネントのアドレス。 例 : ADDRESS1=/PLC,10000d01
net	いくつかの NCU を 1 つの操作パネルに接続するための初期設定ファイルの名前。例 : NETNAMES=netnames.ini
Remote	リモート診断用パラメータ
SIM0	シミュレーション用の CNC 構成要素の名前とアドレス
TIS	ツール識別システム。 例 : コードキャリアデータのテキスト終結
ToolMgmt	ツール管理。 例 : ツールデータベースの名前
V24-Conf_n	シリアルインタフェースを構成する n 番目のデータセット（n=0 ～ 2）
ViewTree	表示制御 ViewTree に関する情報

5.4 NETNAMES.INI

MMC と NCU の組が一対一に対応する場合は，ファイル netnames.ini は評価されません。このファイルは実際のハードウェア構成を表していなければなりません。1 つの MMC 上でこのファイルの編集を完成させてから他の MMC にコピーして，エントリの所有者だけを変更するという処置が勧められています。

セクション	エントリ	値	意味	注釈
own	owner	MMC_1	所有名	
conn MMC_1				このセクションは MMC 当たり 1 つずつあります。
	conn_1	NCU_1	MMC からアクセスできる NCU を定義します。	NCU 当たり 1 つのエントリ
param network	bus	btss		
param MMC_1				このセクションは MMC 当たり 1 つずつあります。
	mmc_address		所有するバスアドレス	S7DPMPI.INI に対応していなければなりません。
param NCU_1				このセクションは NCU 当たり 1 つずつあります。
	nck_address		NCU_1 という名の NCU バスアドレス	
	plc_address		PLCon NCU_1 のバスアドレス	
	name		NCU_1 の名前	

5.5 OEMFRAME.INI

このファイルには，シーケンス制御なしの個々のアプリケーションが画面に表示される際のサイズを指定できます。

パラメータを格納するアプリケーションごとに別々のセクションを，対応するプログラムファイルと同じ名前にして，ファイル名拡張子を付けずに入力する必要があります。

5.6 REGIE.INI

このファイルは，MMC で使用する関数を定義します。独自の OEM アプリケーションをここに入力してください。構成ツールによりサポートされます。

セクション	エントリ	意味
Miscellaneous	RemoveIcons	開発サポート : スタートアップ時にタスクアイコンが表示されます。
	ShowMessageBox	開発サポート : スタートアップ時に生じる警告がメッセージボックスとして表示されます。
	HelpTaskIndex	情報ボタンの分岐先のタスクの数
	ShowResources	開発サポート : システム負荷が表示されます。
	PoweronTaskIndex	スタートアップ後に表示されるタスクの数
	MMCSignOfLife	テスト専用 : MMC により毎秒 DB19,DBW16 に変更が加えられます。
SystemDllConfiguration	DllNamex	開始される WINDOWS モジュールのリスト
MMCDllConfiguration	DllNamex	開始される SINUMERIK モジュールのリスト
StartupConfiguration	Startupx	バックグラウンドで実行されるサーバのリスト
TaskConfiguration	Taskx	フォアグラウンドタスクのリスト

5.7 S7DPMPI.INI

S7DPMPI.INI 内のエントリは，ユーザインタフェースにより，'installation' エリア内でのみ変更を加える必要があります。32 ビットのバスドライバを使用している場合は，これらのエントリは .INI ファイルに格納されず，レジストリデータベースに格納されるようになりました。

セクション	エントリ	意味	注釈
S7ONLINE			
	DRIVER	使用されるドライバ	
	PROFILE	使用されるインタフェースの記述	
	HW_DEVICE	使用されるインタフェース	
S7MPISPC2		インタフェースの記述	
	BAUDRATE	ボーレート	2 = 187,5 kBit 3 7 =1,5 MBit
	HWINT_VECTOR	ハードウェア割り込み	MPI インタフェースの事前設定されているハードウェア割り込みと一致していなければなりません。
	TS_ADR	バスアドレス	mmc.ini をそれぞれ netnames.ini と一致させなければなりません。
	S7 プロトコルのその他のパラメータ		

注 : Windows ディレクトリーの “S7DPMPI.INI” ファイルだけが考慮されます。

6 Regie

Regie は，補助プログラムおよび領域アプリケーション（タスク管理）を柔軟に管理するためのプログラムです。Regie アプリケーションは，次の処理を行います。

- システムの初期化
- システムの起動
- システムおよび MMC のダイナミックリンクライブラリのロード
- 正しい順序でのアプリケーションの始動
- システム構成
- 領域の切り替え

Regie のパラメータは，初期化データファイルから提供されます。このデータファイルの構成ツールが用意されており，これを使うと，新規アプリケーションを簡単に追加することができます。

Regie の最大管理能力は以下のとおりです。

- 32 の補助アプリケーション
- 32 の領域アプリケーション
- 64 のシステムダイナミックリンクライブラリ（DLL）
- 64 の MMC ダイナミックリンクライブラリ

6.1 Regie の概念

概説

Regie は，以下を柔軟に管理するための制御プログラム（Regie コマンドモジュールの場合は REG_CMD.EXE）です。

- 補助プログラム（基本モジュール）
- 領域アプリケーション
- ダイナミックリンクライブラリ（DLL）および VBX ファイル

これらを図 6.1（抜粋）に示しています。

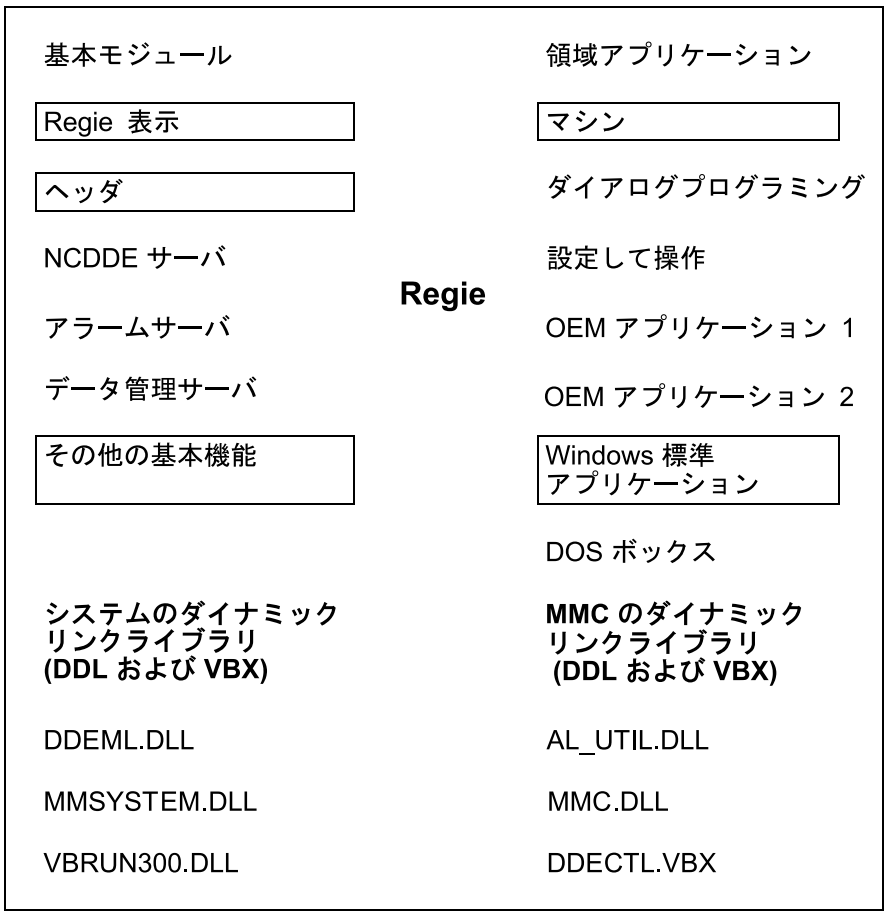


図 6.1 Regie での，領域アプリケーションから補助プログラムおよび DLL へのリンク

アプリケーション Regie は以下を行います。

- システムの初期化
- システムの起動
- システムおよび MMC のダイナミックリンクライブラリのロード
- 正しい順序でのアプリケーションの始動
- システム構成
- 領域の切り替え

開始順序

補助アプリケーション（DLL，VBX）は，領域アプリケーションより先に Regie によって開始されます。アプリケーションの開始後，そのアプリケーションの初期化が終了したことが示されるまで Regie は待機状態になります。

これで，必要な補助プログラムは，必要時に備えて使用可能になります。

Regie によって開始されたアプリケーションがクローズされるときも，これに似たメカニズムがインプリメントされます。

Regie の最大能力

Regie の最大管理能力は以下のとおりです。

- 32 の補助アプリケーション
- 32 の領域アプリケーション
- 64 のシステムダイナミックリンクライブラリ（DLL，VBX）
- 64 の MMC ダイナミックリンクライブラリ

OEM に関して

Regie は，お客さまの OEM アプリケーションも管理します。OEM ユーザは，データファイル REGIE.INI を編集して，該当するアプリケーションをシステムに対して宣言することができます。

領域アプリケーション

Regie では，領域アプリケーションを次の 3 種類に分けます。

- VB で作成され，シーケンス制御に組み込まれた標準 OEM 領域アプリケーション
- お客さまの，Windows または DOS を使用する PC 用のアプリケーション
- Windows または DOS を使用する標準アプリケーション（たとえば EXCEL や EDIT）

ファイル

上記のような領域アプリケーションを実現するために，Regie では，次の 5 つのファイルを使用します。:

- REGIE.INI
- REGIE.DLL
- Regie-language.INI（例 : RE_GR.INI）
- OEMFRAME.INI
- MMC.INI

注 : ソフトウェアリリース 4 以後，ファイル REGIE.INI，MMC.INI，および RE_XX.INI は，ディレクトリ MMC2，ADD_ON，OEM，USER の順に検索されて評価されます。
1 章 23 ページの「新しいディレクトリ構造」も参照してください。

注 : このような理由から，OEM ユーザは，REGIE.INI と LANGUAGE¥RE_XX.INI に関する各自の相違点を OEM パス内だけに入力しなければなりません。つまり，OEM パスへの入力によって，標準 MMC2 パス内のものは上書きされます。

REGIE.INI

このファイルは，他の Windows アプリケーションからも認識される初期化ファイルです。

Regie.ini は，通常のテキストエディタ（EDIT や WRITE など）で編集できるテキストファイルです。

このファイルは，いくつかのセクションで構成されています。それらは表 6.1 に示されており，6.3 で詳しく説明しています。

REGIE.DLL

ファイル REGIE.DLL は次のような特長を備えています。

- タスクの開始元のディレクトリを判別します。
- タスクの現在の言語 .DLL にアクセスします。
- タスクの初期化が終わったかどうかを示します。
- モノクロディスプレイの明度を調節します。

REGIE.DLL の機能は，VB と VC++ で呼び出すことができます。これらの機能は，実行時にリンクされます。

language.INI

データファイルの言語 .INI（たとえば，ドイツ語の場合は RE_GR.INI）には，Regie のソフトキーラベルと一般テキストが入っています。このファイルは，6.3 で説明した通常のテキストエディタを使って編集することができます。

OEMFRAME.INI

必要であれば，このファイルで置き換え用のアプリケーション OEMFRAME についてパラメータを設定して，Windows アプリケーションを組み込むことができます。これについては，6.6 で詳しく説明しています。

MMC.INI

このデータファイルについては，5 章で説明しています。

6.2 OEM アプリケーションの組み込み

概説

アプリケーションがどのように MMC に組み込まれるかは，そのアプリケーションのタイプによって異なります。:

- OEM 領域アプリケーション（VB で作成され，シーケンス制御に組み込まれたもの）
- お客さまの，DOS または Windows を使用する PC 用のアプリケーション
- DOS または Windows を使用する標準アプリケーション

OEM アプリケーション

OEM アプリケーションは，開発されてからシステムに組み込まれます（7.2 を参照）。

Windows プログラム

Windows プログラムのユーザインタフェースと機能が使用可能です。このプログラムは，プレースホルダアプリケーション OEMFRAME によってシステムに組み込むことができます。このためには，構成ツールを使うか，または ASCII エディタでファイル REGIE.INI を編集します。その手順については，6.6 で詳しく説明しています。

DOS プログラム

DOS プログラムのユーザインタフェースと機能は事前定義されています。これらプログラムは，プレースホルダアプリケーション OEMFRAME によってシステムに組み込むことができます。このためには，以下を行います。:

1. ファイル REGIE.INI と言語 .INI のエントリ
2. Windows への戻りキーの定義
3. PIF ファイルの生成
4. ファイル SYSTEM.INI へのデバイスドライバ VMMC2D.386 の組み込み（まだ組み込まれていない場合のみ）

この手順は，6.7 で詳述しています。

注 **:**DOS プログラムは，全画面表示アプリケーションとしてしか実行できません。ヘッダとソフトキー領域は表示されません。

6.3 Regie の初期化ファイル： REGIE.INI，language.INI，および OEMFRAME.INI

要約

Regie の初期化ファイルは次のとおりです。

- REGIE.INI
- Regie-language.INI（例 : RE_GR.INI）
- OEMFRAME.INI.

Windows の .INI ファイルの一般的な構造の詳細については，WINDOWS Programmer´s Reference Manual /WPR/ を参照してください。

REGIE.INI，Sprach.INI

Regie で開始しようとするすべてのアプリケーションは，ファイル REGIE.INI 内で構成しなければなりません。

初期化ファイルの言語 .INI（たとえば，ドイツ語の場合は RE_GR.INI）には，Regie のソフトキーテキストと一般テキストが入っています。どちらのファイルについても，この章で説明しています。

OEMFRAME.INI

初期化ファイル OEMFRAME.INI には，必要時に標準 Windows プログラムを組み込むために利用できる置き換え用のアプリケーション OEMFRAME に関する情報が入っています。このファイルについては，6.6 で説明しています。

6.3.1 初期化ファイル REGIE.INI

REGIE.INI.

Regie で開始しようとするすべてのアプリケーションは，ファイル **REGIE.INI** 内で構成しなければなりません。

ファイル REGIE.INI は，標準の ASCII テキストエディタ（EDIT など）で編集できるテキストファイルです。

このファイルは，表 6.1 で示すセクションで構成されます。

表 6.1 ファイル REGIE.INI のセクション

セクション	機能
Version	Regie の日付とソフトウェアリリース番号。OEM ソフトウェアエンジニアが，リリース別に管理するために使用します。
SystemDllConfiguration	システムのすべての DLL および VBX データファイル（DllName0 = DDEML.DLL ～ DllName63 = VBRUN300 など）DLL
MMCDllConfiguration	MMC 103 のすべての DLL または VBX データファイルのリスト（DllName0 = AL_UTIL.DLL ～ DllName63 = DCTL.OCX など）

StartupConfiguration	領域アプリケーションより先に起動される補助アプリケーションの入ったリスト（例 : NCDDE サーバ）
TaskConfiguration	システム初期化時に起動される領域アプリケーション（属性付き）の入ったリスト（例 : 機械の操作）
Miscellaneous	Regie で使用される以下のパラメータが入ったリスト。 ExitButtonExitButtonIndexHelpTaskIndexMMCSignOfLifePoweronTaskIndexRemoveIconsShowMessageBoxShowResources
CommandToTask	コマンド解釈チャネルに関する情報

開始順序

Regie はまず，SystemDllConfiguration セクションと MMCDllConfiguration セクションの補助プログラム（DLL，VBX）を始動してから，StartupConfiguration セクションの補助プログラムを始動し，さらに TaskConfiguration セクションの領域アプリケーションを始動します。各プログラムの始動の正常完了が知らされるまで，そのつど Regie は待機状態になります。

これにより，必要が生じた時点で補助プログラムをすぐ使用できることが保証されます。

Regie によって開始されたアプリケーションのクローズでも，これに似たメカニズムが使用されます。

以降の項では，REGIE.INI の各セクションについて説明します。

■ 6.3.1.1 ［Version］セクション

目的に合わせてファイル REGIE.INI を変更することができます。たとえば次のように，［Version］セクションにバージョン管理情報を付け加えることができます。

例 6-1 ［Version］セクション

```
[Version]
Version = 5.21 ; specifies the version, set by the OEM
developer
```

■ 6.3.1.2 ［SystemDllConfiguration］セクション

これは，すべてのシステムのダイナミックリンクライブラリ（DLL）と Visual Basic の拡張子（VBX）のリストです。ここに入れる有効なエントリは，64 個を超えることはできません。

Windows システムの特性上，必要なすべての DLL をこのセクションに入れることをお勧めします。システムの起動時に，Regie は，ここで構成されているすべての DLL をロードします。

DLL を編成するには，次の 2 とおりの方法があります。:

- パス指定を使わない入力。DLL は，検索パスのディレクトリ内になければなりません。
- パス指定を使う入力。DLL は，指定のディレクトリから始動されます。

例 6-2 ［SystemDllConfiguration］セクション

```
[SystemDllConfiguration]
DllName0 = DDEML.DLL
; DLL must be located on the search path and depends on the WINDOWS
 release
DllName1 = C:¥MYSYSTEM¥GRID.VBX
; the entry does not depend on the WINDOWS release
```

注 : DLL は，ファイル拡張子を付けて指定しなければなりません。ファイル名だけでは固有名にならないからです（例 : NAME.DLL または NAME.VBX）。

■ 6.3.1.3 ［MMCDllConfiguration］セクション

これは，MMC のダイナミックリンクライブラリ（DLL）と Visual Basic の拡張子（VBX）のリストです。これは，MMC2 システムディレクトリに置かれていなければなりません。パス仕様ディレクトリを使うことはできません。そのため，各 DLL（VBX ごとの）は，他と重複しないで個々のソフトウェアリリースに割り当てられます。このリストに入れる有効なエントリは，64 個を超えることはできません。

Regie は，このセクション内で構成された DLL を，サーバアプリケーションおよび領域アプリケーションより先にロードします（また，クローズ時にはアンロードも行います)。領域アプリケーションは，LoadLibrary や FreeLibrary などの呼び出しに注意する必要はありません。

例 6-3 ［MMCDllConfiguration］セクション

```
[MMCDllConfiguration]
DllName0 = MMC.DLL ;DLL is loaded automatically during
system
                              start-up.
```

注 : 現在まで有効であったエントリ
`DllNameXX = DDEFIFO.VBX`
に注釈のマークを付けます。この制御は，リリース 3.2 以後はもう使えないからです。

注 :DLL は，ファイル拡張子を付けて指定しなければなりません。ファイル名だけでは固有名にならないからです（例 : NAME.DLL または NAME.VBX）。

■ 6.3.1.4 ［StartupConfiguration］セクション

概説

［StartupConfiguration］セクションには，領域アプリケーションより先に始動される予定の補助アプリケーションをすべてリストする必要があります。どのアプリケーションにも，そのアプリケーションの特性を説明した 1 つ以上の属性（1 つずつコンマで区切る）を追加しなければなりません。

エントリ識別子

各アプリケーションは，エントリ識別子（startup2 など）で付けられた番号順に始動

されます。

たとえば，

```
Startup2 = Name := HEADER
```

というエントリを使って構成されたアプリケーションがあるとすると，それは，

```
Startup1 = Name := SERVER
```

というエントリを使って構成された始動対象のアプリケーションがシステム内にすでに存在していることを示します。

［StartupConfiguration］セクションの OEM 用のエントリ

OEM アプリケーションは，Regie.ini ファイルの［StartupConfiguration］セクション内の Startup12 ～ Startup24 までの領域に入力しなければなりません。そうしないと，MMC 103 標準システムと対立する可能性があります。

属性

表 6.2 は，［StartupConfiguration］セクション内のエントリの属性の概要を示しています。

表 6.2 ［StartupConfiguration］セクションのエントリの属性

属性	意味
Name	ファイルマネージャにリストされているアプリケーションの名前（拡張子 .EXE なし）
Timeout	初期化の最長時間（ミリ秒数）。デフォルト値は 10000（10 秒）です。
CmdLine	クライアントアプリケーションのパラメータ。ASCII ストリング。
ShowTask	アイコンの表示／非表示のパラメータ。TRUE または FALSE。

Name 属性 :

ファイルマネージャに表示されるとおりのアプリケーション名（つまり属性名に拡張子 .EXE を付けないもの）。

例 6-4 Name 属性

```
[StartupConfiguration]

Startup5 = Name := MBDDE ; correct

Startup6 = Name := MBDDE.EXE ;W R O N G
```

Timeout 属性 :

Regie でのアプリケーションの初期化に対する最大許容時間（ミリ秒）。初期化が終了したら，アプリケーションは，REGIE.DLL の機能 InitComplete を呼び出して Regie に知らせなければなりません。

指定された最大許容時間内にメッセージ InitComplete を受け取らないと，Regie はシステムエラーを示します。その場合，その後システムが正常に起動するとは限りません。

重要な注意事項 **:** この time-out 値を低く設定しすぎてはなりません。MMC の処理量が非常に大きい場合，待ち時間がきわめて長くなることがあるからです。

この属性は任意選択です。省略した場合，Regie でのデフォルト値は 10 秒です。

処理可能な最大長は 9 桁です。

注 :InitComplete の呼び出しを挿入できない（たとえば，補助プログラムのソースファイルが使用できない）場合，Timeout 値として 0 が使われます。

CmdLine 属性 :

任意選択の **CmdLine** 属性では，クライアントアプリケーションにパラメータを指定することができます。

Show-Task 属性 :

デフォルトでは，補助アプリケーションはアイコン化されて始動され，画面の表示領域外に移動させることができます（つまりアイコンが画面の更新を妨げない）。補助アプリケーションを表示する必要のある場合，ShowTask 属性を使わなければなりません（たとえば HEADER）。

例 6-5 ShowTask 属性

```
[StartupConfiguration]

Startup0 = name := shutdown, Timeout:= 15000

; this application may take up to 15 seconds to indicate its
execution

Startup1 = name := header, ShowTask := TRUE

; this application may take up to 10 seconds (preset value) and
re-mainson the screen.
```

■ 6.3.1.5 ［TaskConfiguration］セクション

概説

［TaskConfiguration］セクションには，すべての領域アプリケーションと，Regie によって始動される他のすべてのアプリケーションが入っています。各アプリケーションごとに，特殊特性を記述する 1 つ以上の属性（1 つずつコンマで区切る）を指定することができます。

エントリ

各アプリケーションは，Task エントリ識別子（Task5 など）で指定された順序で始動されます。たとえば，以下のようなエントリで構成されたアプリケーションがあるとします。

```
Task5 = Name := IB
```

また，以下のようなエントリで構成された別のアプリケーションがあるとします。

```
Task2 = Name := DIENSTE
```

注 : 例外が 1 つあります。**Poweron TaskIndex**（通常は Task0 が使われます）を指定して［**Miscellaneous**］セクションに入力されたアプリケーションは，システムの起動時には最後に始動されます。これは，起動時の不要な領域切り替えを行うことを避けるためです。

［TaskConfiguration］セクションの OEM 用のエントリ

Regie.ini ファイルの［TaskConfiguration］セクション内の 0 ～ 23 の範囲は，領域メニューによる活動化が可能なタスク用に予約されています。残りの番号（24 ～ 63）は，いわゆる子アプリケーション用に予約されています。そのうち，51 ～ 63 をお客さまの OEM 子アプリケーションに使用することができます。

Regie のソフトキー

同様にして，ソフトキーの割り当てもタスクエントリで設定します。その場合，最小番号の付いたタスクをマシン領域キー上に置き，Task1 をソフトキー 1 の上に置き，その後 Task23 までを，ソフトキーバーの 3 番目の拡張子内の正しいキーの上に順に置きます。

ソフトキーのラベル

言語 .INI ファイルは，ソフトキーラベルを指定します。各言語ごとに，サブディレクトリ LANGUAGE 内に固有の言語 .INI(たとえばドイツ語では RE_GR.INI）が用意されています。言語 .INI 内にソフトキーテキストが入力されている場合のみ，アプリケーションをソフトキーで開始することができます。

属性

表 6.3 は，タスク構成のエントリをリストしています。

表 6.3 : タスク構成の属性

属性	意味
name	ファイルマネージャで使われているアプリケーションの名前（拡張子 .EXE なし）
Timeout	始動に要してもよい最大時間 (ミリ秒単位)。デフォルトは 10000(10 秒) です。
CmdLine	クライアントアプリケーションのパラメータ。OEM フレーム内のアプリケーション名
HeaderOnTop	ヘッダが活動または非活動のどちらの状態かの表示
PreLoad	始動時のアプリケーションのロード
TerminateTask	アプリケーションの始動時の，メモリからの他のアプリケーションの除去
DosBox	DOS プログラムに対する領域アプリケーションのマーク付け
AccessLevel	アクセス権のレベル
ClassName	アプリケーションのクラス名
WindowName	アプリケーションのウィンドウ名
ShowTask	アイコンの表示または非表示。デフォルト値は TRUE

GIMMEKEYS	領域アプリケーションが処理する Regie のキーのマスクの使用可能化
ShowAppMenu-Key	領域アプリケーションが使用する領域切り替えキーのマスクの使用可能化。

Name 属性

ファイルマネージャで使われるアプリケーションの名前（指定の拡張子 .EXE なし）

Timeout 属性

Regie でのアプリケーションの始動に対する最大許容時間（ミリ秒）。詳細については，前章を参照してください。

CmdLine 属性

CmdLine 属性を使って，領域アプリケーションのコマンド行でクライアントアプリケーションのパラメータを指定することができます。デフォルト設定は空ストリング "" です。

これは静的割り当てです。

REGIE.DLL の WriteCmdLine 関数を使って，コマンド行パラメータをタスクに動的に割り当てることができます。そのパラメータは，ReadCmdLine 関数で読み取ることができます。これについては，第 6.4.6 章に説明があります。

DOS および Windows の標準アプリケーションについては，タスク名として置き換え OEMFRAME を入力すると，標準の DOS プログラム名または Windows タスク名が CmdLine 属性に渡されます。

例 6-6 CmdLine 属性

```
[TaskConfiguration]
Task2 = Name := OEMFRAME, CmdLine := "EXCEL.EXE TAB.XLS"
; the spread sheet program EXCEL is called by OEMFRAME
```

ClassName 属性と WindowName 属性

Regie（および OEM フレーム）がクライアントアプリケーションを処理できるのは，そのクライアントアプリケーションが所有者ウィンドウハンドルを認識している場合のみです。このハンドルは Windows アプリケーションにおける固有ハンドルであるとは限らない（Windows アプリケーションは複数の所有者ウィンドウハンドルを持つことができます）ので，ClassName 属性と WindowName 属性別に REGIE.INI ファイルに指定する必要があります。:

例 6-7 ClassName 属性と WindowName 属性

```
[TaskConfiguration]Task2 = Name := OEMFRAME, CmdLine := "WinWord",_
ClassName := "OpusApp",_ WindowName:= "Microsoft Word"; Calling the
text processing program WinWord by ClassName and; WindowsName.
```

ソースコードとして使えるクライアントアプリケーションの場合，この情報に直接アクセスすることができます。これは，RegisterClass および CreateWindow という 2 つの API 呼び出しで見つける必要があります。

ヒント : ソースコードを使えない場合，上記のストリングは，WinWalk などのパブリックドメインツールを使て確かめることができます。

注 : 基本的に，領域アプリケーションは WinExec（lpszCmdLine, SW_SHOW) という命令で開始します。
詳細について，Programmer´s Reference Manual, Volume 2: Func-tions"/WPR/ の中の WinExec のパラメータ lpszCmdLine の説明を参照してください。

画面の暗転と砂時計

注 : REGIE.INI にアプリケーションを入力した後，画面が黒色に変わって砂時計が表示されたならば，SIEMENS シーケンス制御はそのアプリケーションにリンクされ，そのアプリケーションは OEM フレームを介して追加で開始されているということです。
対処策 : OEM フレームなしで REGIE.INI にアプリケーションを直接入力します。

HeaderOnTop 属性

この任意選択属性を使って，領域アプリケーションのアプリケーションのヘッダ（操作領域や操作モードなどを表示します）を非活動（HeaderOnTop = False）または活動（HeaderOnTop = True）のどちらにするかを指定することができます。

デフォルト設定は True です。つまり，活動解除されていない限りヘッダは表示されます。

例 6-8 HeaderOnTop 属性

```
[TaskConfiguration]
Task5 = Name := MACHINE, HeaderOnTop := True
; application machine with header

Task6 = Name:= DP, HeaderOnTop := False
; dialog programming without header

Task7 = Name := aeditor
; ASCII editor with header (default header active).
```

PreLoad 属性

この任意選択属性を使って，MMC の始動時に領域アプリケーションが開始されないようにすることができます。

この属性を設定しない（PreLoad := False）場合でも，領域アプリケーションは，初めて活動化されるまで開始しません。

利点 :

MMC システムの始動を高速化することができます。

欠点 :

MMC の始動時に，

すべての構成済み領域アプリケーションを実行時に処理できるかどうかを判断できません（リソースの能力)。

デフォルト設定は PreLoad := True ですが，その場合，この属性を指定されていないアプリケーションは，始動時に開始されます。

例 6-9 PreLoad 属性

```
[TaskConfiguration]
Task5 = Name := SIMULA, PreLoad := False
; The operating area Simulation (SIMULA) was configured in the
; REGIE.INI, but will not be started during start-up.
```

注 : 開発およびテストの場合は，この属性を False に設定してください。安全上の理由から，リリースされているバージョンは，始動時にすべての操作領域を始動する必要があります。デフォルト設定ではこのことを考慮に入れています。

AttributeTerminateTasks:.

この属性を使って，他の領域アプリケーションの始動時にメモリから特定の領域アプリケーションを除去することができます（例 6-10）。そうすれば，Windows リソース不足の問題が解決されます。

例 6-10 アプリケーションの除去

```
[TaskConfiguration]
Task6 = name := param, TerminateTasks:= dp
; Calling PARAM.EXE removes the application DP.EXE from the memory.
```

一度に複数のアプリケーションを除去する場合，該当する名前を 1 つずつコンマで区切って丸括弧で囲みます（例 6-11）。

例 6-11 複数の領域アプリケーションの除去

```
[TaskConfiguration]
Task7 = name := dp, TerminateTasks := (param, dg)
; Calling DP.EXE removes the applications PARAM.EXE and DG.EXE
; from the memory.
```

同一名の付いた 2 つのアプリケーションを相互排他にするには，コロンと，続けて対応するタスク番号を付けたものを付け加えます（例 6-12)。

例 6.12 2 つのアプリケーションの交互実行

```
[TaskConfiguration]
Task8 = name := oemframe, cmdline := "EXCEL.EXE",
_ TerminateTasks := oemframe:12
Task12 = name := oemframe, cmdline := "NOTEPAD.EXE",
_ TerminateTasks := oemframe:8
; Two applications are running alternately under OEMFRAME: either
; EXCEL or NOTEPAD. If OEMFRAME with EXCEL as task 8 is loaded,
; OEMFRAME with NOTEPAD as task 12 is removed from the memory
; and vice versa
```

複数のアプリケーション（その一部は異なるタスク番号で活動中）を一度に除去したい場合，それぞれの名前を 1 つずつコンマで区切って丸括弧で囲み，必要があれば例 6-12 に示されているとおりに，タスク番号を付け加えます（例 6-13）。

例 6-13 活動中の複数のアプリケーションの除去

```
[TaskConfiguration]
Task3 = name := dienste, TerminateTasks:= (rh, aeditor:25)
Task7 = name := aeditor, PreLoad := False, TerminateTasks := rh
Task25 = name := aeditor, PreLoad := False, TerminateTasks := rh
; The program 'ASCII editor' (aeditor) is started more than once
e.g.
; by 'services' as task 7 and by'programming' as task 25. If
; 'services' is called as task 3, the Regie only deactivates the
ASCII
; editor for 'programming' (task 25), however not the editor for
; 'services' itself (task 7). RemoteHelp (rh) will be deactivated
in
; any case.
```

DosBox 属性

DOS プログラムを領域アプリケーションとして処理するには，これが置き換え OEMFRAME のクライアントアプリケーションであることを注記し，そして DosBox 属性を True に設定しなければなりません。

この区別が必要なのは，Windows アプリケーションまたは DOS アプリケーションのどちらの処理であっても，ファイル名だけでプログラムを評価できないからです。デフォルト値は DosBox := False に設定されています。DOS プログラムを領域アプリケーションとして使用したい場合に実行するステップは，6.7 に説明されています。

例 6-14 DosBox 属性

```
[TaskConfiguration]
Task2 = Name := OEMFRAME, CmdLine := "DOSAPPL.PIF", DosBox := True
; Now the DOS application is noted as an area application..
```

AccessLevel 属性

この属性を使って，Regie におけるタスクのアクセス許可レベルを設定することができます。リリース 3.2 以後，Regie を処理するソフトキーにアクセス権が追加されました。

表 6.4 Regie の処理のためのソフトキー

［Machine］	［Parameter］	[Program]	［Services］	［Diagnosis］	［Set-up］

アクセス許可レベルのデフォルト値は，AccessLevel := 4 です。
アクセス許可の指定可能なレベルを表 6.5 に示してあります。デフォルト値は，グレーの背景色で表示しています。

表 6.5 アクセス許可の 8 つのレベル

アクセス	必要な許可	ユーザグループ
S0	システムパスワード	SIEMENS
S1	MTB パスワード	工作機械メーカ
S2	システムパスワード	セットアップ／サービスのスタッフ（工作機械メーカ）

アクセス	必要な許可	ユーザグループ
S3	ユーザパスワード	特権ユーザ（社内サービス）
S4	キー切り替え位置 3	プログラマ
S5	キー切り替え位置 2	上級オペレータ
S6	キー切り替え位置 1	オペレータ
S7	キー切り替え位置 0	中級オペレータ（NC の始動／停止，操作パネル）

例 6-15 AccessLevel 属性

```
[TaskConfiguration]
Task2 = Name := ib, AccessLevel := 2
; Only the set-up staff may access the area application ‘start-up’
by
; entering the MTB password.
```

ClassName 属性

領域アプリケーションとして Windows プログラムを置き換えアプリケーションに組み入れる場合は，ClassName 属性と CmdLine 属性とを共に使います。例は，CmdLine 属性の説明の項で示しています。

WindowName 属性

領域アプリケーションとして Windows プログラムを置き換えアプリケーションに組み入れる場合は，WindowName 属性と CmdLine 属性とを共に使います。例は，CmdLine 属性の説明の項で示しています。

Show-Task 属性

PER デフォルトアプリケーションが，始動時に表示されます。アイコン形式でアプリケーションを始動したい場合，ShowTask := FALSE 属性を使います。この場合，ソフトキー操作でアプリケーションを始動できなくなります。

キーボードフィルタのカスタマイズ

領域アプリケーションごとに，領域切り替えキー，チャネル切り替えキーなどの「MMC キー」をカスタマイズするには，これ以降で説明する拡張子を使用します。

これにより OEM アプリケーションは，たとえば，F10 キーを独自に処理したり別のキーを使用したり（ShowAppMenuKey でパラメータを指定）して，領域切り替えを開始することができます。

次のように，新規の設定項目 GIMMEKEYS および ShowAppMenuKey を使って，REGIE.INI ファイルの［TaskConfiguration］セクションに必須設定値を入力します。

```
...
[TaskConfiguration]
TaskX = name := oemframe, ..., GIMMEKEYS:= n,
ShowAppMenuKey := m
```

パラメータ n および m はビットマスクです。属性と共にその意味について説明します。

GIMMEKEYS 属性 :.

これは領域アプリケーションが処理する Regie のキーの使用可能化マスクです。
GIMMEKEYS:=n,

この場合，32 ビットのビットマスクとして n が使われています。n は，アプリケーションが独自に処理することになる REGIE キーを定義します。

bit0 : 領域切り替え

1=OEMApp は F10 を処理することを指示しますが，0=OEMApp は F10 を標準どおり処理することを指示します（領域切り替え）。

bit1 : チャネル切り替え

1=OEMApp は F11 を処理することを指示しますが，0=OEMApp は F11 を標準どおり処理することを指示します。

bit2 : 取り消しキー（BigMac）

1=OEMApp は ESC を処理することを指示しますが，0=OEMApp は ESC を標準どおり処理することを指示します。

bit3 : マシン領域キー

1=OEMApp は SH-F10 を処理することを指示しますが，0=OEMApp は SH-F10 を標準どおり処理することを指示します。

bit4 : タブキーの代わりの End キー

End キーを押したとき，1= では End キーが OEMApp に渡され，0= ではタブが OEMApp に渡されます。

例 :

GIMMEKEYS:=15 OEMApp は，F10，F11，ESC，SH-F10 を独自に処理することを指示します。

GIMMEKEYS:=1 OEMApp は，F10 を独自に処理することを指示します。

Show-AppMenuKey 属性 :

アプリケーションが使用する領域切り替えキーの使用可能化マスク。

ShowAppMenuKey:=m,

m は，32 ビットのビットマスクです。これは，該当するアプリケーション内でどのキーが（F10 に代わって）領域切り替えを活動化するかを指定します。

bit0 ～ 7 は，定義する領域切り替えキーの仮想キーコードです（winuser.h VK_xxx エントリを参照）。

bit16 が，1= の場合は Shift キーを押す必要があり，0= の場合は押す必要はありません。

bit17 が，1= の場合は Ctrl キーを押す必要があり，0= の場合は押す必要はありません。

bit18 が，1= の場合は Alt キーを押す必要があり，0= の場合は押す必要はありません。

例 **:**

ShowAppMenuKey := 65659

65659 = 0x1007B VK_F12 = 0x7B

→ Shift ＋ F12 を押すと，該当するアプリケーションの領域切り替え機能が起動されます。

VK_F1 = 0x70, VK_F1 = 0x71, ..., VK_F24 = 0x87

注：ShowAppMenuKey を使って領域切り替えに別のキーを割り当てても，F10 は作動します。これを避けるには，GimmeKey を明示的に指定します。

■ 6.3.1.6 ［Miscellaneous］セクション

概説

［Miscellaneous］セクションで，Regie のいくつかのパラメータを設定することができます。表 6.6 に概要を示しています。この表に入っていないエントリは，内部使用専用です。

表 6.6 ［Miscellaneous］セクション内のエントリ

エントリ	意味
ExitButton	ExitButtons を活動化します。
ExitButtonIndex	ソフトキーを ExitButton と定義します。
HelpTaskIndex	置き換えタスク Rhelp を割り当てます。
MMCSignOfLife	MMC の活動モニタリングの印。
PoweronTaskIndex	電源オン後の最初のタスクの索引。
RemoveIcons	画面からアイコンを除去します。
ShowMessageBox	始動時に警告を表示します。
ShowResources	システムリソースを表示します。

ExitButton エントリ

Regie の水平ソフトキーバー（表 6.7 を参照）のソフトキーのうちの 1 つを，ExitButton 用に予約することができます。つまりそのソフトキーを使って，MMC システムのシャットダウンを制御して実行することができます。補助アプリケーションと領域アプリケーションはすべて，オープンされたときとは逆順にクローズされます。

これは，テスト目的の機能です。たとえば，別のバージョンの MMC システムを始動することができます。

表 6.7 Exit 機能を使って Regie を操作するためのソフトキー

[Machine]	[Parameter]	[Program]	[Services]	[Diagnosis]	[Set-up]	[Exit]

ExitButtonIndex エントリは，Exit 機能をソフトキーに割り当てます。ExitButton を使って，この機能を活動化したり（例 6-16）非活動化したり（例 6-17）することができます。

Exit は Regie を使って，MMC システムの終了を制御して実行します。

例 6-16 ［Exit］ボタンをオンにする

```
[Miscellaneous]
ExitButtonIndex = 7 ; the right hand horizontal softkey is used
ExitButton = True ; as exit key
```

例 6-17 ［Exit］ボタンをオフにする

```
[Miscellaneous]
ExitButtonIndex = 7 ; the right hand horizontal softkey is not
ExitButton = False ; used as exit key
```

ExitButtonIndex エントリ

Exit 機能の場合，Regie の水平ソフトキーバー（表 6.7 を参照）は，ExitButtonIndex エントリで割り当てます。

例 6-18 ExitButtonIndex エントリ

```
[Miscellaneous]
ExitButtonIndex = 6 ; The softkey Exit(to the right next to the
; Set-Up key)is defined as Exit key.
```

注： ソフトキーの番号付けは 0 から始まります。

HelpTaskIndex エントリ :

その他の情報 : 次のように，置き換え Rh（リモートヘルプ）を使って，Windows の標準アプリケーション WinHelp を，領域アプリケーションとして MMC システムに統合することができます。

例 6-19 領域アプリケーションとしてのリモートヘルプの定義

```
ó·:
[TaskConfiguration]
Task3 = Name := RH, Timeout := 10000, HeaderOnTop := False
; Rhelp is defined as fourth area application
```

通常，置き換えを活動化するには，［Info］キーを使用するか，または標準キーボードの F12 機能キーを使います。

置き換えは，次のように HelpTaskIndex エントリを使ってタスクに割り当てられます。

例 6-20 ヘルプタスクとしてのリモートヘルプの定義

```
[Miscellaneous]
HelpTaskIndex = 3 ; this refers to the example before
[TaskConfiguration]
Task3 = Name := RH, Timeout := 10000, HeaderOnTop := False
```

SwitchToHelpTask を使って暗黙の領域選択を行うには，必ずこのエントリがなければなりません。SwitchToHelpTask は，以下を単純化した表現です。

```
SwitchToTask(HelpTaskIndex)
```

注 **:** 理論上は，どのタスクでも HelpTaskIndex に割り当てることができます。［Info］キーは，一種の HotKey または ShortKey を表しますが，これにより，領域を直接（Regie ソフトキーバーを経由してアナログから領域キー［Machine］をたどらずに）選択することができます。

MMCSignOfLife エントリ

MMC の活動確認のモニタリング。これを使って，OEM プロジェクトの開発に役立つと思われる機能を使用可能にします。:
PLC 変数（データモジュール DB19，データワード DW16）は，1 秒ごとに増やされます。

例 6-21 MMCSignOfLife エントリ

```
[Miscellaneous]
MMCSignOfLife = False ; The sign-of-life monitoring not active.
```

注 : この機能は，開発用としての使用だけを意図しています。これは，リリースされたバージョンには使えません。

RemoveIcons エントリ

システムの始動時に，Regie は，表示領域で始動されているタスクのアイコンを移動させます。これによって，領域の切り替えの場合に，背景のアイコンによって画面変更が妨げられないという利点があります。

しかし，テストや開発の段階ではこれは不利な場合もあります。それで次のようにして，Regie のこの機能を構成することもできます。

例 6-22 RemoveIcons エントリ

```
[Miscellaneous]
RemoveIcons = False ; For testing purposes the icons remain in the
; visible screen area.
```

PoweronTaskIndex エントリ

その他の情報 : システムの起動の後，［TaskConfiguration］セクション内に構成されているタスクは，番号の小さいものから順に開始されます。

PoweronTaskIndex に入力されたアプリケーション（通常，task0 が選択されます）のみが，始動時に最後に開始されます。これで，不要な領域切り替えを行わずに済みます。PoweronTaskIndex に入力されたアプリケーションは，始動後に活動化されるタスクです。

以下の領域「Installation」（Task3）の例で示すとおり，PoweronTaskIndex エントリを使って，ユーザは領域「machine」より先の別のアプリケーションに進むことができます。

例 6-23 PowerOnTaskIndex エントリ

```
[TaskConfiguration]
Task0 = name := machine ; machine
Task1 = name := param ; parameters
Task2 = name := dpmill ; dialog programming milling
Task3 = name := ib ; set-up
; Usually the application 'machine' is selected after start-
up.

[Miscellaneous]
PoweronTaskIndex = 3
; This entry enables you to activate the application 'set-
up' (task3)
; immediately following Power On.
```

ShowMessageBox エントリ

エントリ ShowMessageBox=True を指定すると，古い DDL がまだロードされたままかどうかを調べることができます。

その他の情報 : ダイナミックリンクライブラリまたはアプリケーション（現在の MMC ディレクトリにあるものだけ）がすでにロードされている場合，MMC システムの初期化時にオペレータに対して警告（Windows メッセージ・ボックス）が出されます。特に，MMC システムを繰り返しロードした場合にこの警告が出されます。

これは，使用しているダイナミックリンクライブラリが正しくアンロードされていないか，またはこの前にアプリケーションがクラッシュしたことを示します。そのような場合，通常はそのアプリケーションはデスクトップには表示されなくなりますが，Windows カーネルの内部タスクリストには存在し続けます。

ダイナミックリンクをアンロードするかまたはアプリケーションを終了するかを尋ねるメッセージボックス内の質問に対して yes と応答してください。この応答でしか，有効なソフトウェアバージョンへのアクセスは保証されません。

注 : この関数はテスト用であるので，ShowMessageBox = True は，OEM アプリケーションの開発時に使用してください。

どうしてもこの警告を抑止する必要がある場合，フラグを False に設定しなければなりません（その必要があると思われるのは，たとえば，配布予定のバージョンリリースの場合です)。その場合，このエントリは以下のようになります。:

例 6-24

```
[Miscellaneous]
ShowMessageBox = False
; This is recommended for released versions only, the DLLs
and
; applications will not be unloaded.
```

ShowResources エントリ :

その他の情報 : Windows 3.11 では，使用可能な Windows システムリソースは限られているため，アプリケーションの開発時には使用したリソースの記録を取っておく必要があります。F10（データ領域バーを画面上に表示する）と F12 を押すか，または［Info］キーを押して，フラグ ShowResources = TRUE を正しく設定（例を参照）すれば，残っている Windows システムリソースの一部がパーセントで表示されます。

例 6-25 ShowResources エントリ

```
[Miscellaneous]
ShowResources = True
; Important for development !
; displays remaining resources.
```

■ 6.3.1.7 ［CommandToTask］セクション

このセクションには，MMC でアクションを開始するために NCK で使われるコマンドインタープリタチャネルのデフォルト値が入っています。

例 6-26 Cycles エントリ

```
[CommandToTask]
Cycles = comic
; Among others the measuring cycles are controlled by the
command
; interpreter channel.
```

6.3.2 初期化ファイルの言語 .INI

概説

初期化ファイルの言語 .INI には，Regie のソフトキーテキストと一般テキストが入っています。表 6.8 は，ここに入っているセクションの概要を示しています。

表 6.8 言語 .INI ファイルのセクション

セクション	意味
HSoftkeyTexts	水平ソフトキーバーのテキスト
VSoftkeyTexts	垂直ソフトキーバーのテキスト
GeneralTexts	メッセージなどのその他のテキスト
DISP	
Keyboard_State	

言語 .INI ファイル（たとえばドイツ語バージョンは RE_GR.INI）は，任意の ASCII エディタを使って編集することができます。

例 6-27 ファイル language.INI の抜粋

```
[HSoftkeyTexts]
; 1. softkey bar
HSK0 = "machine" // 20
```

```
HSK1 = "parameters" // 20
...
[GeneralTexts]
GT0 = "SINUMERIK system start-up. Please wait ..." // 50
...
WIth the following meaning
```

HSK0	水平ソフトキー 0
GT0	一般テキスト番号 0
"machine"	ソフトキーテキスト。2 つのスペースを使って 2 行のテキストを作成します。
//20	テキストごとの最大許容文字数（任意指定）を指定する注釈

■ 6.3.2.1 ［HSoftkeyTexts］セクション

このセクションには，水平ソフトキーのテキストが入ります。REGIE.INI の［TaskConfiguration］セクション内で，タスク（Task0 ～ Task23）として構成されるどのアプリケーションにも，それに対応するソフトキーラベルが言語 .INI 内になければなりません。たとえば，タスク 6 を REGIE.INI 内に構成した場合に，言語 .INI 内の HSK6 にエントリがないと，ソフトキーを使ってアプリケーションを開始することはできません。これにより，言語を特定してアプリケーションをロックすることができます。

例 6-28 ソフトキーラベルの割り当て

```
Excerpt from REGIE.INI:
[TaskConfiguration]
Task1=name := param, Timeout := 60000, PreLoad := False
Excerpt from RE_GR.INI:
[HSoftkeyTexts]
; 1. softkey bar
HSK1 = "Parameter" // 20.
```

■ 6.3.2.2 ［VSoftkeyTexts］セクション

このセクションには，Regie の垂直ソフトキーのテキストが入ります。ソフトキーのラベル付けを再構成することはできません。

■ 6.3.2.3 ［GeneralTexts］セクション

このセクションには，Regie で示されるテキストが入ります。

6.4 Regie のダイナミックリンクライブラリ

機能の概要

ダイナミックリンクライブラリ REGIE.DLL には，以下を行うための関数が用意されています。

- タスクの切り替え
- 非表示領域へのタスクの切り替え
- 非表示領域へのタスクの即時切り替え
- 画面制御
- ロック設定
- コマンド行
- その他

表 6.9 は，REGIE.DLL の関数の概要を示しています。

表 6.9 REGIE.DLL の関数

名前	摘要
タスクの切り替え	
SwitchToHelpTask	ヘルプタスクへ切り替えます
SwitchToPreviousTask	直前のタスクへ切り替えます
SwitchToTask	タスク索引を使ってタスクを切り替えます
非表示領域でのタスクの切り替え	
SwitchToChild	タスク番号を使って非表示領域でタスクを活動化します
SwitchToChildEx	タスク名を使って非表示領域でタスクを活動化します
SwitchToParent	活動化プログラムへ戻ります
SwitchToParentAndKillMe	活動化プログラムに戻り，メモリからタスクを除去します
非表示領域へのタスクの即時切り替え	
SwitchToChildImmediate	タスク番号を使った非表示領域でのタスクの活動化（即時）
SwitchToChildImmediateEx	タスク名を使って非表示領域でタスクを活動化します（即時）
SwitchToParentImmediate	活動化プログラムへ戻ります（即時）
(SwitchToParentAndKill-MeImmediate	活動化プログラムへ戻り，メモリからのタスクを除去します（即時）
画面制御	
ScreenOn	画面をオンにします
ScreenOff	画面をオフにします
MMCScreenOn	画面＋インタフェースビットをオンにします
MMCScreenOff	画面＋インタフェースビットをオフにします

ContrastDown	平面画面モニタのコントラストを低減します
ContrastUp	平面画面モニタのコントラストを増加させます

名前	摘要
ロック設定	
LockCurrentNCU	現在のタスク用の NCU をロックします
LockNCU	指定のタスク用の NCU をロックします
UnlockCurrentNCU	現在のタスク用の NCU のロックを解除します
UnlockNCU	指定のタスク用の NCU のロックを解除します
IsCurrentNCULocked	現在の NCU のロック設定ユーザを判別します
コマンド行	
ReadCmdLine	領域コマンド行を読み取ります（C または C++ の呼び出し）
ReadCmdLineVB	領域コマンド行を読み取ります（C での Visual Basic の呼び出し）
ReadCmdLineMe	現行領域のコマンド行を読み取ります（C または C++ の呼び出し）
ReadCmdLineMeVB	現行領域のコマンド行を読み取ります（C での Visual Basic の呼び出し）
WriteCmdLine	「子」用のコマンド行を作成します（C または C++ の呼び出し）
WriteCmdLineVB	「子」用のコマンド行を作成します（タスク索引を使った C の Visaul Basic の呼び出し）
WriteCmdLineVBEx	「子」用のコマンド行を作成します（タスク名を使った C の Visaul Basic の呼び出し）
その他	
GetMMCDir	MMC ディレクトリを判別します
GetMMCLanguagePath	現行タスクの DLL にアクセスします
InitComplete	初期化の正常完了を通知します
StopRegieEvents	新規イベントの起動を阻止します
TestAndStopRegieEvents	StopRegieEvents を拡張します
ResumeRegieEvents	StopRegieEvents を再開します
SetModeChannelSwitchKey	チャネル切り替えキーをロックまたは活動化します

使用法

REGIE.DLL に入っている関数は, Visual Basic からだけでなく , C および C++ からも呼び出すことができます。Visual Basic からの呼び出しの場合，特別なヒントが示され

ます。

実行時のリンク

REGIE.DLL に入っている関数（および他のすべての MMC システム DLL に入っている関数）は必ず，静的に（つまり開発ツール IMPLIB および LINK を使って）ではなく，実行時に（LoadLibrary と GetProcAddr を使って）リンクさせなければなりません。

静的なリンクには，次のような大きな欠点があります。:

- システム DLL を再コンパイルすると，使われているすべてのユーザアプリケーションも再コンパイルする必要が生じます（静的にリンクされた関数表のポインタの順序は変更されている可能性があります)。
- 実行時に静的にリンクされた DLL は，動的にリンクされた DLL よりも遅くなります（関数表のポインタに起因する追加の間接ステップのため)。

DLL のロード

LoadLibrary を使って DLL（たとえば REGIE.DLL）を初めてロードすると，REGIE.DLL ファイルは，現在のディレクトリ内または Windows 検索パス内に必ず置かれます。

注： 場合によっては開発中（たとえばデバッガでのテスト中）に，**LoadLibrary** が REGIE.DLL ファイルを見つけられないことがあります。

対処策 : WPS（Visual Basic 開発環境にある Windows Process Status）を使って，手動で DLL を始動します。そうすれば，以後の DLL 呼び出しは正常に完了します。DDL の参照カウンタだけが増やされるからです（ロードプロセスを含みません)。

6.4.1 タスクの切り替え

概説

キーボードアクションで領域を選択する（明示的選択）だけでなく，Regie の DLL インタフェースのうちのいずれかを使って，領域切り替えを行う（暗黙的選択）こともできます。表 6.10 に，領域切り替え用の REGIE.DLL の関数の概要を示しています。

表 6.10 REGIE.DLL の関数（タスク切り替え）

名前	摘要
タスクの切り替え	
SwitchToTask	タスク索引でタスクへ切り替えます
SwitchToPreviousTask	直前のタスクへ切り替えます
SwitchToHelpTask	ヘルプへ切り替えます

■ SwitchToTask

説明

キー操作の代わりに，DLL 機能を使って暗黙の領域選択を実行します。

宣言

```
Declare Sub SwitchToTask Lib "Regie"
                    └> (ByVal nTaskIndex As Integer)
```

構文

Sub SwitchToTask (ByVal nTaskIndex As Integer)

引き数

表 6.11 SwitchToTask の引き数

引き数	データタイプ	意味
NTaskIndex	整数	*REGIE.INI* ファイルの［*TaskConfiguration*］セクション内に指定されているタスクの索引

C の，SwitchToTask を使った暗黙の領域切り替え

例 6-29 C の，SwitchToTask を使った暗黙の領域切り替え

REGIE.INI ファイルの[TaskConfiguration]セクション内のタスク 5 としての診断 DG のエントリは，以下のとおりです。

```
[TaskConfiguration]
Task5 = Name := DG
```

タスク 5（診断）への暗黙の領域切り替えは次のようにします。

```
void (WINAPI * lpfnSwitchToTask) (int);
...
(* lpfnSwitchToTask) (5);
```

■ SwitchToPreviousTask

説明

現在の領域の直前に活動状態にあった領域が選択されます。ある意味では，この関数は，暗黙の領域選択とは反対の機能と言えます。

SwitchToPreviousTask. 関数を使って，2 つの領域アプリケーションのどちらか一方からもう一方へ切り替えることができます。

注 : **SwitchToPreviousTask** は，「領域切り替え」（**F10**）を **2** 回押すのと同じ働きをします。

宣言

```
Declare Sub SwitchToPreviousTask Lib "Regie" ()
```

構文

Sub SwitchToPreviousTask ()

パラメータ

なし

例 6-30 C の，直前に活動状態であった領域への暗黙の切り替え

```
void (WINAPI * lpfnSwitchToPreviousTask) (void)
...
(* lpfnSwitchToPreviousTask) ();
```

■ SwitchToHelpTask

説明

SwitchToHelpTask 関数は，領域 help，つまりヘルプ索引で指定されたタスクに切り替えます。

詳細については，6.8 のヘルプサポートに関する項で記載しています。

宣言

Declare Sub SwitchToHelpTask Lib "Regie" ()

構文

Sub SwitchToHelpTask ()

パラメータ

なし

C の，SwitchToHelpTask によるヘルプタスクへの切り替え

例 6-31 C の，SwitchToHelpTask によるヘルプタスクへの切り替え

```
void (WINAPI * lpfnSwitchToHelpTask) (void)
...
(* lpfnSwitchToHelpTask) ();
```

6.4.2 非表示領域でのタスクの切り替え

構造化領域の切り替え

REGIE.DLL には，親子関係にある領域を編成するための関数ファミリが用意されています。たとえば，大きい領域を複数の小さな独立サブ領域に分割することができます。これらは，関数ファミリでまとめて管理され，「非表示領域内でのタスクの切り替え」が行われます。

表 6.12 は，非表示領域でのタスクの切り替えの場合の REGIE.DLL 関数の概要を示しています。

表 6.12 REGIE.DLL の関数（タスクの切り替え）

名前	摘要
非表示領域でのタスクの切り替え	

SwitchToChild	タスク番号を使って非表示領域でタスクを活動化します
SwitchToChildEx	タスク名を使って非表示領域でタスクを活動化します
SwitchToParent	活動化プログラムへ戻ります
SwitchToParentAndKillMe	活動化プログラムに戻り，メモリからタスクを除去します

非表示領域での REGIE.INI 内のエントリ

操作領域「diagnosis」には，さまざまなサイズのデータ領域の編集用のエディタが入っています。DIAGNOSE.EXE ファイルのサイズを制限するため，エディタの機能は 2 番目の「非表示」領域に移されています。

例 6-32 非表示領域の REGIE.INI のエントリ

```
[TaskConfiguration]
Task5 = Name := DIAGNOSE, ...
Task24 = Name := AEDITOR, ... ; ASCII editor is located in
the
; hidden area
```

「非表示」領域

このコンテキストでの「非表示」の意味は次のとおりです。つまり，構成しようとする 32 個の領域のうち，上位の 24 から 31 までの領域には，データ領域バーから（ETC キーを使って）アクセスできません。アクセスは，アプリケーション（上記の例では diagnosis）で，関数呼び出し SwitchToChild または **SwitchToChildEx** だけで実行できます。この場合，**SwitchToChild** はタスク番号を使い，**SwitchToChildEx** はタスク名を使います。オペレータがエディタを使って，たとえばマシン領域から，データ領域バーを使って診断領域に（ソフトキー **F6** を使って）切り替えてから，また元の領域に切り替えると，非表示領域 **AEDITOR** は自動的に活動化されます。

SwitchToParent 関数を呼び出せば，「子」を切り替えて「親」に戻ることもできますが，これをキーボードアクションで行うことはできません。「子」を「親」に切り替える場合，同時に子をメモリから削除することによって，Windows リソースなどを節約することができます。その場合に切り替えて戻るには，子で SwitchToParentAndKillMe を使います。図 6.2 は，非表示領域の関数を示しています。

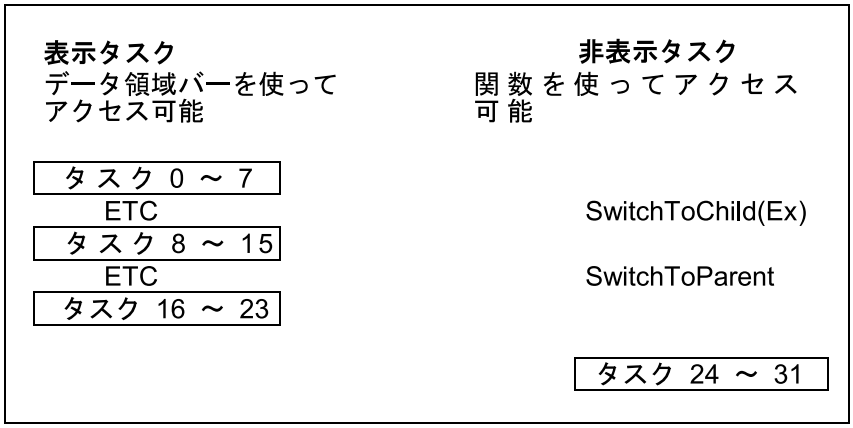


図 6.2 非表示領域の使用

注 **: Regie** のこの新規関数を使用する場合，**State_Changed** の戻り値を **0** に設定しないように注意してください。**0** に設定すると，シーケンス制御で，以後の領域切り替えやディスプレイの構築などのアクションが阻害されることがあります。

注 **:**
以下は 2 つの関数ファミリのうちの 1 つです。
SwitchToTask
SwitchToPreviousTask
SwitchToHelpTask

以下は 2 つの関数ファミリのうちのもう 1 つです。
SwitchToChild
SwitchToChildEx
SwitchToParent
SwitchToParentAndKillMe

この 2 つのファミリは，意図的に区別されています。つまり，**SwitchToChild** がう回した領域が **SwitchToTask** によって活動化された場合，う回された領域ではなく，**SwitchToTask** の引き数に指定された領域が切り替えられます。

SwitchToChild

説明

［TaskConfiguration］セクション内の「子」として構成された領域を選択します。そのタスク索引は 24 ～ 63 の範囲になります。番号 51 ～ 63 は，お客様の OEM 子アプリケーションに使うことができます。

したがって，選択した「子」領域が，呼び出し側の「親」に重ねて置かれるかまたは置換されます。データ領域バーを使ってこれをキーボードで行うと，「親」の代わりに「子」が表示されます。

この重なりを取り消すには，SwitchToParent または SwitchToParentAndKillMe を使います。

ヒント **:** この「親／子」領域の編成をネスト化することができます。つまり，SwitchToChild を使えば，「子」は「孫」をもつことができます。

宣言

```
Declare Sub SwitchToChild Lib "Regie"
_ (ByVal nTaskIndex As Integer)
```

構文

Sub SwitchToChild (ByVal nTaskIndex As Integer)

引き数

表 6.13 SwitchToChild の引き数

引き数	データ型	意味
nTaskIndex	整数	REGIE.INI 内の［TaskConfiguration］セクションに指定されるタスクの索引。ここで n は 24 以上。

例 6-33 C の，SwitchToChild を使った子への領域切り替え

次のように，ASCII エディタ AEDITOR をタスク 24 として，REGIE.INI 内の［TaskConfiguration］セクションに入力します。

```
[TaskConfiguration]
Task24 = Name := AEDITOR
```

次のようにして，領域をタスク 24（ASCII エディタ）に切り替えます。

```
void (WINAPI * lpfnSwitchToChild) (int);
...
(* lpfnSwitchToChild) (24);
```

■ SwitchToChildEx

説明

アプリケーションの初期化ファイル内の［Childs］セクション内に構成されている領域を選択します。このセクションで，REGIE.INI 内に構成されている［TaskConfiguration］セクションの領域のタスク索引に名前を割り当てることができます。［Childs］セクションに入力された名前で呼び出すことが，SwitchToChild 関数との唯一の相違です。

宣言

```
Declare Sub SwitchToChildEx Lib "Regie"
_ (ByVal szTaskName As String)
```

構文

Sub SwitchToChildEx (ByVal szTaskName As String)

引き数

表 6.14 SwitchToChildEx の引き数

引き数	データ型	意味
szTaskName	ストリング	ご使用のアプリケーションの INI ファイルの［CHILDS］セクションに指定しているタスク名

例 6-34 C の，SwitchToChildEx を使った「子」への領域切り替え

次のように，ASCII エディタ AEDITOR をタスク 5，24，および 25 として，REGIE.INI ファイルの［TaskConfiguration］セクションに入力します。

```
[TaskConfiguration]
Task05 = name := AEDITOR
Task24 = name := AEDITOR
Task25 = name := AEDITOR
```

アプリケーション OEMBSP1 を，タスク 24（ASCII エディタ）に変更できなければなりません。そのため，初期化ファイル OEMBSP1.INI に次のエントリが必要になります。

```
[CHILDS]
MYEDIT = 24
```

次のようにして，タスク 24（ASCII エディタ）に領域を切り替えます。

```
void (WINAPI * lpfnSwitchToChildEx) (string);
...
(* lpfnSwitchToChildEx) ("MYEDIT");
```

■ SwitchToParent

説明

選択された「子」領域から呼び出し側の「親」に戻って，重なり状態を解消します。

注 **: SwitchToParent** を続けて **2** 回呼び出すと，「孫」から切り替えて「親」に戻ります。

宣言

```
Declare Sub SwitchToParent Lib "Regie" ()
```

構文

Sub SwitchToParent ()

引き数

なし

例 6-35 C の，SwitchToParent を使った「親」への領域切り替え

次のようにして「親」に戻ります。

```
void (WINAPI * lpfnSwitchToParent) (void);
...
(* lpfnSwitchToParent) ();.
```

■ SwitchToParentAndKillMe

説明

SwitchToParent と同様に，選択された「子」領域から呼び出し側の「親」に戻って，重なり状態を解消します。次に，WM_CLOSE を使って，PostMessage とともに「子」領域をクローズします。

注 : この関数の使用をお勧めするのは，**Windows** のシステムリソースが不足した場合や，「子」領域をまれにしか呼び出さない場合です。**SwitchToChild** をもう一度呼び出すと，**Regie** によりこれは再ロードされます。

宣言

```
Declare Sub SwitchToParentAndKillMe Lib "Regie" ()
```

構文

Sub SwitchToParentAndKillMe ()

引き数

なし

例 6-36 C の，「親」領域からの戻りと「子」領域のクローズ

```
void (WINAPI * lpfnSwitchToParentAndKillMe) (void);
...
(* lpfnSwitchToParentAndKillMe) ();
```

6.4.3 非表示領域へのタスクの即時切り替え

表 6.15 は，タスクの即時切り替えの場合の REGIE.DLL 関数の概要を示しています。

表 6.15 REGIE.DLL の関数（タスクの即時切り替え）

名前	摘要
非表示領域へタスクを即時切り替えます	
SwitchToChildImmediate	タスク番号を使って非表示領域でタスクを（即時）活動化します
SwitchToChildImmediateEx	タスク名を使って非表示領域でタスクを（即時）活動化します
SwitchToParentImmediate	活動化プログラムへ（即時）戻ります
SwitchToParentAndKillMeImmediate	活動化プログラムへ（即時）戻り，メモリからタスクを除去します

構文

これらの関数の構文は，前章に説明されている構文に対応します。

Regie キュー

その他の情報 :

Regie は，押されたキー（直前のアプリケーションに戻る F10 ＋ F10 など）や，SwitchToParent などの関数に応答します。

このような Regie アクションのトリガは，Regie キューに入って内部で管理されます。キーボードアクションと関数を同時に実行すると，コマンド順序の中断という望ましくない結果につながります。

例 :

あるアプリケーションで，垂直ソフトキー F8（略して +F8）を使って，子アプリケーション（エディタなど）をクローズするとします。アプリケーションは，このアクションの終了時点で，SwitchToParent によって直前のアプリケーションに戻るはずです。同時に，F10，F10，F4 とキーを順にクリックして，直前のアプリケーションのソフトキー 4 に割り当てられた関数を起動しようとしたとします。

この場合，SwitchToParent とキーシーケンス F10，F10，F4 は対立することになります。その結果，図 6.3 に示されているとおり，アクションの順序が混乱します。SwitchToParent 関数の Regie キューへの追加は遅くなってしまいます。

キー	指示したいアクション	関数	Regie キュー
+ F8	アプリケーションのクローズ	関数と +F8 の実行	+F8
F10	ステップ 1: データ領域バーの活動化	関数と +F8 の実行	F10
F10	ステップ 2: 親の活動化	関数と +F8 の実行	F10
F4	ソフトキー 4 を使ったアクションの実行	関数と +F8 の実行	F4
		SwtichToParent を呼び出して関数が終了	**SwitchTo Parent**

図 6.3 キーが早く押された場合の SwitchToParent の動作。F10，F10，および F4 が押された後，SwitchToParent イベントが呼び出されますが，通常はすでに実行されています。

解決方法 **:**

ソフトウェアリリース 3.2 以前の DLL 関数の関数ファミリは次のとおりです。

SwitchToChild
SwitchToChildEx
SwitchToParent
SwitchToParent AndKillMe

ソフトウェアリリース 3.2 以後，以上のファミリに，以下のような第 2 バージョンの即時 DLL 関数が追加されました。これらは基本的に同じ意味を持っています。

SwitchToChildImmediate
SwitchToChildImmediateEx
SwitchToParentImmediate
SwitchToParent AndKillMeImmediate.

関数呼び出しがあると，これらは対応するイベントを Regie キューの先頭に挿入します。これにより，次回そのキューへのアクセスが行われたとき，そのイベントが次に取り出されます。

図 6.4 は，SwitchToParentImmediate の関数を示しています。

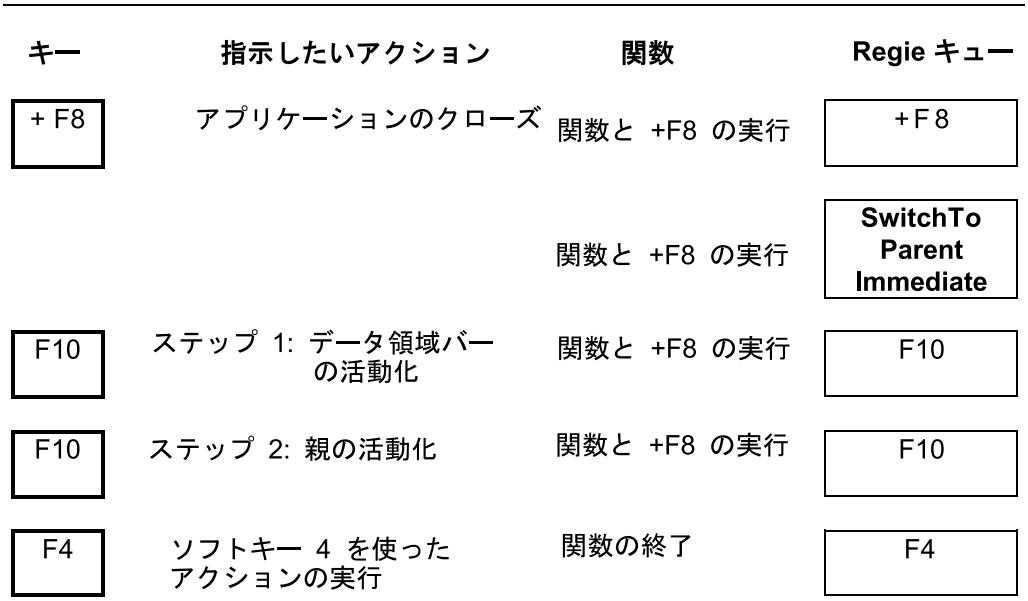


図 6.4 キーが早く押された場合の SwitchToParentImmediate の動作。
SwitchToParent イベントは，+F8 コマンドの直後に挿入されます。

結論 **: OEM** アプリケーションの開発では，キーを押すのが早すぎる場合に上記のような順序問題が起きるかどうかを調べて確認してください。

これは，Regie による，キーをトリガするすべての領域切り替えに対しても当てはまります。それには，状態切り替えやアプリケーション内でのアクションも含まれます。

順序問題が起きる可能性があると思われる場合，それに対応する即時バージョンを使用してください。

6.4.4 画面制御

表 6.16 は，画面制御に関する REGIE.DLL の関数の概要を示しています。

表 6.16 REGIE.DLL の関数（画面制御）

名前	摘要
画面制御	
ScreenOn	画面をオンにします
ScreenOff	画面をオフにします
MMCScreenOn	画面＋インタフェースビットをオンにします
MMCScreenOff	画面＋インタフェースビットをオフにします
ContrastDown	平面画面モニタのコントラストを下げます

ContrastUp	平面画面モニタのコントラストを上げます

詳細については，関数 /A2/ の解説を参照してください。

■ (MMC)ScreenOn と (MMC)ScreenOff

説明

画面のオン／オフを行います。

REGIE.DLL には，次のように，モノクロ LCD 画面を備えた MMC 操作パネルの画面のオン／オフを行うための 4 つの関数が用意されています。:

ScreenOn
ScreenOff
MMCScreenOn
MMCScreenOff

ScreenOff は，画面とバックライトの両方をオフにします。

MMCScreenOff は，画面とバックライトの両方をオフにし，さらにインタフェース信号 **DB19**，**DBB0**，ビット **1**（ビット **1=1**）を設定します。

ScreenOn は，画面とバックライトの両方をオンにします。

MMCScreenOn は，画面とバックライトの両方をオンにし，さらにインタフェース信号 **DB19**，**DBB0**，ビット **1**（ビット **1=0**）を設定します。

宣言

```
Declare Sub ScreenOn Lib "Regie" ()
Declare Sub ScreenOff Lib "Regie" ()
Declare Function MMCScreenOn Lib "Regie"()
_ As Integer
Declare Function MMCScreenOff Lib "Regie" ()
_ As Integer
```

構文

Sub ScreenOn ()
Sub ScreenOff ()
Function MMCScreenOn () As Integer
Function MMCScreenOff () As Integer

C での呼び出し

```
void WINAPI ScreenOn(void)
void WINAPI ScreenOff(void)
BOOL WINAPI MMCScreenOn(void)
BOOL WINAPI MMCScreenOff(void)
```

引き数

なし

戻り値 :

MMCScreenOn と MMCScreenOff の場合のみ :

0　　エラー

1　　エラーなし

■ ContrastUp と ContrastDown

説明

LCD 画面のコントラストを設定します。

REGIE.DLL には, モノクロ LCD 画面を備えた MMC 操作パネルの画面のコントラストを設定するために，ContrastUp および ContrastDown という 2 つの関数が用意されています。

コントラストの設定値は 16 のレベルに分かれています。現在のレベルは 4 ビットカウンタに保管されます。Systemreset を実行すると，カウンタは 0111b にプリセットされます。これは，デフォルトのコントラストレベル 8 です。

カウンタを 1 ステップずつ増減するには，ContrastUp 関数と ContrastDown 関数を使います。

MMC 103 基本システムでは，どちらの関数も，REGIE の未使用の 2 つの垂直ソフトキー 7 および 8 に置かれます。

注 **:** 領域アプリケーション（サービス，診断など）では当然，さらに快適なコントラストの設定が実現できます。**.** たとえば，カウンタを **INI** ファイルに保管しておくことができます。そうすれば，システムの再始動後に，該当する回数だけ **ContrastUp** または **ContrastDown** の呼び出しを実行することで，カウンタ値を復元することができます。

アプリケーション

ContrastUp はモノクロ LCD 画面のコントラストを増大します。**ContrastDown** はモノクロ LCD 画面のコントラストを低減します。これは，ContrastUp に似た働きをしますが，逆の効果を生じます。

宣言

```
Declare Function ContrastUp Lib "Regie" ()
_ As Integer
Declare Function ContrastDown Lib "Regie" ()
_ As Integer
```

構文

Function ContrastUp () As Integer
Function ContrastDown () As Integer

C での呼び出し

```
BOOL WINAPI ContrastUp(void)
BOOL WINAPI ContrastDown(void)
```

引き数

なし

戻り値 :

0　　仮想デバイスドライバVMMC2D.386がロードされていない可能性があります。

1　　エラーなし

6.4.5 タスクのロック

概説 :

ソフトウェアリリース 2.3 以後，次のように，ネット内の任意の NCU または PLC モジュールに MMC オペレータインタフェースを接続できるようになりました。

接続は，操作領域「connection」または NCDDE 変数「MachineSwitch」で行います。

REGIE.DLL のロック関数を呼び出せば，切り替えに選択的に影響を与えることができます。つまり，特殊 NCU のロックまたはロック解除を行います。

ヒント : この関数はシステムの柔軟性の妨げになるので，慎重に使用してください。

表 6.17 は，REGIE.DLL のロック関数の概要を示しています。詳細については続く記述で説明しています。

表 6.17 REGIE.DLL のロック関数

名前	摘要
ロック設定	
LockCurrentNCU	現在のタスク用の NCU をロックします
LockNCU	指定のタスク用の NCU をロックします
UnlockCurrentNCU	現在のタスク用の NCU のロックを解除します
UnlockNCU	指定のタスク用の NCU のロックを解除します
IsCurrentNCULocked	現在の NCU のロック設定ユーザを判別します

■ LockCurrentNCU，UnlockCurrentNCU

説明

それぞれ，指定されたタスクの NCU のロック（LockCurrentNCU）と指定されたタスクの NCU のロック解除（UnlockCurrentNCU）を行います。

宣言

```
Declare Function LockCurrentNCU Lib "Regie" ()
_ As Integer
Declare Function UnlockCurrentNCU Lib "Regie" ()
_ As Integer
```

構文

```
Function LockCurrenNCU () As Integer
Function UnlockCurrentNCU () As Integer
```

引き数

なし

戻り値 :

0　　エラー

1　　関数の実行完了

Visual Basic での呼び出し :

以下は，LockCurrentNCU 関数と UnlockCurrentNCUin 関数の Visual Basic の例です。

例 6-37 Visual Basic での呼び出し

```
Dim i As Integer
i = LockCurrentNCU()
.....
i = UnlockCurrentNCU()
```

注 : これらの関数は，Regie においては役立ちます。Visual Basic ワークベンチのもとでは，REGIE.DLL がロードされていても，これらの関数には効果はありません。

C での呼び出し

```
BOOL _export WINAPI LockCurrentNCU (VOID)
BOOL _export WINAPI UnlockCurrentNCU (VOID)
```

例

例 6-38 C の，現在のタスクの NCU のロック解除

```
BOOL _export WINAPI UnlockCurrentNCU (VOID);
```

■ LockNCU，UnlockNCU

説明

特定のタスクの NCU をロックし（LockNCU），特定のタスクの NCU をロック解除します（UnlockNCU）。

用途

この関数が役に立つのは，「親／子」の関係をさらに分割する場合です。この関数を使って，たとえは「親」をロックして「子」をロックを解除することができます。

注 : ロックとロック解除は，同じ索引を使って行わなければなりません。

宣言

```
Declare Function LockNCU Lib "Regie"
_ (ByVal nTaskIndex As Integer) As Integer
Declare Function UnlockNCU Lib "Regie"
_ (ByVal nTaskIndex As Integer) As Integer
```

構文

Sub LockNCU (ByVal nTaskIndex As Integer)
Sub UnlockNCU (ByVal nTaskIndex As Integer)

引き数

表 6.18 LockNCU と UnlockNCU の引き数

引き数	意味
nTaskIndex	REGIE.INI の［TaskConfiguration］セクションに指定されるタスクの索引。

Visual Basic での呼び出し

Visual Basic で関数 LockNCU および UnlockNCU を呼び出すには, 次のようにします。

例 6-39 Visual Basic での呼び出し

REGIE.INI の[TaskConfiguration]セクション内のタスク 5 としての診断 DG を次のように入力します。

```
[TaskConfiguration]
Task5 = Name := DG
```

次のように呼び出します。

```
LockNCU 5
...
UnlockNCU 5
```

注 : これらの関数は，Regie においては役立ちます。Visual Basic ワークベンチのもとでは，REGIE.DLL がロードされていても，これらの関数には効果はありません。

C での呼び出し

VOID _export WINAPI LockNCU (int nTaskIndex)
VOID _export WINAPI UnlockNCU (int nTaskIndex)

例 6-40 C の，タスク 5 の NCU のロック

次のように呼び出してタスク 5（diagnosis）の NCU をロックします。

```
void (WINAPI * lpfnLockNCU) (int);
...
(* lpfnLockNCU) (5);
```

■ IsCurrentNCULocked

説明

NCU が現在ロックされているかどうかを報告します。

宣言

```
Declare Function IsCurrentNCULocked Lib "Regie" ()
_ As Integer
```

構文

```
Function IsCurrentNCULocked () As Integer
```

引き数 :

なし

戻り値 :

-1　　現在の NCU はロックされていません。

0 ～ 31　　現在の NCU はロックされています。戻り値は，ロックを設定されたタスクの索引を示します。

Visual Basic での宣言と呼び出し

Visual Basic で関数 IsCurrentNCULocked を次のように宣言します。

例 6-41 Visual Basic での宣言と呼び出し

REGIE.INI の[TaskConfiguration]セクション内のタスク 5 としての診断 DG を次のように入力します。

```
[TaskConfiguration]
Task5 = Name := DG
```

次のように，タスク 5（diagnosis）の NCU をロックします。

```
LockNCU 5
```

次のように呼び出します。

```
Dim i As Integer
i = IsCurrentNCULocked()
```

IsCurrentNCULocked を使ってこのロックを検査すると，[TaskConfiguration] セクションに指定されているタスク「diagnosis」のタスク索引である i = 5 が戻されます。

C での呼び出し

```
int _export WINAPI IsCurrentNCULocked (VOID)
```

例 6-42 C の，現在の NCU のロックの検査

REGIE.INI の[TaskConfiguration]セクション内のタスク 5 としての診断 DG を次のように入力します。

```
[TaskConfiguration]
Task5 = Name := DG
```

次のように，タスク 5（diagnosis）の NCU をロックします。

```
int (WINAPI * lpfnIsCurrentNCULocked) ();
int i;
...
i = (*lpfnIsCurrentNCULocked)();
```

IsCurrentNCULocked を使ってこのロックを検査すると，[TaskConfiguration] セクションに指定されているタスク「diagnosis」のタスク索引である 5 が戻されます。

注 **:** これらの関数は，**Regie** においては役立ちます。**Visual Basic** ワークベンチのもとでは，**REGIE.DLL** がロードされていても，これらの関数には効果はありません。

6.4.6 コマンド行を管理するための関数

概説

コマンド行のインストールは，次のようにして各領域アプリケーションごとに実行することができます。

- REGIE.INI ファイルの［TaskConfigura-tion］セクション内の CmdLine 属性を使って静的に
- REGIE.DLL のコマンド WriteCmdLine，WriteCmdLineVB，または WriteCmdLineVBEx を使って動的に

コマンド行の情報を読み取るには，適切なバージョンのコマンド ReadCmdLine を使います。

表 6.19 は，REGIE.DLL のコマンド行関数の概要を示しています。

表 6.19 REGIE.DLL のコマンド行関数

名前	摘要
コマンド行	
ReadCmdLine	領域コマンド行を読み取ります（C または C++ での呼び出し）
ReadCmdLineVB	領域コマンド行を読み取ります（Visual Basic での呼び出し）
ReadCmdLineMe	現行領域のコマンド行を読み取ります（C または C++ の呼び出し）
ReadCmdLineMeVB	現行領域のコマンド行を読み取ります（Visual Basic の呼び出し）
WriteCmdLine	「子」用のコマンド行をインストールします（C または C++ の呼び出し）
WriteCmdLineVB	「子」用のコマンド行をインストールします（タスク索引による Visual Basic の呼び出し）
WriteCmdLineVBEx	「子」用のコマンド行をインストールします（タスク名による Visual Basic の呼び出し）

■ ReadCmdLine

説明

C プログラムで，ReadCmdLine を使って領域のコマンド行を読み取ります。領域は，そのタスク番号で指定しなければなりません。

構文

```
int WINAPI
ReadCmdLine (int nTaskIndex, LPSTR szDest, intnDestLen)
```

引き数

表 6.20 ReadCmdLine の引き数

引き数	データ型	意味
nTaskIndex	整数	REGIE.INI ファイルの［TaskConfiguration］セクションに指定されるタスクの索引。
szDest	ポインタ	コマンド行が書き込まれる先のバッファ
nDestLen	整数	バッファの長さ

戻り値 :

0　　エラー。タスクまたはコマンド行を使えません。

n　　実際に読み取られた文字数

領域のコマンド行の読み取り

例 6-43 領域のコマンド行の読み取り

```
int len;
char szBuf [128];
int (WINAPI * lpfnReadCmdLine) (int, LPSTR, int);
...
len = (* lpfnReadCmdLine) (16, (LPSTR) szBuf, sizeof
(szBuf));
```

■ ReadCmdLineVB

説明

VB プログラムで，ReadCmdLineVB を使って領域のコマンド行を読み取ります。領域は，そのタスク番号で指定しなければなりません。

宣言

```
Declare Function ReadCmdLineVB Lib "Regie"_ (ByVal nTaskIn-
dex As Integer) As String
```

構文

Function ReadCmdLineVB (ByVal nTaskIndex As Integer) AsString

引き数

表 6.21 ReadCmdLineVB の引き数

引き数	データ型	意味

nTaskIndex	整数	REGIE.INI ファイルの［TaskConfiguration］セクションに指定されるタスクの索引。

戻り値

読み取られたコマンド行

Visual Basic での呼び出し

Visual Basic で ReadCmdLineVB 関数を呼び出すには，次のようにします。

例 6-44 Visual Basic での呼び出し

```
Dim nTaskIndex As Integer
Dim szString As String
...
szString = ReadCmdLineVB(nTaskIndex)
```

■ ReadCmdLineMe

説明

C プログラムで ReadCmdLineMe を使って, 現在の領域のコマンド行を読み取ります。

C での呼び出し

```
int WINAPI
ReadCmdLineMe (LPSTR szDest, int nDestLen)
```

引き数

表 6.22 ReadCmdLineMe の引き数

引き数	データ型	意味
szDest		コマンド行が書き込まれる先のバッファ
nDestLen		バッファの長さ

戻り値 :

0　　エラー。タスクまたはコマンド行を使えません。

n　　実際に読み取られた文字数

例 6-45 現在の領域のコマンド行の読み取り

```
int len;
char szBuf [128];
int (WINAPI * lpfnReadCmdLineMe) (LPSTR, int);
...
len = (* lpfnReadCmdLineMe) ((LPSTR) szBuf, sizeof (szBuf));
```

■ ReadCmdLineMeVB

説明

VB プログラムで ReadCmdLineMeVB を使って，現在の領域のコマンド行を読み取ります。

VB での宣言

```
Declare Function ReadCmdLineMeVB Lib "Regie"_ ()
                                    As String
```

引き数

なし

戻り値

読み取られたコマンド行

Visual Basic での呼び出し

Visual Basic で ReadCmdLineMeVB 関数を呼び出すには，次のようにします。

例 6-46 Visual Basic での呼び出し

```
Dim szString As String
...
szString = ReadCmdLineMeVB()
```

■ WriteCmdLine

説明

C プログラムで，WriteCmdLine を使って「子」領域のコマンド行をインストールします。領域は，そのタスク番号で指定しなければなりません。

構文

```
int WINAPI
WriteCmdLine (int nTaskIndex, LPSTR szBuf)
```

引き数

表 6.23 WriteCmdLine の引き数

引き数	データ型	意味
nTaskIndex	整数	REGIE.INI ファイルの［TaskConfiguration］セクションに指定されるタスクの索引。
szBuf	ポインタ	任意の長さのコマンド行（0 終了の C ストリング）のバッファ

戻り値 :

0　　エラー。タスクが欠落しているか，またはコマンド行はすでにインス

トールされています。

n　　終了の 0 文字を除いて，バッファに書き込まれた文字数

例 6-47 領域のコマンド行のインストール

```
int len;int (WINAPI * lpfnWriteCmdLine) (int, LPSTR);...len = (*
lpfnWriteCmdLine) (24, "a new command line");
```

■ WriteCmdLineVB

説明

VB プログラムで，WriteCmdLineVB を使って「子」領域のコマンド行をインストールします。領域は，そのタスク番号で指定しなければなりません。

宣言

```
Declare Function WriteCmdLineVB Lib "Regie"
_ (ByVal nTaskIndex As Integer,
_ CmdLine As String) AS Integer
```

注 : このサンプルのキーワード **ByVal** の使用位置に注意してください。

構文

Function WriteCmdLineVB (ByVal nTaskIndex As Integer,
_ szCmdLine As String) As Integer

引き数

表 6.24　WriteCmdLineVB の引き数

引き数	データ型	意味
nTaskIndex	整数	REGIE.INI ファイルの［TaskConfiguration］セクションに指定されるタスクの索引。
szBuf	ストリング	任意の長さのコマンド行（0 終了の C ストリング）のバッファ

戻り値 :

0　　エラー。タスクが欠落しているか，またはコマンド行はすでにインストールされています。

n　　終了の 0 文字を除いて，バッファに書き込まれた文字数

例

Visual Basic で WriteCmdLineVB 関数を呼び出すには，次のようにします。

例 6-48　Visual Basic での呼び出し

```
Dim szString As String
Dim i As Integer
...
szString = ...
```

```
i = WriteCmdLineVB(24, szString)
```

■ WriteCmdLineVBEx

説明

VB プログラムで「子」領域のコマンド行をインストールします。領域は，タスク番号で指定しなければなりません。

宣言

```
Declare Function WriteCmdLineVBEx Lib "Regie"
_ (ByVal sTaskName As String,
_ CmdLine As String) As Integer
```

注 : このサンプルのキーワード ByVal の使用位置に注意してください。

構文

Function WriteCmdLineVBEx (ByVal szTaskName As String,_ szCmdLine As String) As Integer

引き数

表 6.25 WriteCmdLineVBEx の引き数

引き数	データ型	意味
szTaskName	ストリング	REGIE.INI ファイルの［TaskConfiguration］セクションに指定されるタスクの名前。
szBuf	ストリング	任意の長さのコマンド行（0 終了のストリング）のバッファ

戻り値 :

0　エラー。タスクは利用不能であるか，またはコマンド行はすでにインストールされています。

n　終了の 0 を除いて，バッファに書き込まれた文字数

Visual Basic での呼び出し

Visual Basic で WriteCmdLineVBEx 関数を呼び出すには，次のようにします。

例 6-49 Visual Basic での呼び出しの宣言

INI-Datei 内のエントリは次のとおりです。

```
[CHILDS]
HUGO=2
```

次のように呼び出します。

```
Dim i As Integer
i = WriteCmdLineVBEx("HUGO","command line for Task 2").
```

6.4.7 その他

表 6.26 は，REGIE.DLL のその他の関数の概要を示しています。

表 6.26 REGIE.DLL の関数（その他）

名前	摘要
その他	
GetMMCDir	MMC ディレクトリを判別します
GetMMCLanguagePath	現行タスクの DLL にアクセスします
InitComplete	初期化の正常完了を示します
StopRegieEvents	新規イベントの起動を阻止します
ResumeRegieEvents	StopRegieEventsT を再開します
estAndStopRegieEvents	StopRegieEvents を拡張します
SetModeChannelSwitchKey	チャネル切り替えキーをロックします

GetMMCDir

説明

MMC ディレクトリを判別します。

GetMMCDir は，呼び出し側タスクの開始元のディレクトリである ***MMC*** ディレクトリを戻します。

この関数を使って，MMC ディレクトリのサブディレクトリ内に置かれていなければならないタスク別の一時ファイルにアクセスします。

重要な注意 **:** MMC ソフトウェアで絶対パス名をプログラムすることはお勧めしません。絶対パス名を使うと，1 つの PC で複数のバージョンの MMC ソフトウェアを処理できなくなるからです。

宣言

```
Declare Function GetMMCDir Lib "Regie"
_ (ByVal hWnd As Integer, ByVal szBuf As String,
_ ByVal nLen As Integer) As Integer
```

構文

Function GetMMCDir (ByVal hWnd As Integer,

_ ByVal szBuf As String, ByVal nLen As Integer) As Integer

引き数

表 6.27 GetMMCDir の引き数

引き数	データ型	意味
hWnd	整数	呼び出し側タスクの WINDOW ハンドル
szBuf	ストリング	MMC ディレクトリを戻すために関数が使うバッファのアドレス
nlen	整数	lpszTaskExePath の長さ

戻り値

書き込まれたバイト数

例

引き数を渡す場合，ストリングは C または C++ に互換でなければならないことに注意してください。以下にその方法の 1 つの例を示します。

例 6-50 Visual Basic での GetMMCDir の使用

```
Dim szCBuf As String * 128 ' C-String compatible
Dim szVBExePath As String ' Visual Basic String
Dim nlen As Integer
...
nlen = GetMMCDir (hWnd, szCBuf, Len (szCBuf))
vbExePath = Left$ (szCBuf, nlen)
hRegieDll = LoadLibrary (szVBExePath)
```

C での呼び出し

```
int WINAPI
GetMMCDir (HWND hWnd, LPSTR lpszTaskExePath,_ int nlen)
```

例 6-51 C での GetMMCDir の使用

```
include <windows.h>
...
int len;
HWND hWnd;
char szBuf [128];
HINSTANCE hRegieDll;
int (WINAPI * lpfnGetMMCDir) (HWND, LPSTR, int);
...
hRegieDll = LoadLibrary ("Regie");
..
lpfnGetMMCDir = (int (WINAPI *) (HWND, LPSTR, int))
GetProcAddress (hRegieDll, "GetMMCDir");
...
// call using ANSI-C
len = lpfnGetMMCDir (hWnd, szBuf, sizeof (szBuf));
// call not using ANSI-C
len = (lpfnGetMMCDir *) (hWnd, szBuf, sizeof (szBuf));
```

アプリケーションのローカルテスト

まだ開発ディレクトリで開発途中にある領域アプリケーションをローカルにテストすることがよい場合があります。そのために GetMMCDir には，ローカルファイルと言語別 DLL をすべて備えた有効な MMC システムディレクトリのパスが用意されています。

対処策 :

1.WPS を使って, 要求先の MMC システムディレクトリから REGIE.DLL を手動でロードします。

2. ヌルハンドル（HWND）0 を GetMMCDir 引き数 hWnd に渡します。こうすると

GetMMCDir は，REGIE.DLL のロード元（最初のロード）のパスを戻します。

■ GetMMCLanguagePath

説明

タスクの現在の DLL にアクセスします。

GetMMCLanguagePath を使って，現在タスクで設定されている言語別のダイナミックリンクライブラリにアクセスすることができます。

GetMMCDir に関する注意事項は状況に応じて適用されます。

宣言

```
Declare Function GetMMCLanguagePath Lib "Regie"
_ (ByVal hWnd As Integer,
_ ByVal szPrefix As String,
_ ByVal szBuf As String,
_ ByVal nlen As Integer) As Integer
```

構文

```
Function GetMMCLanguagePath (ByVal hWnd As Integer,
_ ByValszPrefix As String,
_ ByVal szBuf As String,
_ ByVal nLen As Integer) As Integer
```

引き数

表 6.28 GetMMCLanguagePath の引き数

引き数	データ型	意味
hWnd	整数	呼び出し側タスクの Window ハンドル
szPrefix	ストリング	MMC 領域アプリケーションの略称（通常は 2 文字で構成しなければなりません）
SzBuf	ストリング	MMCパスを戻すために関数が使うバッファのアドレス
nlen	整数	lpszDLLPath の長さ

戻り値 :

書き込まれたバイト数

C での呼び出し :

```
int WINAPI
GetMMCLanguagePath (HWND hWnd, LPSTR lpszPrefix,
                    _ LPSTR lpszDLLPath, int nlen)
```

例 :

略称 OE を使用する OEM アプリケーションは，現在の言語 DLL を次のように設定し

ます。

例 6-52 C の，現在選択されている DLL の判別

```
# include <windows.h>
...
int len;
HWND hWnd;
char szBuf [128];
HINSTANCE hRegieDll;
int (WINAPI * lpfnGetMMCPath) (HWND, LPSTR, LPSTR, int);
...
hRegieDll = LoadLibrary ("Regie");
...
lpfnGetMMCLanguagePath = (int (WINAPI *) (HWND, LPSTR, LPSTR, int))
GetProcAddress (hRegieDll, "GetMMCLanguagePath");
...
// Call in ANSI-C
len = lpfnGetMMCLanguagePath (hWnd, "OE", szBuf, sizeof (szBuf));.
```

szBuf に戻されるストリングは，次のようなものです。:

D:¥MMC¥SW_42¥LANGUAGE¥OE_GR.DLL

注 **:** 略称 **GR** から分かるとおり，ドイツ語バージョンは現在，次のように **MMC** システムファイル **MMC.INI** 内で構成されています。
`[Language]`
`Language=GR`
言語別の DLL は，OE_GR.DLL と呼ばれ，MMC システムディレクトリ D:¥MMC¥SW_42 のサブディレクトリ LANGUAGE に置かれます。

■ InitComplete

説明

初期化が正常に完了したことを示します。

領域アプリケーションは，内部初期化の終了を Regie に知らせる必要があります（同期のため)。そのために領域アプリケーションは，初期化の終了時に InitComplete を呼び出して，そのタスクハンドルとメインウィンドウのハンドルを渡します。

注 **: REGIE.INI** の［**StartupConfiguration**］セクションに入力されたすべてのサーバは，**InitComplete** を呼び出さなければなりません。

注 **:** シーケンス制御を備えた **OEM** 領域アプリケーションを作成する場合や，置き換え **OEMFRAME** を使用する場合，**InitComplete** を考慮する必要はありません。シーケンス制御では，**OEM** フレームにその呼び出しがすでに暗黙指定されている（**MDIFORM1.FRM** 内で）からです。

宣言

```
Declare Sub InitComplete Lib "Regie"
_ (ByVal hTask As Integer,
```

```
_ ByVal hOwnerWnd As Integer,
_ ByVal hControlWndTask As Integer)
```

構文

Sub InitComplete (ByVal hTask As Integer,

_ ByVal hOwnderWnd As Integer,

_ ByVal hControlWndTask As Integer)

引き数

表 6.29 InitComplete 関数の引き数

引き数	データ型	意味
hTask	整数	タスクハンドル（Windows API の GetCurrentTask で判別されます）
hOwnerWnd	整数	メインウィンドウのウィンドウハンドル（SW 3.1 以後）：（通常は CreateWindow によって作成されます）
hControlWnd	整数	Regie 制御のウィンドウハンドル。シーケンス制御がないアプリケーションでは 0 に設定します

Visual Basic での呼び出し

シーケンス制御を使用するアプリケーションでは，シーケンス制御が次のように関数を呼び出します。

例 6-53 Visual Basic での呼び出し

```
Call InitComplete (GetCurrentTask(), Mdiform1.hWnd,
RegieControl.hWnd)
```

C での呼び出し

```
void WINAPI
InitComplete (HTASK hTask,
 HWND hOwnerWnd, HWNDhControlWnd)
```

例

シーケンス制御を使わない InitComplete の典型的な呼び出し例は次のとおりです。

例 6-54 C の，InitComplete の使用

```
# include <windows.h>
...
HINSTANCE hRegieDll;
HTASK hTask;
HWND hOwnerWnd; //look at CreateWindow
void (WINAPI * lpfnInitComplete) (HTASK, HWND, HWND);
...
hRegieDll = LoadLibrary ("Regie");
...
lpfnInitComplete = (void (WINAPI *) (HTASK, HWND, HWND))
GetProcAddress (hRegieDll, "InitComplete");
```

```
...
lpfnInitComplete (GetCurrentTask (), hOwnerWnd, (HWND) 0); // ANSI-
C
```

■ StopRegieEvents, ResumeRegieEvents

説明

イベントが活動化されないようにして再びロック解除します。

たとえば，キーが早く押された場合に，現在のイベントが終了するまで他のイベントが発生しないようにしたい場合，StopRegieEvents を使って新規のイベントが活動化されないようにします。ResumeRegieEvents 関数は，StopRegieEvent を取り消します。

注 **: StopRegieEvents** を呼び出した場合，その後で必ず **ResumeRegieEvents** を呼び出すようにしてください。

注 **:** 新規アプリケーションを作成する場合，**StopRegieEvents** 関数は今後使用するべきではありません。代わりに，以下に説明されている **TestAndStopRegieEvents** 関数を使います。

宣言

```
Declare Sub StopRegieEvents Lib "Regie" ()
Declare Sub ResumeRegieEvents Lib "Regie" ()
```

構文

Sub StopRegieEvents ()

Sub ResumeRegieEvents ()

引き数

なし

VB での呼び出し

Visual Basic で関数を呼び出すには，次のようにします。

例 6-55 Visual Basic での呼び出し

```
StopRegieEvents
...
ResumeRegieEvents
```

C での呼び出し

```
BOOL WINAPI
StopRegieEvents (void) ; prevents events from being
; activated
BOOL WINAPI
ResumeRegieEvents (void); resume StopRegieEvents
```

■ TestAndStopRegieEvents

説明

Regie が停止されたかどうかを調べて，イベントが活動化されないようにします。

これは，StopRegieEvents 関数の拡張です。

宣言

```
Declare Function TestAndStopRegieEvents Lib "Regie"
_ () As Integer
```

構文

Function TestAndStopRegieEvents () As Integer

引き数

なし

戻り値 :

0: Regie はすでに停止されました（呼び出し元自身によってと考えられます)。

1: Regie は停止されませんでした。ただし現在は停止していて，ResumeRegieEvents で解放しなければなりません。

注 **: Regie** が別のアプリケーションによって停止されていることが確かめられた場合（戻り値 **0**），**ResumeRegieEvents** を呼び出すと，他のアプリケーションに望ましくない影響を与えることがあります。したがって，自分自身のアプリケーションで **Regie** を停止していない限り，**ResumeRegieEvents** の呼び出しには注意が必要です。

アプリケーション

Visual Basic で関数を呼び出すには，次のようにします。

例 6-56 TestAndStopRegieEvents の使用

典型的な部分コードの例 :

```
Dim nSema As Integer
nSema = TestAndStopRegieEvents ()
' area protected from the Regie
....
If nSema = 1 Then ResumeRegieEvents
```

C での呼び出し

```
BOOL WINAPI
TestAndStopRegieEvents (void) ; checks stop of Regie,
;blocks new events
```

■ SetModeChannelSwitchKey

説明

SetModeChannelSwitchKey 関数は，チャネル切り替えキーをロックまたはロック解除します。

注 **:** チャネル切り替えキーがロックされている限り，チャネル切り替えが必要になった場合は，**MMC** ローカル **NCDDE** 変数 **/nck/nck/channel** が正しく設定されていることを **OEM** アプリケーションで確認する必要があります。

宣言

```
Declare Sub SetModeChannelSwitchKey Lib "Regie"
_ (ByVal nMode As Integer)
```

構文

```
Sub SetModeChannelSwitchKey (ByVal nMode As Integer)
```

引き数

表 6.30 SetModeChannelSwitchKey の引き数

引き数	データタイプ	意味
nMode	整数	nMode = 0　チャネル切り替えキーをロックします nMode = 1　チャネル切り替えキーを活動化します

C での呼び出し

void WINAPI
SetModeChannelSwitchKey (int nMode)

例 6-58 C の，チャネル切り替えキーのロック

```
void (WINAPI * SetModeChannelSwitchKey (int nMode);
...
(*SetModeChannelSwitchKey) (0);.
```

6.5 Regie の Visual Basic コントロール

この章では，Regie 用の Visual Basic 拡張（VBX データファイル）についてさらに詳しく考察します。

SW リリース 3.2 以後の OEM パッケージには, 新しいコントロール RECTLP32.VBX が入っています。これは，それまでのコントロール REGIECTL.VBX の機能に，コマンドインタープリタチャネルの機能を追加したものです。

コントロール RECTLP32.VBX はシーケンス制御で使われます。このコントロールは，フレーム MDIFrame 内にあります（ALSTART.FRM を参照）。シーケンス制御のユーザは，シーケンス制御を介してこのコントロールに直接アクセスします。独自のシーケンス制御を作成するユーザは，コントロールのイベントに直接アクセスします。

6.5.1 Regie コントロール RECTLP32.VBX

概説

デフォルトでは，Regie コントロールはフレーム MDIFrame（ALSTART.FRM も参照）内に置かれます。このファイルは変更できません。したがって，イベント内でシーケンス制御の状態（PRIVATE.BAS 内の State_Changed）を並列して呼び出して，ユーザはそれにフックすることができます。

Regie コントロール：	シーケンス制御：
Activate	AL_ACTIVATE
Deactivate	AL_DEACTIVATE
NoDeactivate	AL_NODEACTIVATE
FormLoad	AL_FORMLOAD
FormUnload	AL_FORMUNLOAD
NoFormUnload	AL_NOFORMUNLOAD
QueryForShutdown	

データファイル ALDEFINE.BAS は，次のように該当するエントリで拡張されます。

```
Global Const AL_ACTIVATE = ...
Global Const AL_DEACTIVATE = ...
...
```

注 **: Regie** コントロールの **Deactivate** イベントと，アプリケーションの非活動化を混同しないでください（「領域アプリケーションの終了」を参照してください）。

Regie コントロールのイベント

以下に，このコントロールのすべてのイベントを説明します（プロパティは重要ではありません）。一連のイベントには以下の 2 種類があり，これらは領域から領域への切り替え時に発生します。

同一領域のオンとオフの切り替え：

Deactivate Activate

領域から領域への切り替え：
Deactivate FormUnload FormLoad Activate

表 6.31 は，RECTLP32.VBX のイベントを示しています。

これらのイベントは，Visual Basic の関数にちなんで命名された Regie 関数としてインプリメントされています。

表 6.31 RECTLP32.VBX のイベント

イベント	意味
Activate	オペレータは領域を選択しました
Deactivate	オペレータは F10 キーを使って領域バーを選択しました
FormLoad	アイコン化領域が選択されました
FormUnload	領域の選択が解除されました
QueryForShutdown	アプリケーションをクローズします

■ Activate

説明

これは，ユーザによる領域の活動化後に起動されます。ACTIVATE の状態処理の後でフォーカスを変更することはできません。活動中の MDI-child ウィンドウに Regie はフォーカスを置くからです。アプリケーションは，モジュール PRIVATE.BAS 内の State_Changed を呼び出して，Regie イベント Activate に関する通知を受けます。

通知

Completed = State_Changed ("-990",AL_ACTIVATE,_ AL_ACTIVATE)

戻り値

1 正常完了

注：他の戻り値は使えません。ロードされた領域は常に活動化できなければなりません。

Deactivate

説明

これは，ソフトキー F10 を使って領域バーを活動化し，アプリケーションを非活動化した後（フォーカスは Regie のデータ領域バーに切り替えらる）で起動されます。アプリケーションでは機能キーを使えなくなります。新しい領域が選択されると，FormUnload イベントが Deactivate イベントの後に続きます。アプリケーションは，PRIVATE.BAS モジュール内の State_Changed を呼び出して，Regie イベント Deactivate の通知を受けます。

通知

```
Completed = State_Changed ("-991",AL_DEACTIVATE,_ AL_DEACTIVATE)
```

戻り値

1. Okay
2. *Deactivate* は受け入れられません。実際のアプリケーションは活動中のままになるので，*Regie* はアラームメッセージを出す必要があります。

   この時点でマシン領域を非活動化することはできません。

   もう一度やり直してください。
3. 戻り値 2 と同じです。ただし，Regie はアラームメッセージを出しません。アプリケーションで，適切なアラームメッセージを生成する必要があります。

利点 : アプリケーションは，Regie よりも詳細なメッセージを生成することができます。

ヒント : Deactivate に対する否定応答は，緊急時のみに限定する必要があります。以下に使用可能な例を示します。:

- ダイアログのプログラミング : シミュレーションのロード
- 診断 : 構成の判別

このイベントに対する否定応答の直後に，NoDeactivate イベントが生成されます。これに対してユーザは，それに応じた独自のコマンドをプログラミングすることができます。

注 **: Deactivate**（戻り値 **1** または **2** の場合）の後，アプリケーションは，フォーカスに影響を与える可能性のあるコマンドを処理できません（非同期アクションなどでは注意してください)。
ここで（または，遅くとも FormUnload イベントと一緒に）アプリケーションは，領域の非活動化に対して準備ができています。
2 つのイベントのどちらを使うかは，次のように，関数の種類によって決まります。
フォーカスに影響を与える関数は，この時点で停止する必要があります。
アプリケーションのアンロード時にのみ動作する関数は，FormUnload イベントにシフトさせることができます。

■ NoDeactivate

説明

Deactivate の説明を参照してください。

通知

```
Completed = State_Changed ("-994", AL_NODEACTIVATE,_
AL_NODEACTIVATE)
```

戻り値 :

1 正常完了

■ FormLoad

説明

アイコン化領域が再活動化されると，このイベントが起動されます。シーケンス制御は，それ以前の FormUnload でアンロードされたすべての MDI-Child に対してロードコマンドを実行します。これと関連して，Regie の FormLoad イベントは，PRIVATE.BAS モジュール内の State_Changed 関数を呼び出します。

アプリケーションは，すべての DDEML リンク（LinkMode=1 または LinkMode=2）を活動化します。FormLoad イベントは Activate イベントのすぐ後に続きます。

通知

```
Completed = State_Changed ("-992",AL_FORMLOAD,_ AL_FORMLOAD)
```

戻り値

1 正常完了

注 : 否定応答は使えません。背景のアイコン化領域は，常にロード可能でなければなりません。

■ FormUnload

説明

領域の非活動化時に起動されます。シーケンス制御は，.MDI ファイル内でアンロード可能と指定されているすべての MDI-Child に対してアンロードコマンドを実行します。

アプリケーションは，DDEML リンク（LinkMode=0）のクローズを処理しなければなりません。肯定応答（Completed=1）後にアプリケーションはアイコン化されます（WindowState=1）。

イベントが否定応答された（戻り値 2 または 3 で）場合，NoFormUnload 通知とともに，ただちに State_Changed が呼び出されます。この場合ユーザは，自分独自のコマンドをプログラミングすることができます。

通知

```
Completed = State_Changed ("-993",AL_FORMUNLOAD,_ AL_FORMUNLOAD)
```

戻り値

1 正常完了

2 FormUnload は受け入れられません。アプリケーションは Activate によって活動化され，Regie はアラームメッセージを出す必要があります。

この時点でマシン領域を非活動化することはできません。

もう一度やり直してください。

3 戻り値 52 と同じです。ただし，アプリケーションから適切なアラームメッセージを生成する必要があります。

ヒント : FormUnload の否定応答は，Deactivate ほど効力は高くありません（Regie が Activate を使ってもう一度アプリケーションを活動化する必要があります)。

■ NoFormUnload

説明

FormUnload イベントの説明を参照してください。

構文

```
Completed = State_Changed ("-995", AL_NOFORMUNLOAD,
_AL_NOFORMUNLOAD)
```

戻り値

1 正常完了

■ QueryForShutdown

説明

背景情報 :

SW 3.1 以後，アプリケーションをクローズするために次の 2 とおりの方法が使えるようになりました。

- 通常操作時にクローズする
- Regie のクローズ時にクローズする（PC バージョン）

通常操作時のクローズ :

アプリケーションの構成によっては（REGIE.INI で説明)，すべてのアプリケーションを同時にメモリに置くことができない場合もあります。したがって，アプリケーションによっては，領域の切り替え時にクローズする必要があります。

理想的なケースではこれは「ひそかに」実行されます。Regie が QueryForShutdown イベントを発行すると，アプリケーションは肯定応答を出してから，それ以上ユーザと対話せずにクローズします。

通常，QueryForShutdown 要求は，背景でアイコン化されたままのアプリケーションに対して使用することが推奨されています。

シーケンス制御の使用時には，Regie では，QueryForShutdown イベントによって SHUTDOWN.BAS モジュール内の QueryForShutdown 関数が呼び出されます。ここでユーザは，終了するかどうかの質問に対する応答をプログラミングすることができます。

例

SW 2.2 ではユーザがマシンから診断への切り替えを行います。QueryForShutdown は，ダイアログプログラミングに送られます。

ユーザが QueryForShutdown に対して NO と返答した場合，Regie はユーザとの対話を開始します。これは，Windows 3.11 に似ています（メモリまたはリソースが不足した場合のアプリケーションのオープンの項を参照)。その場合（ダイアログプログラミングのロードやマシンから診断への切り替えなどのとき)，次のようなメッセージが出されます。

Not enough memory for area Diagnosis.（領域診断のメモリが足りません）

Please deactivate the area Dialog programming.（領域ダイアログプログラミングを非活動化してください。）

このメッセージに対してユーザが肯定応答すると，システムはマシン領域にとどまるので，ユーザは，ダイアログプログラミングを非活動化する（たとえばマシンデータを設定するため）か，または別のアクションを行うかを決めることができます。

領域アプリケーションの結果 **:**

どの領域アプリケーションも，QueryForShutdown イベントに対して NO と応答するためには，Deactivate アクションをもつソフトキーをメインメニュー内に用意する必要があります。領域が非活動された後，別の QueryForShutdown に対しては YES で肯定応答しなければなりません。

Regie のクローズ時のアプリケーションのクローズ **:**

上記の方式は，Regie のクローズ時には有効ではありません。メッセージが正しくないからです。領域アプリケーションが Regie によってクローズされることになっている場合，関連するすべてのアプリケーションに対して QueryForShutdown イベントが発行されます。これに YES と応答することは，アプリケーションが「ひそかに」クローズされることを意味します。アプリケーションが QueryForShutdown に対して NO と応答すると，Windows 3.11 に似たユーザダイアログが開始されます。活動中の MS-DOS 入力行を使った Windows のクローズの項を参照してください。

Regie は次のようなメッセージを発行します。

Area Diagnosis still active.（領域診断はまだ活動中です）

Please deactivate that area.（領域を非活動化してください）

ユーザがメッセージに肯定応答すると，Regie は，非活動する対象の領域を選択します。Regie をクローズするソフトキーは無視されます。

通知

```
Result = QueryForShutdown (Context)
```

パラメータ Context は，次のように，Regie から発行されたとおりの QueryForShutdown イベントの環境を示します。

「通常操作時の領域切り替え」というコンテキストは，以下に対応します。

```
Global Const CONTEXT_TASK_EXIT = 1
```

「Regie のクローズ時（PC バージョン)」というコンテキストは，以下に対応します。

```
Global Const CONTEXT_REGIE_EXIT = 2
```

どちらの場合も，QueryForShutdown イベントは，モジュール SHUTDOWN.BAS 内の QueryForShutdown 関数（同じ名前）を呼び出します。

例 6-59 QueryForShutdown 関数

```
Function QueryForShutdown (Context As Integer) As Integer
If Context <> CONTEXT_TASK_EXIT And Context <> CONTEXT_REGIE_EXIT
Then
QueryForShutdown = 0 'internal error: wrong argument CONTEXT
Else
QueryForShutdown = 1 'everything OK
Rem QueryForShutdown = 2 'Regie sends a message
Rem QueryForShutdown = 3 'application must respond
End If
End Function
```

ユーザは，実際のコンテキストに応じて異なる戻り値が正しいかどうかに注意する必要があります。

「領域切り替え」コンテキストでの戻り値

1. はい。アプリケーションをクローズすることができます。

2. いいえ。アプリケーションを「ひそかに」クローズすることはできません。

以後のユーザ対話は Regie によって制御され，前のアプリケーションは活動中のままになります。

3. 要求された領域は，この要求を格納し，状態の切り替えを内部的に行うことができます。次回の活動化のときに，該当するメッセージが生成されます。

「Regie のクローズ」コンテキストでの戻り値 :

1. はい。アプリケーションをクローズすることができます。

2. いいえ。アプリケーションを「ひそかに」クローズすることはできません。

Regie は適切なメッセージ（「背景情報」を参照）を発行し，その領域に切り替えます。その後，アプリケーションをクローズできる状態にするのは，ユーザの責任です(「アプリケーションの非活動化」の項を参照)。

3. 戻り値 2 と同じです。ただし，Regie はメッセージを出しません。選択された領域が，メッセージを生成する必要があります。

注 **:** アイコン状態の領域で実行可能な関数だけを使えます。さらに，**Windows** システム内の実際のフォーカスは変更できません。

6.6 OEM フレームを使った Windows プログラムの組み込み

Windows アプリケーションを領域アプリケーションとして処理するには，次のステップを実行する必要があります。

1. プログラムファイルを OEM ディレクトリにコピーします。

2. プレースホルダアプリケーション OEMFRAME.EXE を使って, OEM ディレクトリの REGIE.INI ファイル内の TaskConfiguration セクションにエントリを作成します。

3. たとえばポインティングデバイス（マウス）などのための必要なドライバをインストールします。

4. :C¥OEM¥LANGUAGE ディレクトリの softkeytext ファイルにソフトキーテキストを入力します。

5.OEM ディレクトリの OEMFRAME.INI ファイルの［PROGRAM NAME］セクションに特殊値を入力します。

注 **: EXCEL** などの **Windows** アプリケーションを，**REGIE.INI** ファイルの［**TaskConfiguration**］セクションに直接入力することはできません。

例 6-60 REGIE.INI のエントリ

```
[TaskConfiguration]
Task2 = Name := EXCEL ; This does not work !
Task2 = Name := OEMFRAME, CmdLine := "EXCEL.EXE" ; but this will
```

注 **: OEM** フレームは，開始されたアプリケーションをモニタします。そのアプリケーションがクローズすると，**OEM** フレームそのものもクローズします。

6.6.1 **REGIE.INI** ファイルのエントリ

Windows プログラムは，置き換えアプリケーション OEMFRAME にリンクされます。

例 6-61 REGIE.INI のエントリ

```
[TaskConfiguration]
Task2 = Name := OEMFRAME, CmdLine := "C:¥'
_ ClassName := "OpusApp",
_ WindowName := "Microsoft Word"
; Calling the word processing program WinWord with ClassName
; and WindowName and opening the text file Text.DOC..
```

注 **:** パスを指定する場合，単一の円記号ではなく，二重円記号（¥¥）

注 **: REGIE.INI** ファイルでプログラムを識別するには，クラス名またはウィンドウ名を入力するだけで十分です。

注 **: CmdLine** パラメータを指定するときに，ディレクトリ名またはファイル名にスペース文字を入れることはできません。

属性 :

OEMFRAME の有効な属性は次のとおりです。

表 6.32 OEMFRAME の属性

属性	意味
CmdLine	Windows の標準タスク名
ClassName	所有者ウィンドウハンドルの情報
WindowName	所有者ウィンドウハンドルの情報

ソースコードで使えるクライアントアプリケーションは，ClassName と WindowName の情報に直接アクセスすることができます。Windows-API 呼び出しの RegisterClass および CreateWindow をご覧ください。

ヒント : ソースコードを使えない場合，**WinWalk** などのパブリック・ドメインツールを使って文字列を確かめることができます。

6.6.2 OEMFRAME.INI ファイルのエントリ

必要があれば，OEMFRAME.INI 初期化ファイルを使って，置き換えアプリケーション OEMFRAME を追加してパラメータ化することができます。ただし，通常はその必要はありません。

それぞれの Windows プログラムには，特殊セクションが割り当てられます。アプリケーションが画面に表示される方法は，セクション内の属性によって構成されます。図 6.33 にその概要を示しています。

表 6.33 OEMFRAME.INI ファイル内のセクション（例 : エディタのメモ帳）

セクション	意味
notepad	属性を指定してエディタを表示します。 WindowStyle_On = WindowStyle_Off = x =0 y = 100 Width = 560 Height = 325 nDelayInitComplete = xx nSecondsToFindWindow = 30

ウィンドウスタイル

追加情報 : Windows アプリケーションの画面の外観は，GWL_STYLE 引き数を指定した Windows-API 関数 GetWin-dowLong および SetWindowLong で決まります。これは，表 6.34 で示しているとおり，8 バイトの長さのワードで制御します。属性

WindowStyle_On および WindowStyle_Off で，この 2 つのバイト（下記の表にマークを付けて詳述されています）を変更できます。

表 6.34 WindowStyle 属性を使ったディスプレイの構成

0000	0000	xxxx	xxxx	0000	0000	0000	0000
		1010		表題			
		1000		枠			
		0100		dlgframe			
		0010		垂直スクロールバー			
		0001		水平スクロールバー			
			1000	システムメニュー			
			0100	Thickframe			
			0010	最小化ボックス			
			0001	最大化ボックス			

2進値が10進数としてWindowStyle属性に割り当てられます。2進数から10進数への変換またはその逆の変換には，通常は［アクセサリ］プログラムグループにある Windows 電卓を使います。

DeterminingWINDOW style

例 6-62 ウィンドウスタイルの指定

［プロパティ］システムメニューの他に，水平および垂直スクロールバーの特徴を決めます。表 6-34 に従うと，以下のようになります。
0000 0000 0011 1000 0000 0000 0000 0000 binary or
0038 0000 Hex.
次に電卓を呼び出して次のようにします。

- ［16 進］ボタンをクリックします。
- 380000 と入力します（先行ゼロ省略できます）。
- ［10 進］ボタンをクリックします。
- ［編集］メニューで［コピー］を選択して，計算結果 3670016 をコピーします。
- 結果を属性に挿入します。

WindowStyle_On 属性 :

この属性は，ウィンドウに割り当てられるプロパティの特徴となります。

例 6-63 WindowStyle_On 属性 : ［メモ帳］エディタは，システムメニュー，水平および垂直スクロールバー付きで表示されます。

```
[notepad]
WindowStyle_On = 3670016.
```

WindowStyle_Off 属性 :

この属性は，ウィンドウに割り当てられないプロパティをオフにします。

例 6-64 WindowStyle_Off 属性 : ［最大化］ボックスおよび［最大化］ボックスなしで［メモ帳］エディタを表示します。

```
[notepad]
WindowStyle_Off = 196608
```

X 属性と Y 属性 :

これらの属性は，画面の左上隅を原点として，Windows アプリケーションの開始座標を指定します。X は水平座標，Y は垂直座標であり，下に向かって数字が増加します。計測単位はピクセルです。使用可能な作業域は 560 x 325 ピクセルです。

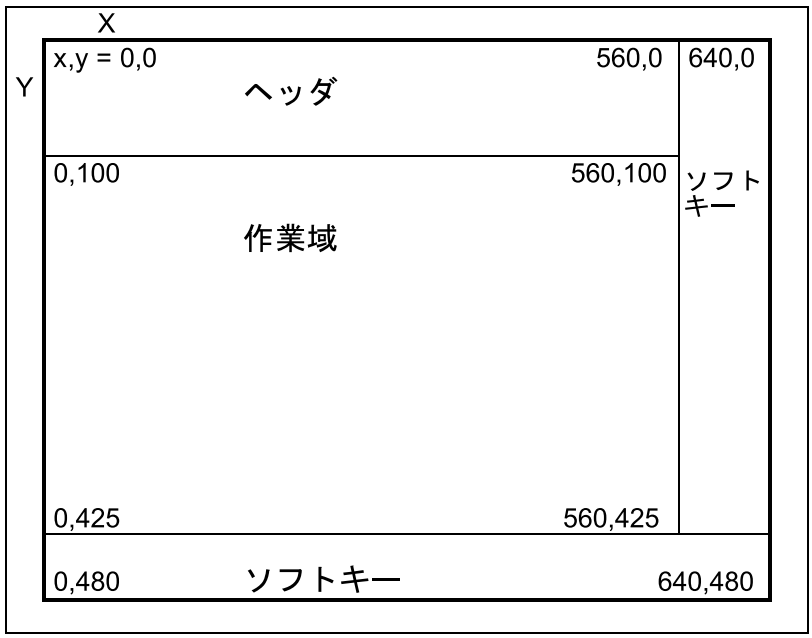


図 6.5 ピクセル単位の座標系 x と y での画面の配列

注 **:** アプリケーションの「ヘッダ」には，**Y** 方向に **100** ピクセル必要です。したがって，ヘッダがアプリケーションの上にかぶさらないようにするには，**Y** を **100** 以上に設定しなければなりません。これは図 **6.5** で示されています。エディタはヘッダのすぐ下に置かれます。

Width 属性 :

この属性は，ピクセル単位の X 属性に従って原点をとり，Windows アプリケーションのウィンドウの幅を設定します。

Height 属性 :

この属性は，ピクセル単位の Y 属性に従って原点をとり，Windows アプリケーションのウィンドウの高さを設定します。

デフォルト設定

Windows アプリケーションが REGIE.INI ファイルの［TaskConfigura-tion］セクションに入力されても，OEMFRAME.INI ファイル内に特別なセクションがない場合，そのデフォルト値は次のようになります。

WindowStyle_On = 0

WindowStyle_Off = 0

x =0

y =0

Width = width (DesktopWindow)

Height = height (DesktopWindow)

注 : 値が **x = 0** と **y = 0** の場合，アプリケーションの「ヘッダ」は覆われてしまいます。

例 6-65 図 6-6 の場合の OEMFRAME.INI ファイルの設定

```
[program]
WindowStyle_On = 0
WindowStyle_Off = 0
x = 100
y = 150
Width = 300
Hight = 120
```

設定例

図 6.6 は，アプリケーション「プログラム」用に構成されたウィンドウを示しています。

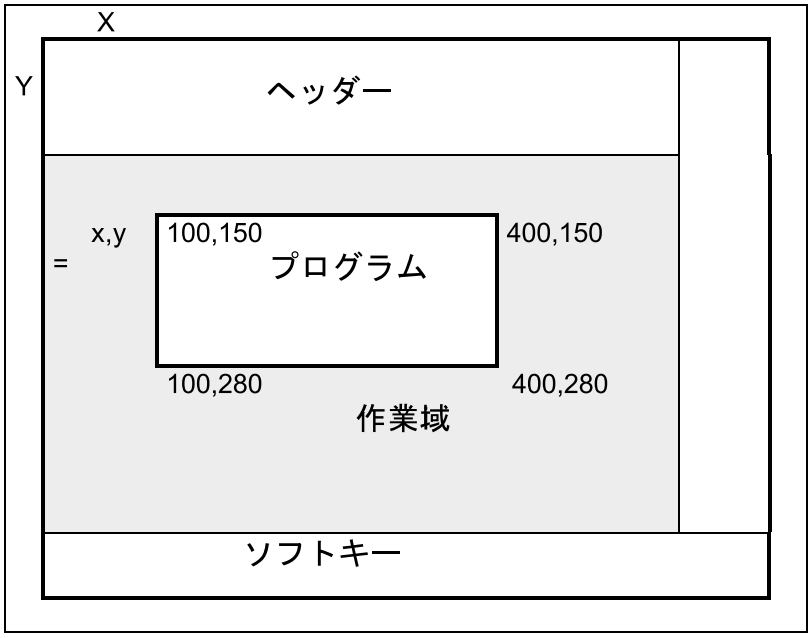


図 6.6 Windows アプリケーション「プログラム」で設定された画面座標の例

nDelayInitComplete 属性

OEMFrame.ini ファイル内のエントリによって, WM_INITCOMPLETE の送信を遅らせたり抑制したりすることができます。そのためには，OEMFrame.ini ファイル内の対

応するセクション内に，この nDelayInitComplet=xx を入力しなければなりません。ここで xx は，マイクロ秒単位の時間を表します。

Attrigbute nSecondsToFind Windows:

OEM アプリケーションによっては（たとえばステップ 7），画面を表示するまでに長時間かかることがあります。OEM フレームは，一定時間（20 秒まで）だけアプリケーションウィンドウを待機します。一定時間が過ぎると，アプリケーションは開始されていないとみなして終了します。アプリケーションをもう一度選択すると，再始動が実行されてしまいます。そのため，次のような修正が加えられました。

1. 最大待ち時間が 20 秒から 40 秒に延長されました。

2.OEMFrame.INI を使って待ち時間を構成することができるようになりました。

<ApplName>]

nSecondsToFindWindow = ...

注 **:** この値は，**REGIE.INI** ファイル内のタイムアウト値に対応していなければなりません。

例 **:**

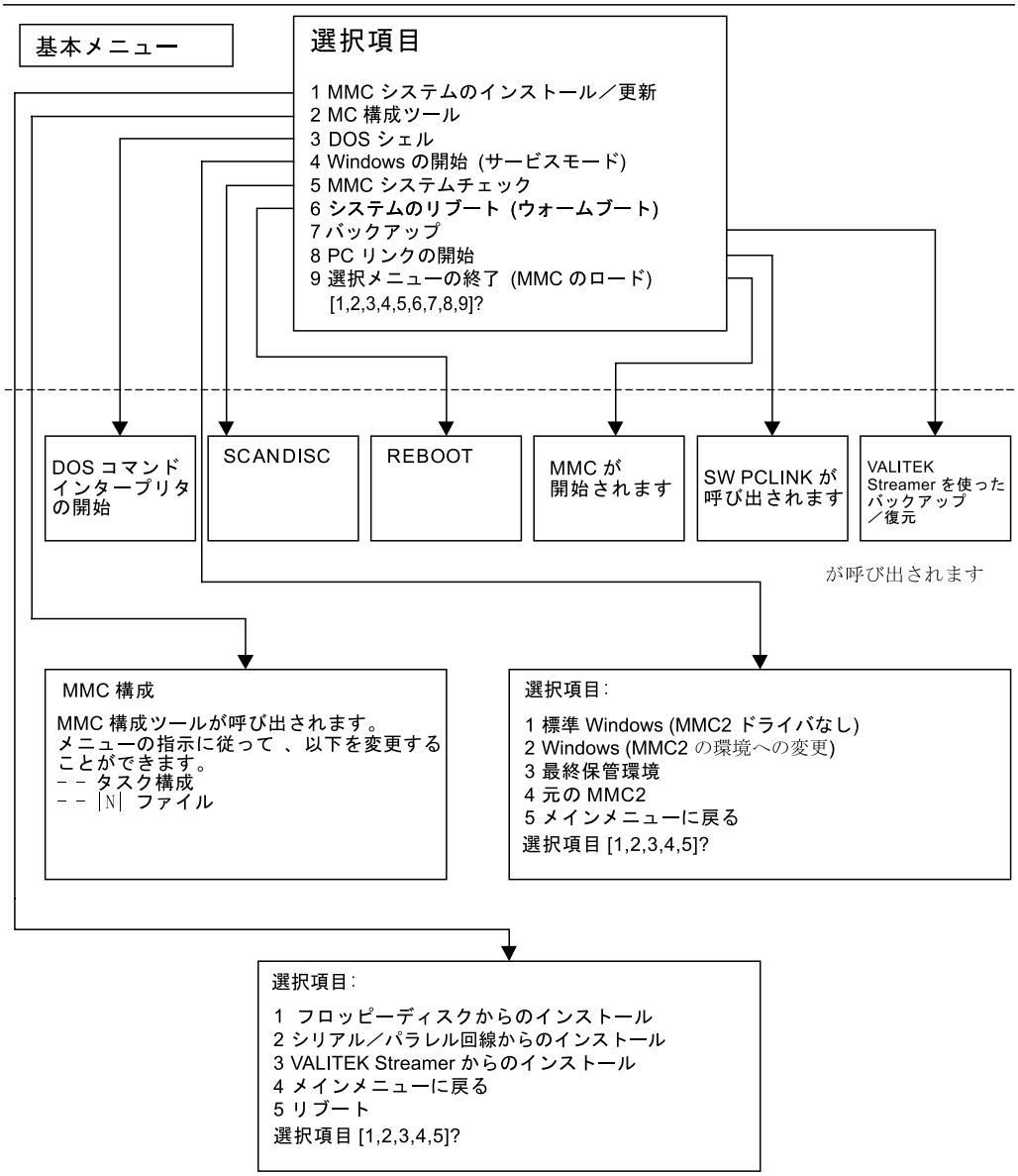
Regie.ini, Section [TaskConfiguration
TaskX=name:=oemframe, ..., TimeOut:=30000
OEMFrame.ini, Section [Application name
nSecondsToFindWindow = 30

6.7 840DI への OEM アプリケーションの組み込み

この章では，840DI コントロールに OEM アプリケーションを組み込む方法について説明します。

ソフトウェアのインストール用に，システムの起動時に活動化できるいくつかのメニューがあります。840DI の始動時に［Windows 95］メッセージが表示されたら，キー 6 を押します。

メニューの概観：



メインメニューの活動化：

コントロールの電源をオンにします。Windows 95 起動メッセージが表示されたら，キー 6 を押します。以下の表では，メニューツリーに用意されている機能をさらに詳

しく説明しています。

840DI のシステム機能の活動化		
メインメニュー		**機能**
1		MMC システムのインストールと補足または更新
2		MMC システムの構成（メニューガイド）
3		DOS コマンドインタープリタの呼び出し
4		サービスモードでの Windows の開始
5		ファイルシステムの整合性のテスト（整合性を復元するには，［スキャンディスク］を使用）
6		システムのリブート（ウォームリブート）
7		Valitek Streamer を使ったバックアップ／復元
8		PC リンクの始動（SW を CD-ROM ／ネットからインストールする場合）
9		終了と MMC の始動
4	1	PC 用の標準 Windows（MMC ドライバなし, 840DI 環境は未変更のまま）
	2	MMC 用の Windows（INI ファイル／ハードウェア構成の変更, MMC ドライバのロード）
	3	MMC システムは，最後に保管した環境で活動化されます
	4	MMC システムは，元の環境（購入時の環境）で活動化されます
	5	メインメニューに戻ります

ユーザデータ：

SW 4.3 以後，SW を更新してもユーザデータは変更されなくなりました。

■ フロッピードライブからのインストール：

以下のステップを行います。

1. コントロールの電源をオンにします。

2.MMC の始動時に，Windows 95 起動メッセージが表示されたら，操作パネルのキー 6 を押します。次のメニューが表示されます。

```
選択項目
1 MMC システムのインストール／更新
2 MMC 構成ツール
3 DOS シェル
4 Windows の開始（サービスモード）
5 MMC システムチェック
6 システムのリブート（ウォームブート）
7 Streamer
8 PC リンクの開始
9 選択の終了（MMC のロード）[1,2,3,4,5,6,7,8,9]?
```

3. キー 4 を押します。

システムからパスワードを尋ねるプロンプトが出されます。

4.

```
パスワード :.
```

0 ～ 2 レベルのパスワードを入力します。

– システム

– メーカ

– サービス

次のシステムが表示されます。:

```
選択項目 :
1  標準 Windows（MMC2 ドライバなし）
2 Windows (MMC2 の環境への変更 )
3 最新保管環境の MMC2
4 元の MMC2
5 メインメニューに戻る
```

1. キー 2 を押します。

2.Windows が開始します。2. ¥oem¥<application> ディレクトリを作成します。ここで <application> は，ご使用の OEM アプリケーションの名前（最大 8 文字）でなければなりません。

3. ディスクドライブにディスクを挿入します。次の OEM アプリケーション用のファイルがディスク上になければなりません。

-<application>.exe

-<application>.ini

- <application>.mdi

-<application>.zus

-<application abbreviation>_<language abbreviation >.dll

-< application abbreviation >_< language abbreviation 2nd language >.dll

4. 対応するディレクトリにファイルをコピーします。

-<application>.exe → ¥oem¥<application>.exe

-<application>.ini → ¥oem¥<application>.ini

-<application>.mdi → ¥oem¥<application>.¥<application>.mdi

-<application>.zus →
¥oem¥<application>.¥<application>.zus

-<application abbreviation>_<language abbreviation>.dll

→ ¥oem¥Language¥<application abbreviation>_<language abbreviation>.dll

-<application abbreviation>_<language abbreviation 2ndlanguage>.dll

→ ¥oem¥Language¥<application

abbrevia tion>_<language abbreviation2ndlan guage>.dll

5. \oem\regie.ini ファイルを使って，OEM アプリケーションを MMC に組み込みます。そのためには OEM ディレクトリにある Regi.ini ファイルを使わなければなりません。ここでは MMC2 ディレクトリにあるもとのファイル修正だけを行ってください。このファイルにはたとえば次のようなエントリがあります。[TaskConfiguration]

Task<free task number>=name:=<application>, Time-out:=20000

OEM の文書も参照してください。OEM ディレクトリ内に Regie.ini ファイルがない場合，この名前で新しいファイルを作成します。こうすれば，ソフトウェアの更新時に必ずデータが保存されます。たとえばソフトキーテキストの変更などの言語別の変更は，\oem\Language

6. ディスクドライブからディスクを取り出します。

7. Windows を終了します。次のメッセージが表示されます。

Save Windows Environment for next MMC start [Y,N]?（次の MMC の起動に備えて Windows 環境を保存しますか）

8. Windows 独自の設定を変更しなかったので，キー N を押します。次のメニューに戻ります。

選択項目 :
1　標準 Windows（MMC2 ドライバなし）
2 Windows (MMC2 の環境への変更)
3 最新保管環境の MMC2
4 元の MMC2
5 メインメニューに戻る
選択項目 [1,2,3,4,5]?

9. 選択項目 5 を選択します。これにより，メインメニューに戻ります。

選択項目
1 MMC システムのインストール／更新
2 MMC 構成ツール

3 DOS シェル
4 Windows の開始（サービスモード）
5 MMC システムチェック
6 システムのリブート（ウォームブート）
7 Valitek Streamer を使ったバックアップ／復元
8 PC リンクの開始
9 終了（MMC のロード）
選択項目 [1,2,3,4,5,6,7,8,9]?

10. 選択項目 9 を選択します。MMC が再始動され, ご使用の OEM アプリケーションが組み込まれます。

■ PC/PG からのインストール

以下のステップを行います。

1. MMC へのシリアルまたはパラレルインタフェースを使って，PC/PG に接続します。そのためには，3 線または 7 線ゼロモデムケーブルまたは双方向パラレルケーブルを使います。
2. コントロールの電源をオンにします。
3. MMC の始動時（コントロールの電源を入れた後）に Windows 95 起動メッセージが表示されたら，すぐに操作キーボード上のキー 6 を押します。次のメニューが表示されます。

   選択項目
   1 MMC システムのインストール／更新
   2 MMC 構成ツール
   3 DOS シェル
   4 Windows の開始（サービスモード）
   5 MMC システムチェック
   6 システムのリブート（ウォームブート）
   7 Valitek Streamer を使ったバックアップ／復元
   8 PC リンクの開始
   9 選択の終了（MMC のロード）[1,2,3,4,5,6,7,8,9]?
4. 選択項目 4 を選択します。

システムからパスワードを尋ねるプロンプトが出されます。

パスワード:.

5. 0 ～ 2 レベルのパスワードを入力します。

   – システム

   _ メーカ

   _ サービス

次のメニューが表示されます。:

選択項目 :
1　標準 Windows（MMC2 ドライバなし）
2 Windows (MMC2 の環境への変更)
3 最新保管環境の MMC2

4 元の MMC2
5 メインメニューに戻る
選択項目 [1,2,3,4,5]?

1. キー 2 を押します。Windows が起動します。

2.config.sys ファイルに

DEVICE=<PATH>¥INTERLNK.EXE という行が入っているかどうかを調べます。

ここで <Path> は，Interlnk.exe ファイルのパスです。

3. この行が入っていれば，ステップ 9 に進んでください。

4. 欠落行を Config.sys ファイルに挿入します。

5. 次に Windows を終了します。次のメッセージが表示されます。

Save Windows Environment for next MMC start [Y,N]?（次の MMC の起動に備えて WINDOWS 環境を保存しますか）

6. キー N を押します。以下のメニューに戻ります。

選択項目 :
1 標準 Windows（MMC2 ドライバなし）
2 Windows (MMC2 の環境への変更)
3 最新保管環境の MMC2
4 元の MMC2
5 メインメニューに戻る
選択項目 [1,2,3,4,5]?

7. キー 5 を押します。メインメニューに戻ります。

選択項目
1 MMC システムのインストール／更新
2 MMC 構成ツール
3 DOS シェル
4 Windows の開始（サービスモード）
5 MMC システムチェック
6 システムのリブート（ウォームブート）
7 Valitek Streamer を使ったバックアップ／復元
8 PC リンクの開始
9 選択の終了（MMC のロード）[1,2,3,4,5,6,7,8,9]?

8 次にキー 6 を押します。システムが再始動します。Windows を再起動します。ステップ 3 ～ 6 を繰り返します。

9. PG を開始してから, Intersvr.exe ファイルを実行します。指定したファイルが PG 上にない場合，ディスクを使ってこのファイルをコントロールから PG にコピーします。

10. ¥oem¥<application> という名前を付けてディレクトリを MMC 上に作成します。ここで <application> は，OEM アプリケーションの名前です（最大 8 文字）。

11. MMC から PG にアクセスして, MMC 上の対応するディレクトリに PG から以下のファイルをコピーします。

-< application >.exe → ¥oem¥< applica-tion>.exe

-< application >.ini → ¥oem¥< application >.ini

-< application >.mdi → ¥oem¥< application >¥< application >.mdi

-< application >.zus → ¥oem¥< application >¥< application >.zus

-< application abbreviation >_< language abbreviation >.dll

_ ¥oem¥Language¥< applicationabbreviation >_< languageabbreviation >.dll

-< application abbreviation >_< language abbreviation 2.language>.dll

_ ¥oem¥Language¥< applicationabbreviation >_< languageabbreviation 2. language>.dll

12. \oem\regie.ini ファイルを使ってご使用の OEM アプリケーションをコントロールに組み込みます。これには Regie.ini ファイルを使用する必要があります。Regie.ini ファイルが OEM ディレクトリにない場合，この名前で新しいファイルを作成します。こうすればソフトウエアが更新されても，ソフトキーテキストの変更などの言語別の変更は，\oem\Language\re_< 言語省略 >.ini ファイルと \oem\Language\re_< 言語省略 2nd 言語 >.ini ファイルに保管されます。これらのファイルが存在しない場合は，作成してください。

13. PG を終了します。

14. 次に Windows を終了します。次のメッセージが表示されます。

Save Windows Environment for next MMC start [Y,N]?（次の MMC の起動に備えて WINDOWS 環境を保存しますか）

15. キー N を押します。以下のメニューに戻ります。

選択項目：
1　標準 Windows（MMC2 ドライバなし）
2 Windows (MMC2 の環境への変更)
3 最新保管環境の MMC2
4 元の MMC2
5 メインメニューに戻る Your Choice [1,2,3,4,5]?

16. キー 5 を選択します。これにより，メインメニューに戻ります。

選択項目
1 MMC システムのインストール／更新
2 MMC 構成ツール
3 DOS シェル
4 Windows の開始（サービスモード）
5 MMC システムチェック
6 システムのリブート（ウォームブート）
7 Valitek Streamer を使ったバックアップ／復元
8 PC リンクの開始
9 選択の終了（MMC のロード）[1,2,3,4,5,6,7,8,9]?

17. 次に，選択項目 9 を選択します。MMC が始動され，OEM アプリケーションが組み込まれます。

18. 接続ケーブルを PG と MMC から取り外します。

6.8 ヘルプサポートの追加

概説

MMC 103 上の NC アプリケーション用の Microsoft-WinHelp ファイルは, ソフトキーを使ってナビゲートします。このファイルは，Remote Help（以後，略して Rh と呼びます）を使って作成することができます。

置き換え Rh（Remote Help）を使って，Windows の標準アプリケーションの WinHelp を領域アプリケーションとして MMC システムに統合することができます。

```
Task3 = Name := RH, Timeout := 10000,
_ HeaderOnTop := False
```

REGIE.INI のエントリ

置き換えアプリケーション Rh は，[TaskConfigura-tion］セクションに入力します。

例 6-67 REGIE.INI ファイルのエントリ

```
[TaskConfiguration]
...
Task8 = Name := RH, Timeout := 30000 PreLoad := False
...
[Miscellaneous]
HelpTaskIndex = 8
```

SwitchToHelpTask を使って暗黙の領域選択が行われる予定の場合，このエントリが存在しなければなりません。SwitchToHelpTask は，以下の表記が単純化されたものです。

```
SwitchToTask(HelpTaskIndex)
```

注 **:** 理論上は，どのタスクでも **HelpTaskIndex** に割り当てることができます。［**Info**］キーは，一種の **HotKey** または **ShortKey** を表しますが，これにより，領域を直接（**Regie** ソフトキーバーを経由してアナログから領域キー［**Machine**］をたどらずに）選択することができます。

Rh の呼び出し

Rh を呼び出すには次の 2 とおりの方法があります。

- Regie を介した領域切り替え
- アプリケーションで Info キー（あるいは，標準キーボードの F12 キー）を押す。

最初の場合，ヘルプファイルの目次が表示されます。ヘルプ情報を読むには，選択対象をキーワードに移動してから，［follow cross reference］ソフトキーを押します。

2 番目の場合，2 つの変数（helpflag と helpcontext）が呼び出しの前に設定されているかどうかをアプリケーションで確かめる必要があります。

ソフトキー OK で RHelp を終了すると，Regie は呼び出し先アプリケーションに戻ります。

通信

通信は，クライアント／サーバアーキテクチャに従って実行されます。アプリケーション（クライアント）は，NC-DDE 変数の helpflag と helpcontext を使って，Rh（サーバ）に対してヘルプテキストを要求します。

変数

NC-DDE 初期化ファイル（NSK ファイル）内には，次のような変数がなければなりません。

```
NEW("helpcontext","c:¥vb¥vb.hlp|Index")
NEW("helpflag",,"off")
```

注：変数が宣言されていないと，**DDE** 接続を確立できません。その場合，デフォルトヘルプファイルとデフォルトコンテキストが表示されます。

```
helpflag:
```

Regie の始動時，helpflag 変数は「オフ」に設定されています（Rh の開始が早すぎないようにするため）。.

ヘルプ機能を呼び出す場合，事前に helpflag を「オン」にしてください。

```
helpcontext:
```

helpcontext 変数で，検索されるヘルプファイルと索引（パイプシンボル "|" で区切られます）が Rh に渡されます。

以下に文字列の例を示します。

```
c:¥vb¥vb.hlp|Hide
```

これは，Hide という用語に関するヘルプを，c:¥vb¥vb.hlp ファイルから読み取れることを示します。

Rh の使用

Rh を呼び出した後，ソフトキーを使って，表 6.35 を参考にヘルプテキストをナビゲートすることができます。

表 6.35 Rh を使用するためのソフトキー

ソフトキー	意味／アクション
［page down]	ヘルプテキストを 1 ページ下にスクロールします。
［page up］	ヘルプテキストを 1 ページ上にスクロールします。
［previous selection］	選択項目が 1 つ上に移動します。
［nextselection］	選択項目が 1 つ下に移動します。
［follow cross reference］	RHelp は WinHelp に，現在の選択項目を新規の索引項目として取り込んで，それを表示するように指示します。
［back］	RHelp は WinHelp に，直前に表示されていた索引項目に戻るように指示します。これは，ヘルプが要求された最初の索引を見つけ出すまで可能です。
［content］	RHelp は WinHelp に，目次を表示するように指示します。
［OK]	RHelp を終了します。

［notebook］	ヘルプエントリにその他の情報を追加します。

ノートブック

ノートブック機能を使ってユーザは，既存のヘルプファイルに自分独自のテキストを追加することができます。

入力 Rh でテキストフィールドが開かれるので，そこにユーザは，項目に関連する追加情報を入力することができます。

ヘルプテキストの指定先　［helpcontext] 変数

記録　［OK］キーを使用します。

キャンセル　［cancel］キーを使用します。

表示　ヘルプ機能の次回の呼び出し時

この情報は，lhelpcontext.txt ファイル内の mmc2

```
mmc2

    rh                          Remote Help のサブディレクトリファイルと
       Ihelpcontext.txt         追加情報の入ったノートブック
```

図 6.7 ヘルプテキストの追加情報の保管場所

6.8.1 アラームヘルプファイルの作成

まず，アラーム番号に等しいキーワードを使って，RTF ファイルと HPJ ファイルを作成します。次に，Microsoft ヘルプコンパイラ HCP.EXE でそれらのファイルを使って，ヘルプファイルを生成することができます。

既存の脚注をすべて使用できるとは限りません。次の脚注がサポートされています。

#　コンテキスト文字列。ヘルプトピック名。

$　表題。トピックの表題を定義します。これは，検索時にリストボックスに表示されます。

K　キーワード。これをトピックに割り当てることができます。

6.8.2 ユーザ固有のヘルプメカニズムの作成

標準装備のヘルプを固有のヘルプメカニズムに置き換えたい場合，まずシーケンス制御内の State_Changed 関数内の AL_HELPINFO イベント（info キーを押す）をトラップしてから，AL_SwitchToTaskForHelp を使ってそれを確認します。そうすると，ヘルプ情報を表示するジョブは，シーケンス制御からユーザに引き渡されます。これで，ユーザ固有独自のヘルプメカニズムを活動化できるようになります。

6.9 構成ツール

注 **:** ソフトウェアリリース **4.4** 以後，構成ツールは **OEM** パッケージには組み込まれなくなりました。

7 シーケンス制御

概説

ユーザインタフェースの構成は，Visual Basic Professional で実行されます。各アプリケーションのコンポーネント（ウィンドウ）は，サブウィンドウ（MDIchild またはアプリケーションモーダルウィンドウ）として，フレーム（MDIframe）内でオープンされます。

機能

シーケンス構造は，SIEMENS 標準アプリケーションと互換 OEM アプリケーションにフレームを提供します。これは次の機能を提供します。

- シーケンス制御の管理（状態で構成されるメニューツリー)
- ソフトキーの照会（垂直と水平）
- NC 特殊キーの照会
- ソフトキーテキストの表示
- 対話式プロンプト行の管理
- 言語サポート

7.1 紹介

要約

この章では，シーケンス制御の用語や機能について説明します。

状態

シーケンス制御を記述する中央エレメントが状態です。次のような特徴があります。

- 固有の状態番号
- 初期状態
- 水平および垂直ソフトキー用のテキスト索引。ソフトキーテキストは DLL にあります。
- アプリケーションの実行中にオープンする MDIchild のリスト
- 戻りストリング : は，特殊状態が State_Changed または State_Reached に達したときに，アプリケーションに渡されます。
- z-flag: 設定 z-flag: 設定の保管または終了

初期状態は，状態マトリックス（名前の拡張子が ZUS のファイル）の最初の行に保管されます。

構造図では，状態は，図 7.1 のようなシンボルによって表されます。

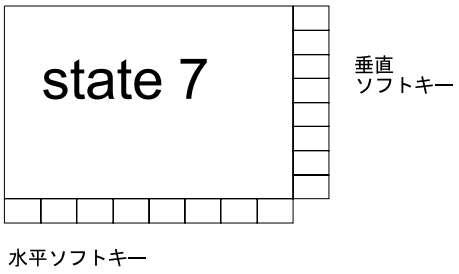


図 7.1 状態を表すシンボル

メニューツリー

これらのシンボルは，1 つのメニューツリーに結合することができます。図 7.2 に例を示します。

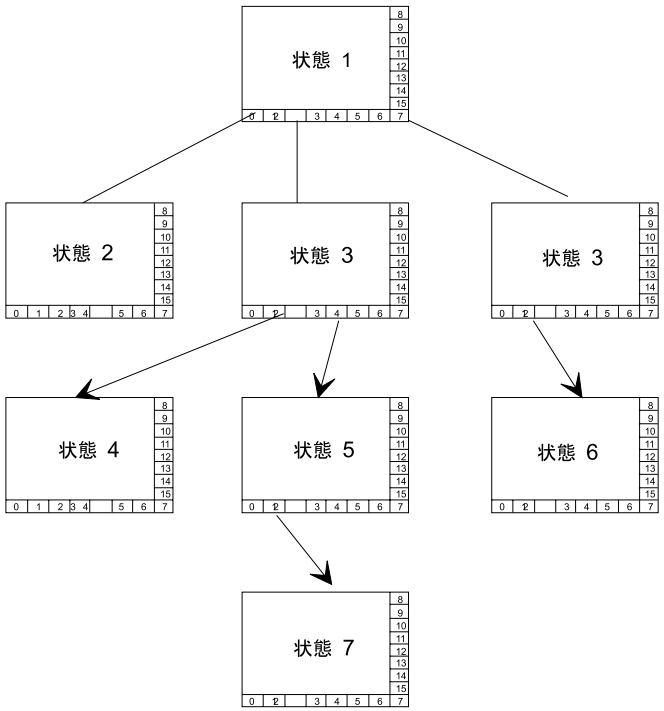


図 7.2 メニューツリーの例

アクション

アクションによって，状態変換が行われます。オペレータインタフェースでのアクションは，次のものによってトリガされます。

- 水平ソフトキーを押す
- 垂直ソフトキーを押す
- RECALL キーを押す
- マウスクリック
- 動作モードの変更

ソフトキー操作によるアクションは，状態マトリックス（拡張子が ZUS のファイル）にリストされています。

状態変換

状態変換の例は，次のとおりです。

- サブウィンドウが他のサブウィンドウを上書きする
- DDE 接続を，新しくオープンしたサブウィンドウの変数に対して構築する
- ソフトキーにラベルを付ける

ファイル

アプリケーション PROGNAME のシーケンス制御は，次の 6 つのファイルを使用します。

- REGIE.INI
- MMC.INI
- PROGNAME.INI（初期状態）
- PROGNAME.MDI（アプリケーションの構成要素，たとえばウィンドウなど）
- PROGNAME.ZUS（状態とアクションの説明）
- LANGUAGE.DLL

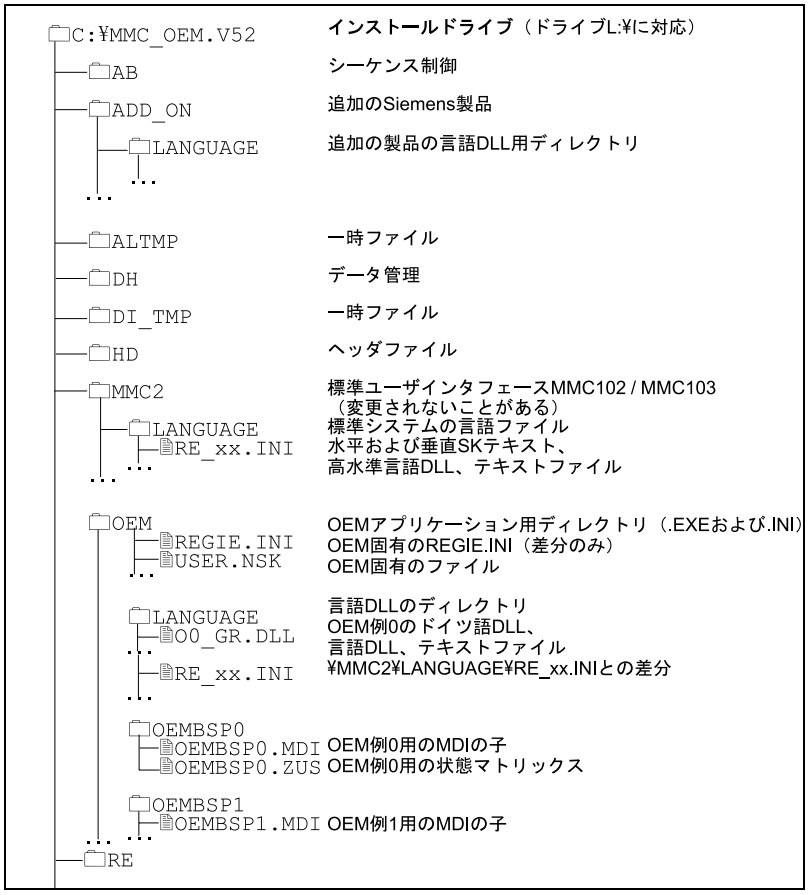
ファイル REGIE.INI および MMC.INI については，前のセクションで説明しています。各アプリケーションの PROGNAME には，アプリケーション固有の情報を含む初期設定ファイルがあります。ファイル名は，モジュール PRIVATE.BAS に定数として保管されます。

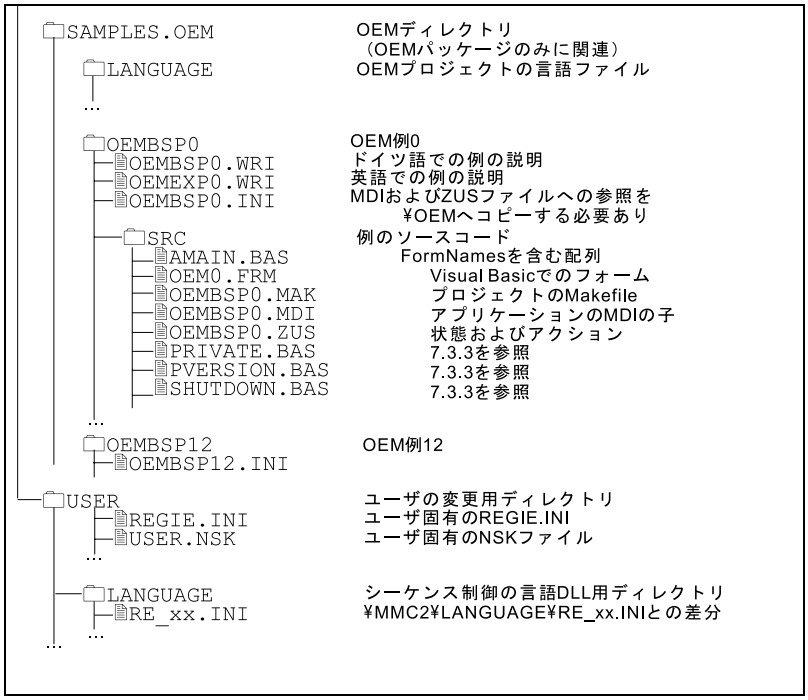
7.2 最初の OEM アプリケーション

要約

この章では，単純な OEM アプリケーションの作成を始めるために役立つ例を示します。@@これは OEM 例 0 として構築されます。

7.2.1 ディレクトリ構造





Siemens ユーザインタフェースは，ユーザがインタフェースを調整するとすぐに，ディレクトリ USER およびそのサブディレクトリにデフォルトと異なる設定を自動的に入力します。

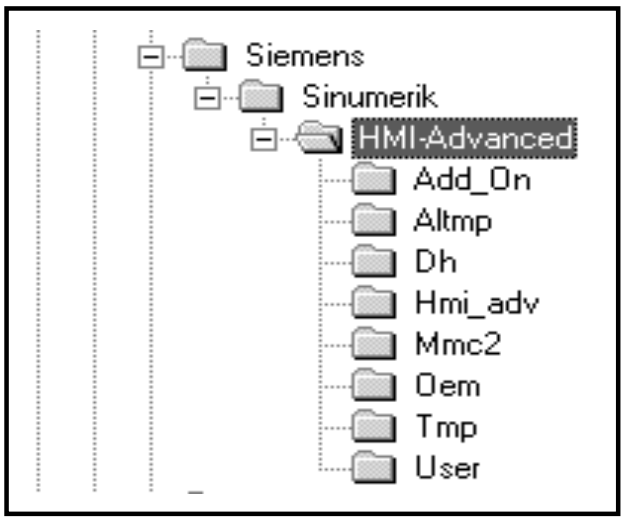
標準システム（ディレクトリ MMC2 にある）を参照するファイル MMC.INI, REGIE.INI および RE_xx.INI（ディレクトリ ADD_ON，OEM および USER にある）の OEM 拡張子は，Regie の起動中に，次の優先順位で収集されます。

MMC2 (固定システム設定)
ADD_ON
OEM
USER

注： ファイル MMC.INI，REGIE.INI および RE_xx.INI は，対応する先祖との差分ととられ，情報が収集されます。USER.NSK の場合，検出された最新のものが有効です。この情報は，完全に先祖と置き換わります。

7.2.2 SW 6 のディレクトリ構造

HMI 環境ソフトウェアのバージョン 6 のディレクトリ構造は次のとおりです :



新しい点 :

・HMI adv の中には以下のものが含まれます。

- ini ファイルを含むアプリケーション領域
- dll ファイル
- その他の MMC 固有データ

・Mmc2 では、次のものがインストールされています。

- サーバおよび COM オブジェクト（NCDDE、MBDDE、DH サーバ、AR サーバ）
- Regie、および HMI ベースからの対応する INI ファイルと NSK ファイル（REGIE、MMCctrls、Ivar、IMCFile…
- MMC.INI

これ以外のディレクトリは、すべて以前のバージョンと同じです。

7.2.3 フレームワークの構築

ステップ 1

パス L:¥SAMPLES.OEM の下にディレクトリ BSP_1 を作成して，ディレクトリ "L:¥SAMPLES.OEM ¥Oembsp0" の内容を，サブディレクトリを使って新しいディレクトリ "L:¥SAMPLES.OEM ¥BSP_1" にコピーします。

ステップ 2

次のファイルの名前を変更します。

L:¥SAMPLES.OEM¥BSP_1¥ OEMBSP0.INIL:¥SAMPLES.OEM

¥BSP_1¥BSP_1.INI

L:¥SAMPLES.OEM ¥BSP_01¥SRC

OEMBSP0.MAK L:¥SAMPLES.OEM¥BSP_1¥BSP_1.MAK
OEMBSP0.MDI L:¥SAMPLES.OEM¥BSP_1¥BSP_1.MDI
OEMBSP0.ZUS L:¥SAMPLES.OEM¥BSP_1¥BSP_1.ZUS
OEM0.FRM
L:¥SAMPLES.OEM¥BSP_1¥OEMFRM1.FRM

ステップ 3

次の言語ファイルをコピーします（DLL）。

L:¥OEM¥LANGUAGE¥

O0_GR.DLL BSP_1_GR.DLL
O0_UK.DLL BSP_1_GR.DLL

ステップ 4

Visual Basic を開始し，プロジェクトをオープンします。

L:¥SAMPLES.OEM ¥BSP_1¥BSP_1.MAK

次のエラーメッセージを OK で確認します。
"File not found 'OEM0.FRM'"
Remedy: Use the menu item FILE/ADD-File to add the file”OEMFRM1” to your project.

ステップ 5

フォーム ”OEMFRM1.FRM” をロードし, FORM の以下のプロパティを ”OEMFRM1” に変更します。

- 表題
- 名前
- タグ

注 : プロパティは大文字小文字を区別します。

ステップ 6

モジュール ”AMAIN.BAS” をロードし，このモジュールの以下の行を変更します。

Set g_frmFormName(0) = OEMFRM1

ステップ 7

モジュール ”PRIVATE.BAS” をロードし，このモジュールの以下の行を変更します。

Global Const TempFile = "BSP_1.4$$"

Global Const LOCAL_PROFILE = "BSP_1.INI"

Global Const APPL_PREFIX = "BSP_1"

ステップ 8

ファイル ”BSP_1.INI” を開き，以下の行を変更します。

MDIList=BSP_1¥BSP_1.MDI

ControlFile=BSP_1¥BSP_1.ZUS

このファイルをディレクトリ ”L:¥OEM” にもコピーします。

ステップ 9

ファイル ”BSP_1.MDI” を開き，以下の行を変更します。

OEM0 "1" を，OEMFRM1 "1" に変更

ステップ 10

ファイル ”BSP_1.ZUS” を開き，以下の行を変更します。

0 0 " OEMFRM1" "" "0" 0

ステップ 11

ディレクトリ L:¥OEM にサブディレクトリ ”BSP_1” を作成し，ファイル ”BSP_1.ZUS” および ”BSP_1.MDI” をそこにコピーします。

ステップ 12

Visual Basic の F5 をクリックしてアプリケーションを開始します。

エラー : "Must have startup form or Sub Main" が表示されます。

- これに応答します。
- Sub Main をダブルクリックします。
- MDIForm1 を選択します。

ステップ 13

Visual Basic のメニュー項目［EXE ファイルの作成］を使って実行可能ファイルを作成します。L:¥OEM を宛先ディレクトリとして選択します。

ステップ 14

以下の行を，ディレクトリ ”OEM” にあるファイル ”REGIE.INI” の［TaskConfigurationin］のセクションに入力します。

Task6 = name:=BSP_1, Timeout:=50000

ステップ 15

以下の行を，ディレクトリ ”OEM¥LANGUAGE” にあるファイル ”RE_GR.INI” および ”RE_UK.INI” に追加します。

HSK6="My example"

ステップ 16

ファイル Regie（REG_CMD.EXE）を開始します。

概説

結果のメニューツリーの概説 :

7.2.4 付加的なウィンドウ／フォームの追加

ステップ 1

Visual Basic を使って新しいフォームを追加します。プロパティ［Caption, Name and Tag］に，任意のストリング（たとえば OEMFRM2）を入力します。

ステップ 2

最初のフォームの以下の Visual Basic ルーチンをコピーします。

- Form_Activate
- Form_Deactivate
- Form_Load

ステップ 3

以下の行をファイル ”AMAIN.BAS” に挿入します。

Set g_frmFormName(1) = OEMFRM2

ステップ 4

以下の行をファイル ”L:¥OEM¥BSP_1¥BSP_1.ZUS” に挿入します。

0 0 " OEMFRM2" "" "0" 0
0 -1-1 "" "0" 0
1 -1-1 "" "1" –1
...
1-1 -1 "" "15" –1

ステップ 5

以下の行をファイル ”L:¥OEM¥BSP_1¥BSP_1.MDI に挿入します。
"OEMFRM2" 1

7.3 シーケンス制御のファイル

7.3.1 初期設定ファイル

各アプリケーションの PROGNAME には，アプリケーション固有の情報を含む初期設定ファイルがあります。ファイル名は，モジュール PRIVATE.BAS に定数として保管されます。

モジュール PRIVATE.BAS 内の，定数としてのアプリケーションの INI ファイルの定義は次のとおりです。

```
Global Const LOCAL_PROFILE = "PROGNAME.INI"
```

この INI ファイルには，次のセクション（表 7.1）も入っています。

表 7.1 ファイル PROGNAME.INI のセクション

セクション	意味
CONTROL	ソフトキーピクトグラム制御ファイル，ヘルプファイル，およびユーザピクチャのディレクトリ。例 : 状態マトリックス PROGNAME.ZUS
DEBUG	開発システム用の DEBUG サポートの設定。 例 : フルスクリーン

■ 例 7-1 OEM アプリケーションの INI ファイル

```
[CONTROL]
MDIList = <progname>.mdi
ControlFile = <progname>.zus ;directories for softkey pictograms,
control files, help files
SKPICTO=..¥skpicto¥ ; directory for user pictograms
HLP_DIR=..¥hlp¥
;Initial state: If there is no entry present, then the value ofINIT_STATE (from
PRIVATE.BAS) is assumed as initial state.
InitState=0
; if the NCCDE-Server is not started (for tests)
NO_NCDDE=1
[DEBUG]
; MDISize 0 = Debug ( for PC ), 1 = Full screen ( for control )
; When starting via the Regie ‘Full Screen’ is set automatically.
MDISize=0
; info-, RECALL-, Stop-Button: 0 = button invisible,
; 1 = button visible
i_Button=1
Stop_Button=1.
```

7.3.2 シーケンス制御のモジュールとフォーム

シーケンス制御のファイルは，ディレクトリ ..¥AB にあります。これらのディレクトリは，表 7.2 にリストされています。

表 7.2 シーケンス制御のファイル

ファイル	内容

ALCOMMON.BAS	汎用のファンクションとプロシージャの集合
ALDECL.BAS	グローバル変数とデータ構造を含む
ALDEFINE.BAS	MDI の子のスクリーン位置などの汎用定数を含む
ALDIALOG.FRM	ダイアログ行のフォーム：この行は，現在テスト出力用にシーケンス制御により使用中
ALENVIR.BAS	検索パスと環境変数を判別するファンクションとプロシージャを含む
ALHELP.FRM	ヘルプテキストの表示のフォーム
ALHSOFT.FRM	水平ソフトキーの処理用プログラムコード
ALLAFCT.BAS	言語切り替え用のプログラム：フォームのロードとアンロードを管理する
ALMODAL.FRM	アプリケーションモーダル出力用標準フォーム
ALPRINFO.BAS	テストプログラム，たとえばウィンドウタイプの出力
ALSTART.FRM	開始フォーム（MDIframe）ALSTART.FRM には，DDE 接続を介して使用できる，SetState と呼ばれるラベルフィールドも含まれている。このテキストフィールドを変更すると，メソッド SetState_Change がプロシージャ Set_State を呼び出す。
ALVSOFT.FRM	垂直ソフトキーの処理用プログラムコード
ALWTRACE.BAS	トレース機能用のプロシージャを含む（内部ファンクションのみ）
AL_UTIL.BAS	内部関数およびシーケンス制御のプロシージャを含むライブラリ

完全なシーケンス制御の使用のみがサポートされています。いずれかの部分をスキップした場合，その結果に対処するのはユーザー自身の責任です。

重要事項：
これらのファイルに変更を加えることは許可されていません。

7.3.3 アプリケーションのモジュール

アプリケーション固有のフォームに加えて，アプリケーションは次のファイルを提供する必要があります（表 7.3）。

表 7.3 アプリケーションが追加するファイル

ファイル	内容
AMAIN.BAS	アプリケーションに含まれるすべてのフォームのリストがある，プロシージャ FormNames を含む
PRIVATE.BAS	プロシージャ Pri-vate_Init_Def，Private_Init，State_Reached，関数 State_Changed，およびアプリケーション固有の宣言，プロシージャ，関数を含む
SHUTDOWN.BAS	アプリケーションの終了方法についての情報を含む。ここで呼び出される関数 QueryForShutDown については，6.5 章にある同じ名前のイベント QueryForShutDown の説明を参照。

各アプリケーションは，ファイル SHUTDOWN.BAS に追加される必要があります。このファイルは，例のアプリケーションの SRC ディレクトリからコピーできます。

このファイルが欠落している場合，コンパイラがエラーメッセージを出します。

7.3.4 シーケンス制御の一時ファイル

シーケンス制御は，一時ファイル用の特別なディレクトリ，たとえば ¥ALTMP などに保管されている一時ファイル（拡張子 .$$$，ソフトウェアリリース 4.x .4$$ から開始）を使用します。一時ディレクトリのデフォルト値 C:¥ALTMP は，初期設定ファイル MMC.INI の［Directories］セクションで設定されます。このディレクトリは，MMC 102 基本システムのインストール時に，自動的に作成されます。

注 : アプリケーション中の状態の配列を変更した場合，このファイルは削除してください。

グローバル定数

モジュール ALPRDECL.BAS は，アプリケーションでグローバル定数を定義するために提供されます。このモジュールには，私用定義および宣言のパターンが含まれます。これを使用する場合は，それぞれのアプリケーションディレクトリにコピーする必要があります。必要な定数は，REM 識別を削除するだけで選択されます。

注 : グローバル定数は，個別のファイルに保管してください。

7.4 アプリケーションの言語サポート

概説

アプリケーションは，プログラムコードのような言語に依存しない部分と，ダイアログテキストなど言語に依存した部分から構成されます。

ユーザインタフェースを言語に依存しない状態にしておくには，以下のいくつかの構成要件を満たす必要があります。

- それぞれの外国語で，判読できるフォントのサイズを指定することは非常に重要です。（たとえば，漢字などのフォントの場合，通常 16 ピクセル以上必要です。）
- テキストは，プログラムコードから独立して，修正したり翻訳したりすることが可能でなければなりません。

このため，言語に依存するテキストは DLL にあります。

7.4.1 ユーザインタフェースおよび言語

外国語

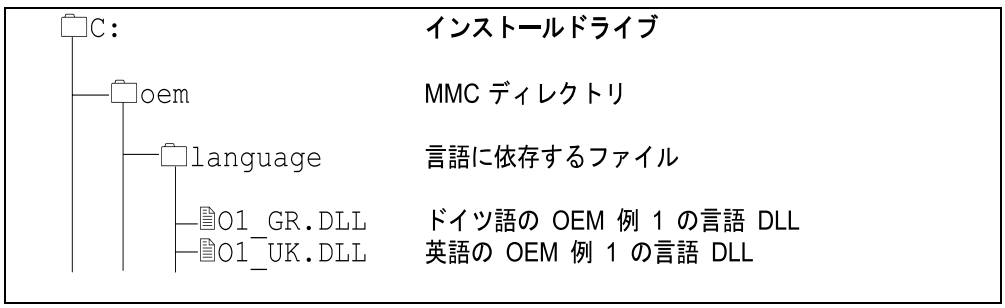
アプリケーションの言語に依存する部分は，アプリケーションが提供するライブラリ（ダイナミックリンクライブラリ）からロードします。これはたとえば，次のような場合に適用されます。

- すべての水平および垂直シフトキーのラベル付け
- ダイアログボックス，フォーム，および表示フィールドを持つテキストフィールド

編成

言語に依存するファイルは，ディレクトリ LANGUAGE に保管されます。例では，第 12 章で詳しく説明している，OEM 例 1 の言語 DLL の保管先 [言語 DLL の保管先を示しています。

■ 例 7-2 言語 DLL の保管先



言語 DLL の名前は，以下のように，REGIE のファイル LANGUAGE.INI の名前と同じように定義されます。

< プロジェクト略称 >_< 言語略称 >.dll

プロジェクト略称は 5 文字以内，言語略称は 2 文字以内でなければなりません。それ

ぞれを下線 "_" で区切る必要があります。

プロジェクト略称は，以下のようにモジュール PRIVATE.BAS で定義されます。

```
Global Const APPL_PREFIX = "O2"
```

指定可能な言語略称は，表 11-27 に記載しています。

7.4.2 テキストの RC ファイル

まず最初に，テキストは RC ファイル（リソースコンパイラ）に入力され，それから Visual C++ 環境で，構成の開始時に使用可能でなければならない DLL に変換されます。

テキストエントリ

ダイアログボックスおよびフォームが構成されると，画面に表示されるテキストラベルを，固有のシーケンステキスト番号（テキスト ID）を使って RC ファイルに入力することが必要です。

表示するラベルを Visual Basic で配列として保管し，それらを現行のテキストで初期化し，これらのテキストを順番に RC ファイルに入力することをお勧めします。

テキスト領域

RC ファイルの最初のエントリは，各テキストエリアの開始位置用に予約されています（表 7.4）。

表 7.4 テキスト識別のリスト

テキスト ID	意味
0	水平ソフトキーバーのテキスト用開始位置
1	垂直ソフトキーバーのテキスト用開始位置
2	ダイアログボックスとフォームのアプリケーション固有のテキスト用開始位置
3 ff	ユーザの処理

テキスト領域の終わり

水平および垂直ソフトキーバー（したがってテキスト用）のテキスト索引では，水平，垂直，およびアプリケーションに依存するテキストのそれぞれの領域にギャップを含めることはできません。(NULL ストリングは，これらのテキストのロード時の打ち切り基準であることに注意してください。)

アプリケーションに依存するテキストの索引には，ギャップを入れることができます。

RC ファイル中のコメント

既存の RC ファイルを外国語に翻訳する社外の翻訳者をサポートするには，以下の 2 つの情報をコメントとして提供するとよいでしょう。

- 最大テキスト長。たとえば //50 は 50 文字を意味します。
- 適切な省略形を知るための，テキストの意味を説明した注釈

さらに，分かりやすい大規模 RC ファイルを構成するためにもこれらのコメントを使用できます。

例として，アプリケーションマシン用の RC ファイルからの短縮した抜粋を示します。この全体は，MA¥LANGUAGE¥MA_.RC にあります。

■ 例 7-3 アプリケーションマシン用の RC ファイルからの抜粋（短縮）

```
STRINGTABLE
BEGIN
0 "10" // start index for horizontal softkey-text
1 "1000" // start index for vertical softkey-text
2 "2000" // start index for text, messages, errors ...
END
//------------------- horizontal softkeys --------------------------
STRINGTABLE
BEGIN
10 " "
11 "Work- piece" // two line texts
12 "Global SP" // complete sequence of the text indices
13 "Macros"
END
//--------------------- vertical softkeys --------------------------
STRINGTABLE
BEGIN
1000 " "
1001 "show G codes"
1002 "show auxiliary fncts"
1003 "spindles"
END
//----------------- regular text, messages, errors -------------------
// --- organized in several tables for better overview ---
// global.bas
STRINGTABLE
BEGIN
2000 "MKS" // 10
2001 "WKS" // 10
2005 "communication to NC broken down" //50
END
// BigIst.frm
STRINGTABLE
BEGIN
2101 "MKS" // 8
2102 "Position" // 14
END
```

開始フォームおよび DLL

開始フォーム（ALSTART.FRM，シーケンス制御の構成要素）はこの DLL をオープンし，最初の 3 つの開始値を，変数 g_nBegin_Hsoft，g_nBegin_Vsoft および g_nBegin_App にコピーします。

ロードメソッドの処理

各 MDI の子のロードメソッドで，コードセクションは，ユーザが提供する必要があります。これは，必要であればループで，従属 DLL からテキストを領域にロードし，

内部テキスト配列にコピーするものです。フラグは，ロードメソッドが呼び出されるたびにテキストが再ロードされるのを防ぎます。

ソフトキーテキスト

水平および垂直ソフトキーバーのそれぞれのテキストは，ロードメソッドによってそれぞれ ALHSOFT.FRM および ALVSOFT:FRM から読み取られます。エリアの読み取りは，新しいエリアに達するか，またはテキスト ID 中のギャップが検出されるまで続行されます。たとえば，水平ソフトキーテキストの読み取りは，テキスト ID にギャップがない場合，垂直テキストの索引の開始まで続行されます。

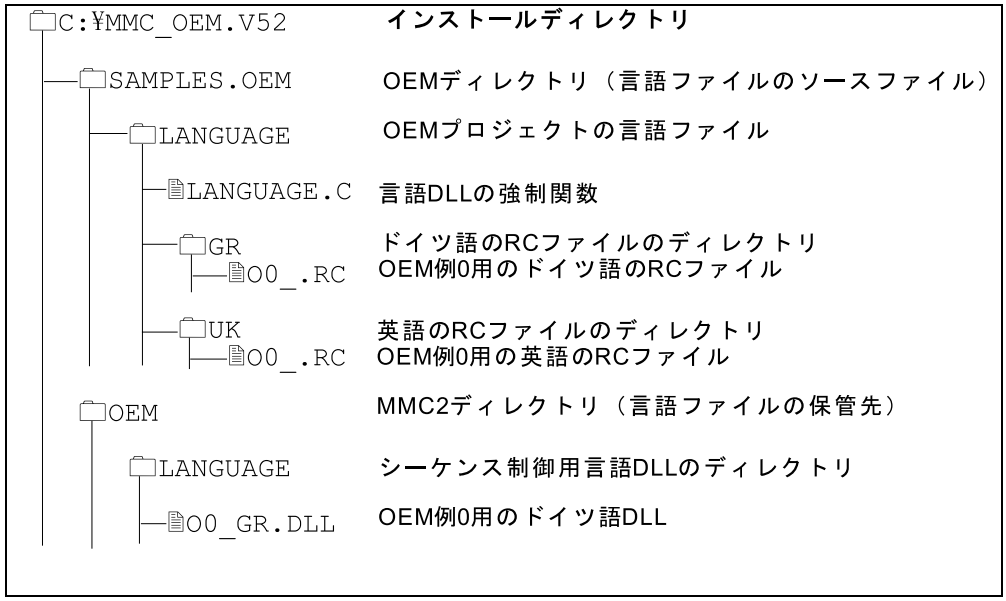
7.4.3 言語 DLL の作成

シーケンス制御のテキスト用の言語 DLL は，Visual C++ のアプリケーションスタジオを使用して，対応する RC ファイルから作成されます。

したがって，アプリケーションのソースファイルに加え，ファイル PROGNAME.RC および LANGUAGE.C も提供されます。アプリケーション OEM 例 0 の場合，これらのファイルは O0_.RC および LANGUAGE.C です。

シーケンス制御および完全な DLL のテキスト（OEM 例 0 の抜粋）を保管する RC ファイルの位置は，次の例で示しています。

■ 例 7-4 RC ファイルの保管先



言語 DLL を英語で作成する

- ファイル LANGUAGE.C および O0_.RC を作業ディレクトリにコピーします。
- Visual C++ を起動します。
- [CLOSE］をクリックしてすべてのファイルをクローズします（何らかのファイルがオープンしている場合）。

- ［PROJECT］をクリックします。
- ［NEW］をクリックします。
- ROJECT NAME O0_GR と入力します。
- ［PROJECT TYPE DLL］を選択します。
- ディレクトリを選択します（[BROWSE］と取り消しで選択)。
- ［OK］で終了します。
  ボックス［EDIT O0_GR.MAK］が表示されます。
- ［ALL FILES（*.*)］を選択します。
- 項目 LANGUAGE.C をクリックして，［ADD］でプロジェクトに追加します。
- 項目 O0_.RC をクリックして，［ADD］でプロジェクトに追加します。
- ［CLOSE］をクリックします。
- ［PROJECT］をクリックします。
- ［BUILD O0_GR.DLL］をクリックします。
- .DEF ファイルを作成するかどうかという質問に［YES］で応答します。
- ［BUILD］を再度クリックします。
  ## プログラムがコンパイルされ，正常に実行されればエラーや警告は戻されません。
- ［FILE］をクリックします。
- ［EXIT］をクリックします。附・¨したファイル O0_GR.DLL を，他の DLL も入っているディレクトリ OEM¥LANGUAGE にコピーします。

7.4.4 アジアの言語

アジアの言語の場合，対応するフォントを自動的にインストールする付加的なツールが必要です。

各言語パッケージで，このツールは自動的に対応するフォントをインストールし，mmc.ini およびレジストリで必要なすべての入力を実行します。

- DLL，テキストファイル（マシンデータファイル）および言語に依存する INI ファイルは，ディレクトリ "¥mmc2¥language" に保管されます。
- アラームテキストは，ディレクトリ "¥dh¥mb.dir" に保管されます。

7.4.5 言語の選択

ソフトキー 'select language' を使用して，制御を他の言語用に構成します。

参考文献 :
/IAD/ Installation and Start-Up Guide 840D,
13 章「MMC, Configure 'Language selection' softkey」

7.5 水平および垂直ソフトキー

概説

シーケンス制御では，水平および垂直ソフトキーを管理する機能とプロシージャが提供されます。これには，現在押しているソフトキーを個別に示すソフトキーのラベルを変更するプロシージャが含まれます。

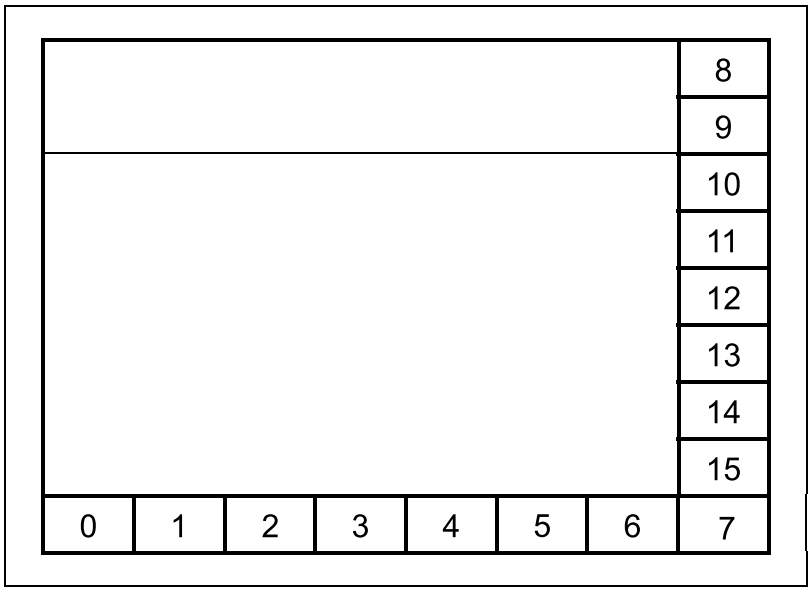


図 7.3 ソフトキーの配置

水平ソフトキーの範囲は 0 ～ 7，垂直ソフトキーの範囲は 8 ～ 15 で，リコールキーは 16 に割り当てられています。

注 : ソフトキーがアクションを解放できるようにするには，テキストエントリまたはピクトグラム名が言語 DLL に含まれている必要があります。DLL にスペースしか入っていない場合，このソフトキーは何も実行しません。

2 行のソフトキーテキスト

通常ソフトキーには，2 行から成る左詰のテキスト行を使ってラベルが付けられます。こうするには，結合部にブランクを 2 つ挿入します。

注 : 長いテキストの場合，VB はスペース文字が検出された場所に改行文字を自動的に挿入します。シーケンス制御では，ブランクが 2 つあると強制的に改行が挿入されます。

1 行のピクチャフォント

中国語などの，ピクチャフォントで書かれたテキストは，1 行だけで表示されます。

アクション

ソフトキーを押した後の応答は，索引 O ～ 15 の下にある状態表に記録する必要があります。この状態表は，アプリケーションごとに 1 つずつ存在し，ASCII 形式

（MMC システムディレクトリのファイルと，リンクされたアプリケーション）で保管されます。

リコールキー

シーケンス制御では，リコールキーはソフトキーと同じように扱われます。つまり，リコールキーを押した後のアクションは，索引 16 の下の状態表で判別することができます。

7.5.1 ソフトキーのアイコン

ソフトキーのピクトグラム

ソフトキーにラベルを付けるため，テキストの代わりにピクトグラムを使用することができます。ピクトグラム（拡張子 .BMP のビットマップファイル）は，Windows のドローイングツールであるペイントで作成できます。

ソフトキーで利用可能な描画領域のサイズ :

水平ソフトキーの場合は 74 x 36 ピクセル，垂直ソフトキーの場合は 79 x 34 ピクセル。

アイコンを作成するには，次の 3 ステップを実行します。

- ピクトグラムのディレクトリを作成します（アプリケーションの対応するディレクトリに置く必要があります)。

¥OEM¥...

- アプリケーション固有の .INI ファイルにパスを指定します。たとえば，次のようにします。

[CONTROL]
SKPICTO=OEMBSP5¥skpicto¥

- ソフトキーのテキストを，言語 DLL のピクトグラムの名前で置き換えます。小文字と大文字の違いに注意してください。さらに，ファイル名の拡張子を指定します。たとえば，次のようにします。

¥'

注 : アプリケーションに特殊なアイコンを割り当てる場合，このアイコンの名前はそのアプリケーションの名称にする必要があります（Visual Basic Makefile の場合は，アプリケーションの名称)。

ソフトキーのピクトグラムは，名前と拡張子の後に 2 つのバックスラッシュを付けて，RC ファイルに入れられます。
言語 DLL（¥'BMP）の名前であると解釈されます。これは，アプリケーション固有の INI ファイルからの基本パス SKPICTO によって完了します。このファイルは，
USER，OEM，ADD_ON，および MMC2 ディレクトリから検索することができます。見つかると，ディレクトリ内のビットマップが画面に表示されます（SW P4)。

P5 での新規事項 : 絶対パス（たとえば，L:¥...）を使ってビットマップを指定すると，このビットマップが表示されます（存在していれば)。重要 : DLL の「¥」には，先行する「¥」でタグ付けする必要があります（たとえば，L:¥...)。

RC ファイルのエントリ

OEM 例 0 の RC ファイルにソフトキーのピクトグラムを入れる方法については，例 7-5 を参照してください。

■ 例 7-5 RC ファイルのエントリ

```
STRINGTABLE
BEGIN
0 "10" // start index of the horizontal softkey texts
END
//------------------- horizontal softkeys ---------------------------
STRINGTABLE
BEGIN
10 ¥SKBILD1.BMP // bitmap file for softkey pictogram
11 "work- pieces" // two line text
12 "global SR" // sequence of text indices without gaps
13 "macros"
END
```

ソフトキーのピクトグラムの作成

最も簡単な方法は，対話プログラミングのソフトキーのピクトグラムのような既存のパターンから始めることです。それらは，パス C:¥MMC2¥DP¥SKPICTO にあるサンプルのインストールに入っています。
例 :DP0_1.BMP ファイル

7.5.2 ソフトキーバーの構成

ソフトキーバーに含めるテキストフィールドの数は，構成することができます。これは，アプリケーション固有 INI ファイルの［CONTROL］セクションに，次の行を追加することによって実行します。

PROGNAME.INI ファイルに 4 つの水平ソフトキーを構成し，垂直ソフトキーを構成しない場合は，以下のようにします。

ソフトキーの数

■ 例 7-6 ソフトキーの数

```
[CONTROL]
NbrHorSoftkeys = 4 ;number of horizontal softkeys = 4
NbrVerSoftkeys = 0 ;number of vertical softkeys = none.
```

これらのエントリのいずれかが欠落していると，それぞれに 8 つのソフトキーを指定した標準の設定になります。

0 を入力すると，ソフトキーバーは表示されません。

8 より大きい値を入力しても，その値は 8 に制限されます。

注 : パラメータを 0 に設定してソフトキーバーをオフにする場合，アプリケーションは未使用の画面区域に対応する必要があります。この構成は，シーケンス制御ではサポートされていません。

7.5.3 ソフトキーバーの拡張

ETC の追加

水平ソフトキーバーは，入力フィールド ETC1 および ETC2 を使用して拡張することができます。拡張するには，0¥8¥16 という ID を，Htext 列の .ZUS ファイルに記録される状態説明に追加する必要があります。この ID には，次のような意味があります。

表 7.5 に例を示します。

表 7.5 ETC の状態表の拡張（抜粋）

状態／アクション	アクセスレベル	Htext	Vtext	子	戻り	Z- フラグ／続く状態	注釈
999 「ETC によって拡張されたソフトキーバーがある状態 10 の説明」							
###10###		0¥8¥16	vt0	「図 1」	"abc"	1	「ETC によって拡張されたソフトキーバーがある状態 10」
0		-1	-1	"a1" ""	"SK0"	17	"Softkey0"

この拡張が適用されるのは，ソフトキーのテキストだけです。
ETC バーがアクティブであることをアプリケーションに示すため，グローバル変数 g_nEtcLevel は，次のいずれかの値で戻されます。

表 7.6 戻される ETC レベルの情報

g_nEtcLevel	意味
0	標準バーのソフトキーテキスト
1	拡張 ETC1 のソフトキーテキスト
1	拡張 ETC2 のソフトキーテキスト

7.6 ウィンドウのタイプ

MDIchild の説明

アプリケーションに属する MDIchild ごとに，以下の情報が PROGNAME.MDI ファイルに記録される必要があります。

- ASCII（引用符（"）でテキストを囲む形式）で記述した MDIchild の名前。
- MDIchild の数（0 ～ 5）
- 注釈（任意指定）

MDI ファイルの編成

アプリケーション「machine」にある MDI ファイルからの抜粋

■ 例 0-1 MDI ファイルの編成

```
"Preset" 1 // This is part of the basic screen Machine
"spin" 2
"feed" 1
"tool" 1
```

表 7.7 では，MDIchild のさまざまなタイプ（ピクチャ）を示しています。

表 7.7 MDIchild のタイプ

タイプ	アクション
0 NOUNLOAD	MDIchild は，常に表示されます。
1 UNLOAD	MDIchild は，オーバーレーの際に削除されます。
2 RESIZE	MDIchild のサイズは，オーバーレーの際に削減されます（サイズ 0)。
3 MODAL	MDIchild は，アプリケーションモーダルウィンドウです。
4 OVERLAYED	MDIchild はオーバーレーされません。部分的にオーバーレーされる場合があります。
5MAYBE_1OR2	MDIchild は，可能な限り長くメモリに保持されます。

ピクチャタイプ 0:NOUNLOAD

このピクチャは，常に画面に表示されます。このピクチャをアンロードする必要がある場合には，関数を使わなければなりません。タイプ「0」の MDIchild は，初期化の終了時に開きます。通常このタイプは，動的に使用されます。

ピクチャタイプ 1:UNLOAD

ピクチャはメモリから削除されます（そのデータセグメントも含む)。すべてのリソースが解放されます。とりわけ，このことは，入力したデータがすでに有効ではなく，ピクチャが再ロードされることを意味します。再ロードされるたびに，関数 Form_Load が呼び出されます。

ピクチャタイプ 2:RESIZE

ピクチャはメモリ内に残り，サイズは 0 に設定されます。ピクチャは依然としてロードされているため，関数 Form_Load は一度だけ呼び出されます。

ピクチャタイプ 3:

領域変更時に，モーダルウィンドウを参照するすべてのテキストやエントリはメモリから削除されるため，それらは消失します。デフォルトでは，シーケンス制御は，最初のテキストボックスの内容をグローバル変数に保管し，戻し時にはその内容をこのテキストボックスに入れます。

MODAL ピクチャタイプ 4:OVERLAYED

タイプ 1 のピクチャが含まれます。このピクチャは，他のウィンドウによってオーバーレーされる場合があります。たとえば，タイプ 1 のピクチャは，別のウィンドウ

によってタッチオーバーレーされるとアンロードされます。

他の（部分的に）隠れた MDIchild によって MDIchild が置換（オーバーレー）されないようにする場合には，この MDIchild にタイプ 4 を割り当てることができます（場合によっては，関数 Set_ChildType を使って一時的にこのようにします)。

ピクチャタイプ 5:MAYBE_1OR2

フォームタイプ 5 は，現行のところフォームタイプ 2 と同じように動作します。
このフォームタイプには，次のようなプロパティを割り当てることが検討されています。
リソースが少ない場合，フォームタイプ 5 はフォームタイプ 1 に変換され，その後メモリから消去することができます。

情報ファイル

アプリケーションの MDIchild の最も重要な仕様は，以下のとおりです。

- Child の名前
- そのタイプ
- および Child を開くとき削除する必要のある Child のリスト（ここに保管されているのは索引だけです)。

これらは，ALTMP ディレクトリにある .4$$ という拡張子の情報ファイルに保管されます。それには，次のようなプロパティがあります。

- このファイルの名前は，モジュール PRIVATE.BAS で定数として定義されます。たとえば，次のようにします。
  Global Const TempFile = "OEM1.4$$"
- 新しいソフトウェアリリースのインストール中に，4$$ という拡張子がついたファイルはすべて ALTMP ディレクトリから削除されます。

状態変更の速度を上げるため，ファイルは動的に作成されます。

情報ファイルの作成

情報ファイル *.4$$ を作成する場合，必要な情報を読み取るため，アプリケーションのすべてのフォームが一時的にロードされます。フォームの Move 命令を終了した後，Form_Load プロシージャを打ち切ると，このプロシージャの速度を上げることができます。
次のようにして，必要な指示を出します。

```
If AlMakeZwidat = True Then Exit Sub
```

Form_Unload または Resize の実行中（たとえば，DDE 接続を中断しているとき）に，特定のアクションを実行すると，上の行をその場所にも挿入する必要が生じる場合があります。

7.6.1 アプリケーションモーダルウィンドウ

アプリケーションモーダルウィンドウは，アラームやメッセージを表示するために役立ちます。このウィンドウには，次のような特徴があります。

- 各アプリケーションモーダルウィンドウには，対応するソフトキーバーで，

別々の状態を割り当てることができます。

- アプリケーションは，プロシージャ Mo-dalDialog または UsrModalDialog を呼び出すと，これらのウィンドウを開きます。
- プロシージャ ModalDlgEnd -1 を呼び出すと，アプリケーションモーダルウィンドウは中断されます。
- 同時に開くことができるアプリケーションモーダルウィンドウは 1 つだけです。

アプリケーションモーダルウィンドウはタイプ 3 で，「常に手前に表示」されます。これはつまり，これらのウィンドウが他のアプリケーションウィンドウより前面にあり，関数を呼び出す場合だけクローズできるということです。こうしたアプリケーションモーダルウィンドウは MDIchild にしてはなりませんが（プロパティ MDIchild = False），MDIchild のようなシーケンス制御によって処理されます（プロパティ「常に手前に表示」は，Windows 3.1 および Windows 95 では MDIchild に適用されません）。

ウィンドウの位置決め

これらのウィンドウの位置を指定するには，定数のオフセットに値 top を必ず追加する必要があります。たとえば，次のようにします。

```
Move BeArtleft, BeArttop + MDIstart , 3000, 1500
```

この定数には，MDI フレームの開始位置とアプリケーションの出力領域の開始位置との差が入ります。

注 : 次の理由で定数が追加されます（ユニット : twips）。モーダルウィンドウが MDIchild ではない。そのため，元のウィンドウが画面の左上隅に配置されている。正しい位置にするためには，MDI フレームの元の座標を追加する必要があります。

標準形式

シーケンス制御と一緒に，データファイル ALMODAL.frm のフォーム ApplModal が渡され，これにより標準のアプリケーションモーダルウィンドウが開きます。プロシージャ WriteModalDlg を使用すると，このウィンドウはさまざまなテキストで何回も使用することができます。

7.6.2 メッセージ／注意事項の表示

シーケンス制御には，メッセージを表示するためのプロシージャ Write_Dialog が備えられています。これは，パラメータテキストで指定したテキストを，固有の形式ではない Siemens ユーザインタフェースの標準のダイアログ行にあるアプリケーションに対して書き込みます。

7.7 メニュー制御

7.7.1 状態表

状態表

アプリケーションの状態およびアクションはすべて，状態表として PROGNAME.ZUS ファイルに記録されます。

表 7.8 状態表の構造

状態／アクション	アクセスレベル	Htext	Vtext	子	戻り	Z- フラグ／続く状態	注釈
999 「状態 10 の記述」							
###10###		ht0	vt0	"x1" "x2" ... ""	"abc"	1	"state 10"
0	John	-1	-1	"a1" ""	"SK0"	17	"Softkey0"
14	3	-1	-1	""	"SK14"	-1	"abort"
15	Mary	-1	-1	""	"OK"	3	"OK"
16						3	”リコールキー”
999 「状態 17 の説明」							
###17###		ht0	vt0	"a" "b" ""	"0017"	0	"state17"
0	4	-1	-1	""	"SK0"	0	"Softkey0"

状態

状態変更の後や，アクションが実行された場合は，現在の状態（状態番号）と，この状態を引き起こしたアクション（ASCII ストリング）がアプリケーションに示されます。これは，関数 State_Changed（func，status，CurrState）を呼び出すことによって行われます。この関数は，アプリケーションによってモジュール PRIVATE.BAS に提供されます。

注 : 状態表の文字列（注釈と戻り値）には，円記号（"￥"）を含めることはできません。

状態／アクション

この状態に関連付けられたアクションに，個別に状態を指定します。

- 状態説明は，必ず固有の状態番号で始まります。この番号は大括弧に囲まれており，この状態の初期値が続いています。
- これらの初期値は，この状態になると開かれるシナリオを定義しています。
- これらの値は，この状態が残されると要求に応じて更新されます（Z -Flag=1，以下を参照）。
- 状態は昇順でファイルに記録されます（SW 2.2 が不要なため）。状態のアクションは，水平（0 ... 7）および垂直（8 ... 15）ソフトキーに対応して，0 ～ 15 の番号が付けられます。リコールキーに付けられる番号は 16 です。
- アクションは，特定のソフトキーを押したときに実行される処理を記述します。状態に対して定義されているソフトキーの番号がない場合，ソフトキーを押しても何のアクションも実行されません。

• 注釈行は，アクション番号 999 で見分けられます。

ソフトキーのアクセス許可のレベル

各ソフトキーには，アクセスレベル（AccessLevel）を割り当てることができます。状態マトリックス PROGNAME.ZUS には，新しい属性 AccessLevel が追加されています。
この属性を使用すると，このソフトキーを操作するのに必要なアクセス権またはキースイッチを設定することができます。
ソフトキーバーをロードすると，現行のキースイッチの位置は，アクションが許可されているかどうかを確認するため，マトリックスで指定した値と比較されます。これが当てはまらない場合，ソフトキーテキストはそれ以上表示されません。

無効なエントリ（たとえば，Otto という名前はアクセスレベルが割り当てられずに指定される）のデフォルト値は，AccessLevel := 4 です。

エントリが欠落しており，AccessLevel = 7 の場合，そのソフトキーにはアクセス許可レベルが割り当てられていないため，そのソフトキーは常に有効です。

表 7.9 は，可能なアクセス許可レベルの概要を示しています。グレーの背景色でマークされているのは，デフォルトの設定です。

表 7.9 8 レベルのアクセス許可

アクセスレベル：	必要な許可：	ユーザグループ
-1	常に必要	すべてのユーザ
0	システムパスワード	当社
1	MTB パスワード	工作機械メーカー
2	サービスパスワード	設定／サービススタッフ（工作機械メーカー）
3	ユーザパスワード	特権ユーザ（企業内サービス）
4	キースイッチ位置 3	プログラマ
5	キースイッチ位置 2	上級オペレータ
6	キースイッチ位置 1	オペレータ
7	キースイッチ位置 0	中級オペレータ（NC 開始／ NC 停止，操作パネル）

記号ユーザクラス

記号ユーザクラスを定義できるようにするため，単なる番号の代わりに，記号名を状態マトリックスに入力することもできます。そうしたエントリに割り当てられる値は，アプリケーション固有の INI ファイルから取られます。ここで，このファイルのセクション［ACCESSLEVEL］は，対応するエントリを含めて作成する必要があります。

記号ユーザクラスの例

アプリケーション PROGNAME では，INI ファイル PROGNAME.INI の［ACCESSLEVEL］セクションに，次のエントリが入っています。

■ **例 7-8 記号ユーザクラスのエントリ**

```
[ACCESSLEVEL]
```

```
John=2 // access for MTB service employee John
George=3 // access for user George
Mary=4 // access for NC programmer Mary
```

状態マトリックスでアクセス許可レベルを定義するには，以下の 3 とおりの方法があります。

(1) アクセス許可を指定しない。つまり，以前のリリースと同じです。
(2) 番号でアクセス許可を指定する。
(3) 記号値でアクセス許可を指定する。

この 3 つの方法については，表 7.8 に記載しています。

Htext / Vtext

状態またはそれぞれのアクションに応じた水平および垂直ソフトキーのテキスト索引。

- ソフトキーラベルは，ダイナミックリンクライブラリ（DLL）に保管されます。
- それぞれのテキスト索引は，ソフトキーバーの最初の要素を指します。0 より小さい設定値（たとえば，-1）は，対応するソフトキーバーの値を変更する必要がないことを示します。
- 状態説明にある水平および垂直ソフトキーバーのテキスト索引は，DLL の最初のテキストブロックと比較して設定する必要があります。

DLL にある垂直ソフトキーバーのテキストが位置 64 から始まる場合，これは状態説明の垂直ソフトキー索引 0 に対応します。2 番目のテキストブロックが位置 72 から始まる場合，状態説明の索引 8 に対応します。このようにして順次続いていきます。

Child

アプリケーションのアクションに対して個々の状態に応じて開かれる MDIchild のリスト。これには，次のような特性があります。

- MDIchild の最大数は，ALDECL.BAS というモジュールにある定数 MaxFormNbr によって判別されます。
- 表示される MDIchild が少ない場合，リストの最後の要素は空ストリングです。
- 標準形式は，アプリケーションが起動すると自動的に開かれるため，リストに含める必要はありません。

標準形式とは，たとえば，ソフトキーバーのフォーム，ダイアログフィールド，状態の行などです。

戻り

状態に達するか，または以下を含むアクションが起動されると，アプリケーションに戻される ASCII テキスト。

- プロシージャの名前
- 状態番号
- フォームの名前

Z -Flag ／連続する状態

エントリには，それが状態説明またはアクション説明のどちらにあるかで意味が異なります。

状態説明 :
Z-Flag は，状態の終了時に，現行の設定（ソフトキーのテキストおよび開いた MDIchild のリスト）を保管する必要があるか（ある場合は Z-Flag=1），またはその必要がないか（ない場合は Z-Flag=0）を示します。

処理説明 :
続く状態は，そのアクションの結果として（そのアクションの最後に）生じる状態を指定します。状態がそれに続く状態に変わると，対応するシナリオが開きます。
この設定値が 0 より小さい場合，指定したアクションだけが起動され，現行の状態は変更されません。

注釈

（任意指定）: 説明のための注釈 ; 長さは最大で 25 文字です。

表の構造

考えられる状態ごとに，表には，状態を説明するための 1 行と，この状態に対応する応答を説明するための行が最大で 17 行入っています。状態表はいつでも拡張できます。データを形式設定するため，スペース行と行内のタブを使用することができます。”¥”（円記号）文字は，ソフトキーバーの拡張と解釈されてしまうため，状態表で使用することはできません。

7.7.2 状態変換

状態変換を行うと，通常はサブウィンドウが他のサブウィンドウによって隠されます。つまり，新しいサブウィンドウがロードされ，以前のサブウィンドウはアンロードされます。

状態変換は，次のイベントにより引き起こされます。

- 状態マトリックスで構成された連続状態を割り当てられているソフトキーをクリックする。
- PLC 信号に対する応答として，状態制御の関数（たとえば，Set_State）を呼び出す。

ソフトキーを押す

次のアクションは，ソフトキーを押すか，状態変換が生じると，シーケンス制御で起動されます。

ソフトキーを押すと，プロシージャ
Set_Action (SoftkeyIndex) が呼び出され，それから次のことが起こります。

(1) 現在の状態に属する MDIchild が表示される。
(2) （関数 State_Changed が呼び出されて）メッセージがアプリケーションに送信される。
(3) 状態を変化させない場合，状態の順序付けが実行される。
(4) 関数 State_Reached。

関数 State_Changed および State_Reached は，ユーザによって提供され，これらの関数には，それぞれの状態変換に対するアクションへのアプリケーション特有の応答が入ります。

シーケンス制御は，状態マトリックスで指定した戻りストリングを，関数 State_Changed の引き数 func に渡し，引き数 status に状態番号を入力します。この関数は，0（状態を変更する）または 0 以外の値（状態を変更しない）を戻します。

引き数 CurrState には，現在の状態の番号が入ります。

特に終了時には，呼び出し State_Changed（-1, -1, -1）は，終了をアプリケーションに通知します。

その他の初期状態

アプリケーションを別の初期状態にするには，ユーザ関数 State_Changed で，モジュール PRIVATE.BAS に，Set_State と共に定義する必要があります。次の例が示すとおり，これを実行する方法はいくつかあります。

INI ファイルへの初期状態の入力

アプリケーション OEMBSP1 は状態 1 で開始します。

■ 例 7-9 INI ファイルへの初期状態の入力

```
[CONTROL]
MDIList=OEMBSP1¥OEMBSP1.MDI
ControlFile=OEMBSP1¥OEMBSP1.ZUS
InitState=1
```

初期状態のエントリが存在しない場合は，（PRIVATE.BAS 内で）定数 INIT_State に指定した値を初期状態として使用します。

```
Global Const INIT_State = 0
```

State_Changed での初期状態の変更

アプリケーションが，関数 State_Changed で，状態 10 で始まるようにします。抜粋のみを表示します。

■ 例 7-10 State_Changed の初期状態の変更

```
' Modifying the initial state
Function State_Changed (ByVal action As String, ByVal newstate As
_ Integer, ByVal oldstate As Integer) As Integer
If func = AL_ACTIVATE then
Set_State (10) 'number of the desired initial states
Endif
' Here the old state is displayed and then the desired initial state
' is immediately installed, when an application is selected.
End Function.
```

アプリケーションがシーケンス制御によってロードされると，関数 PRIVATE.DEF（定数の宣言）と PRIVATE.INI が実行されます（TerminateTask を使ってアプリケーションを終了していない場合は，1 度だけ実行されます）。

開始

アプリケーションが起動すると，ユーザが提供するプロシージャ Pri-vate_Init_Def が呼び出されます。その後，シーケンス制御は初期化され，最終的にユーザプロシージャ Private_Init が呼び出されます。

7.8 プロシージャと関数

この章には，アプリケーションで使用する関数とプロシージャの説明が記載されています。表 7.10 では，次の基準に従って並べられた概要を示しています。

- 状態制御
- 照会関数
- MDIchild 関数
- アンロック／ロックソフトキー
- ソフトキーテキスト関数
- テキスト表示
- モーダルウィンドウ関数
- アクション関数（状態マトリックスの動的変更）
- フォームとコントロールの属性

概説

表 7.10 関数とプロシージャの概要（F/P: F= 関数，P= プロシージャ）

名前	F/P	意味
状態制御		
Set_Previous_State	P	前の状態に戻します。
Set_State	P	状態が切り替わります。
State_Changed	F	アクションに続くアプリケーションの状態に戻します。
State_Reached	P	State_Changed のようなユーザプロシージャ。
照会関数		
AlGetAccessLevel	F	現行のアクセスレベルを読み取ります。
AlGetSkByAction	F	特定の値を戻す現行の状態のソフトキー番号を読み取ります。
Get_FormIdx2	F	フォーム名に指標を与えます。
Get_FormIndex	F	ASCII フォーム名に指標を与えます。
MDIchild 関数		
ChildActivate	P	シーケンス制御に MDIchild を宣言します。
ChildDeactivate	P	シーケンス制御から MDIchild を切断します。
Hide_Childs	P	オープンしているすべての MDIchild を隠します。
Set_ChildType	F	MDIchild のタイプを設定します。
Show_Focus		特定の MDIchild がエントリフォーカスを取得します。
AlSkipFocus	P	MDIchild をフォーカスの取得から除外します。
Hide_A_Child	P	指定した Child を隠します。
Show_Hidden_Childs	P	隠されたすべての Child を表示します。

名前	F/P	意味
アンロック／ロックソフトキー		
Lock_Softkey	P	ソフトキーまたはソフトキーバーをロックします。
Unlock_Softkey	P	ソフトキーまたはソフトキーバーのロックを解除します。
Al_StopSks	P	Al_ResumeSk が発生するまでソフトキーアクションをロックします。
Al_ResumeSks	P	Al_StopSksでロックされたソフトキーアクションのロックを解除します。
LockSkByState	P	特定の状態に帰着するすべての状態ソフトキーをロックします。
UnlockSkByState	P	特定の状態に帰着するすべての状態ソフトキーのロックを解除します。
LockSkByAction	P	特定の戻り値を戻すすべての状態ソフトキーをロックします。
UnlockSkByAction	P	特定の戻り値を戻すすべての状態ソフトキーのロックを解除します。
LockSkByStateAndAction	P	特定の状態に帰着し，特定の戻り値を戻すすべての状態ソフトキーをロックします。
UnlockSkByStateAndAction	P	特定の状態に帰着し，特定の戻り値を戻すすべての状態ソフトキーのロックを解除します。
ソフトキーテキスト関数		
AlGetDllEntries	F	Endindexを使用して言語DLLからいくつかのテキストを読み取ります。
AlGetDllEntriesEx	F	Endindexを使用しないで言語DLLからいくつかのテキストを読み取ります。
Change_SkTextOnScr	P	画面上のソフトキーテキストを変更します。
Al_GetSkTextByIndex	F	ソフトキーテキストを読み取ります。
Al_SetSkTextByIndex	P	ソフトキーテキストを変更します。
Al_GetSkTextByState	F	特定の状態のソフトキーテキストを読み取ります。
Al_SetSkTextByState	P	特定の状態のソフトキーテキストを変更します。
Change_SKtext	P	ソフトキーテキストを変更します。
SK_Highlight	P	ソフトキーを強調表示します。
SK_HighlightUn	P	ソフトキーを即時強調表示します。
テキスト表示		
Write_Dialog	P	テキストをダイアログ行に書き込みます。
モーダルウィンドウ関数		
ModalDialog	F	アプリケーションのモーダルウィンドウをオープンします。
ModalDialogEnd	F	モーダルダイアログをクローズします。
ModalDialogInfo	F	アクティブなモーダルダイアログ。
UsrModalDialog	F	モーダルダイアログの待ちループ。
SysModalDialog	F	システムのモーダルダイアログ。
アクション関数（状態マトリックスの動的変更）		
AlDisableSkAction	P	ソフトキーアクションを使用不可にします。

AlEnableSkAction	P	ソフトキーアクションを使用可能にします。
AlNewActionEntry	F	ソフトキーアクションを変更します。
AlNewActionEntries	F	いくつかのソフトキーアクションを変更します。
AlNewSoftkeyAction	F	後続の状態を変更します。
アクション関数（状態マトリックスの動的変更）		
subSetTFrmAttr	P	フォームの属性を設定します。
subSetTCtrlAttr	P	コントロールの属性を設定します。

7.8.1 状態制御

概説

状態を切り替えるには以下の関数を使用します。:

■ Set_Previous_State

説明

このプロシージャにより，前の状態に切り替えることができます。

用途

このプロシージャにより，OEM アプリケーションが状態番号を渡さず，ソフトキーアクションなしで，アプリケーションを前の状態に切り替えることができます。

構文

Set_Previous_State

引数

なし

■ Set_State

説明

このプロシージャにより，現在の状態から，いずれかの選択可能な状態に切り替えることができます。

用途

アプリケーションを新しい状態に変更します。この状態の番号が指定されます。

構文

Sub Set_State (ByVal state As integer)

引数

表 7.11 Set_State の引数

引数	データタイプ	説明
state	整数	変更先の状態

■ State_Changed

説明

この関数 State_Changed は，各 OEM アプリケーションのモジュール ”PRIVATE.BAS” にインプリメントしなければなりません。この関数は，構成したソフトキーをそれぞれクリックすることにより，アプリケーションのそれぞれの状態変換ごとに実行されます。関数 State_Changed により，アプリケーションはある特定のイベント，たとえばソフトキーアクションや状態の切り替えに対し限定して反応します。特に，EXIT ボタンを押すと，State_Changed を呼び出すことにより，アプリケーションを終了します。

注 : アプリケーションが EXIT で負の値の State_Changed を戻す場合，EXIT は拒否されます。

用途

アプリケーションは，状態の切り替え時に特定のデータを保管したり，ある特定の環境での状態の切り替えを拒否することなどが可能です。

構文

Function State_Changed (ByVal action As String, ByVal newstateAs Integer, ByVal oldstate As Integer) As Integer

引数

表 7.12 State_Changed の引数

引数	データタイプ	説明
action	ストリング	これによって呼び出される関数は，アプリケーションの .zus ファイルなどの中で構成されます。
newstate	整数	変更先の状態
oldstate	整数	前の状態

戻り値

関数 State_Changed の戻り値は，状態が変更されるか，提示されたアクションが実行される（戻り値 =0）か，または中断される（戻り値 =0 以外）を指定します。

ユーザプロシージャ State_Changed は，ソフトキーが押されたときにアクションを実行することができます。

注 : 各状態，また各アクションに関して，アプリケーションは，関数 State_Changed (action, newstate, oldstate) を呼び出すことにより，現在の状態（状態番号）と，その状

態を引き起こした原因となるアクション（ASCII ストリング，以下を参照）を示すメッセージを取得します。

シーケンス制御からアプリケーションに送られる次のメッセージに関して，以下にリストするグローバル定数が変更されました。

- Global Const AL_ACTIVATE = "-990"
- Global Const AL_DEACTIVATE = "-991"
- Global Const AL_FORMLOAD = "-992"
- Global Const AL_FORMUNLOAD = "-993"
- Global Const AL_NODEACTIVATE = "-994"
- Global Const AL_NOFORMUNLOAD = "-995"
- Global Const AL_ETCINFO = "-996"
- Global Const AL_EXITINFO = "-998"
- Global Const AL_HELPINFO = "-999"

これらの場合，引数 newstate は，常に現在の状態を戻します。

アプリケーションを終了するとき，アプリケーションは -1 では呼び出されなくなります。次のように，AL_EXITINFO で呼び出されます。

```
nRet = State_Changed(AL_ EXITINFO, g_nCurrState, Val(AL_EXITINFO))
```

■ State_Reached

説明

プロシージャ State_Reached は，各 OEM アプリケーションのモジュール ”PRIVATE.BAS” にインプリメントされなければなりません。この関数は，関数 State_Changed が以前に 0 の値を戻して，状態に達するとすぐに呼び出されます。

用途

変数を初期化して状態に達する場合，アクションを実行することができます。

構文

Sub State_Reached (ByVal action As String, ByVal newstate AsInteger, ByVal oldstate As Integer)

引数

表 7.13 State_Reached の引数

引数	データタイプ	説明
action	ストリング	この関数によって呼び出されるアクションは，.zus ファイルなどの中で構成されます。
newstate	整数	アプリケーションの変更先の状態
oldstate	整数	直前の状態

注 : この関数はユーザがプログラムしなければなりません。

7.8.2 照会関数

■ AlGetAccessLevel

説明

このシーケンス制御の関数により，OEM ユーザは現在のシステムのアクセスレベルを照会することができます。このため，シーケンス制御には，NCDDE サーバへのホットリンク接続があります。
(LinkItem= "/Nck/Configuration/accessLevel")

用途

OEM アプリケーションは，特定のオペレータアクションをアクセスレベルに応じてロックまたはロック解除することができます。

構文

Function ALGetAccessLevel() as integer

引数

なし

戻り値

関数 ALGetAccessLevel は，現行のアクセスレベルを戻します（エラーにつきそれぞれ 100 を返します)。

■ AlGetSkByAction

説明

関数 AlGetSkByAction は，状態のソフトキー番号を戻します。これを使用して，特定のアクションを実行することができます。

用途

OEM ユーザは，特定のアクションを実行するソフトキー番号を照会することができます。

構文

Function AlGetSkByAction(ByVal action As String) as Integer

引数

表 7.14　AlGetSkByAction の引数

引数	データタイプ	説明
action	ストリング	ソフトキーのサーチ対象のアクションテキスト

戻り値

戻り値はソフトキー番号です。エラーの場合，負の値が戻されます。

注 : 渡された文字列の長さだけが比較されます。つまり，.zus ファイルに入力されたストリングのほうが長い場合，残りは考慮されません。

■ Get_FormIdx2

説明

関数 Get_FormIdx2 は，フォームに対応する指標を与え，その指標のもとでフォームは，モジュール ”AMAIN.BAS” のグローバル配列 g_frmFormName に割り当てられます。この指標は，他のプロシージャや関数の引数として使用することができます。

用途

この関数によって，OEM ユーザはフォームの指標を，別のフォームにアドレッシングしなければならない他のプロシージャや関数の引数として使用することができます。

構文

Function Get_FormIdx2 (FormName As Form) as integer

引数

表 7.15 Get_FormIdx2 の引数

引数	データタイプ	説明
FormName	フォーム	フォームの名前

戻り値

戻り値は，グローバル配列 g_frmFormName の指標です。エラーの場合，負の値が戻されます。

Get_FormIndex

説明

関数 Get_FormIndex は，フォーム名に対応する指標を与え，その指標のもとでフォーム名は，モジュール ”AMAIN.BAS” のグローバル配列 g_frmFormName に割り当てられます。これは，他のプロシージャや関数の引数として使用することができます。

用途

この関数によって，OEM ユーザはフォームの指標を，別のフォームにアドレッシングしなければならない他のプロシージャや関数の引数として使用することができます。

構文

Function Get_FormIndex (ByVal Name As String) as integer

引数

表 7.16 Get_FormIndex の引数

引数	データタイプ	説明
Name	ストリング	フォームの名前

戻り値

戻り値は，グローバル配列 g_frmFormName の指標です。エラーの場合，負の値が戻されます。

7.8.3 MDIChild 関数

■ AlSkipFocus

説明

プロシージャ AlSkipFocus により，ある特定の MDIchild をフォーカス分配から除外することができます。

用途

構文

Sub ALSkipFocus (ByVal ChildIndex As Integer, OnOff As Integer)

引数

表 7.17 AlSkipFocus の引数

引数	データタイプ	説明
ChildIndex	整数	フォーカス分散から除外する MDIchild の指標を指定する
OnOff	整数	真：Child はフォーカス分配時にスキップされる 偽：Child は，フォーカス取得可能な Child のリストに再び追加される

注 : ALSkipFocus 呼び出しは，モーダルウィンドウでは機能しません。

■ ChildActivate

説明

プロシージャ ChildActivate は，シーケンス制御に MDIchild を通知します。

用途

このプロシージャは，イベントルーチン Form_Activate のそれぞれの MDIchild で呼び出さなければなりません。

構文

Sub ChildActivate (FormName As Form)

引数

表 7.18 ChildActivate の引数

引数	データタイプ	説明
FormName	フォーム	通知されるフォーム

注 : このプロシージャは，イベントルーチン Form_Activate で呼び出さなければなりません。

■ ChildDeactivate

説明

プロシージャ ChildDeactivate は，シーケンス制御内の MDIchild を非活動化します。アクティブなフレームがフォーカスフレームを取得します。

用途

このプロシージャは，各 MDIchild のイベントプロシージャ Form_Deactivate で呼び出さなければなりません。

構文

Sub ChildDeactivate (FormName As Form)

引数

表 7.19 ChildDeactivate の引数

引数	データタイプ	説明
FormName	フォーム	非活動化されるフォーム

注 : このプロシージャは，イベントルーチン Form_Deactivate で呼び出さなければなりません。

■ Hide_A_Child

説明

プロシージャ Hide_A_Child により，個々の MDIchild を隠すことができます。

用途

構文

Sub Hide_A_Child (FormName As Form)

引数

表 7.20 Hide_A_Child の引数

引数	データタイプ	説明
FormName	フォーム	隠される MDIchild ／フォーム

■ Hide_Childs

説明

プロシージャ Hide_Childs は，ロードされたすべての MDIchild が画面に表示されないように隠し，Child の記述にフラグ CH_HIDDEN をセットします。

用途

Hide_ Childs により，現在ロードされているすべての MDIchild を隠すことができます。

構文

Sub Hide_Childs

引数

なし

■ Set_ChildType

説明

関数 Set_ChildType により，実行時に個々の MDIchild のタイプを変更することができます。変更可能なタイプは表 7.7 にリストされています。

用途

この関数は，*.MDI ファイルで定義されたタイプを動的に変更するのに使用することができます。

構文

Function Set_ChildType (ByVal ChildIndex as Integer, ByVal newType As Integer) as Integer

引数

表 7.21 Set_ChildType の引数

引数	データタイプ	説明

ChildIndex	整数	隠される MDIchild ／フォーム
newType	整数	設定する Child のタイプ TYPE_NOUNLOAD 0 TYPE_UNLOAD 1 TYPE_RESIZE 2 TYPE_MODAL 3 TYPE_OVERLAYED 4 TYPE_MAYBE_1OR2 5

戻り値

戻り値は整数として，子の変更前のタイプです。

■ Show_Focus

説明

プロシージャ Show_Focus により，MDIchild でのエントリフォーカスを設定することができます。

用途

このプロシージャにより，状態の移行に続くエントリフォーカスのみを設定することができます。

構文

Sub Show_Focus(mdiChild as Form)

引数

表 7.22 Show_Focus の引数

引数	データタイプ	説明
mdiChild	フォーム	フォーカスを取得する MDIChild ／フォーム

■ Show_A_Hidden_Child

説明

プロシージャ Show_A_Hidden_Child は，フラグ CH_HIDDEN が立てられた MDIchild をオープンします。フラグは CH_LOADED に変更されます。

用途

プロシージャ Show_A_Hidden_Child により，現在隠されている MDIChildwhich を表示することができます。

構文

Sub Show_A_Hidden_Child(ByVal fname As Form)

引数

なし

■ Show_Hidden_Childs

説明

プロシージャ Show_Hidden_Childs は，フラグ CH_HIDDEN が立てられているすべての MDIchild をオープンします。フラグは CH_LOADED に変更されます。

用途

プロシージャ Show_Hidden_Childs により，現在隠されているすべての MDIChild をオープンすることができます。

構文

Sub Show_Hidden_Childs

引数

なし

7.8.4 ソフトキーのロック／アンロック

■ AL_RESUMESKS

説明

プロシージャ AL_RESUMESKS を呼び出すのは，AL_STOPSKS を呼び出した後にしなければなりません。そうしないと，OEM アプリケーションへのエントリはブロックされてしまいます。

用途

必ず AL_STOPSKS と併用する

構文

Sub AL_RESUMESKS()

引数

なし

■ AL_STOPSKS

説明

ルーチン AL_STOPSKS を使用して，ルーチン AL_RESUMESKS がソフトキーをクリックして起きるアクション実行のロックを解除するまで，それをブロックすることができます。

AL_STOPSKS 呼び出しの後に押されたソフトキーは，シーケンス制御に渡され，AL_STOPSKS に続く最初のソフトキーが AL_RESUMESKS 呼び出しの後，保管され処理されます（Regie のルーチン StopRegieEvents，ResumeRegieEvents と比較してください)。他のすべてのエントリは失われてしまいます。

用途

このルーチンはアクションと，以降の OEM アプリケーションへのソフトキーエントリをブロックします。

構文

Sub AL_STOPSKS()

引数

なし

注 : AL_STOPSKS が呼び出された後に AL_RESUMESKS が呼び出されない場合，システムはこのアプリケーションでハングアップします。領域が切り替わった後に再びシステムを作動させることができます。

■ LockSkByAction

説明

プロシージャ LockSkByAction は，指定した引数 action と同一の ReturnString（状態マトリックスからのアクション）を持つ，現行の状態のソフトキーをすべてロックします。

用途

特定のアクションに帰着するソフトキーを限定してロックすることができます。

構文

Sub LockSkByAction(ByVal action As String)

引数

表 7.23 LockSkByAction の引数

引数	データタイプ	説明
action	ストリング	ロックされるアクション

■ LockSkByState

説明

プロシージャ LockSkByState は，引数 state によって指定された状態へと導く現在の状態のソフトキーすべてをロックします。

用途

特定の状態に変更するソフトキーを限定してロックすることができます。

構文

Sub LockSkByState(ByVal state As Integer)

引数

表 7.24 LockSkByState の引数

引数	データタイプ	説明
state	整数	ソフトキーの操作に対してロックされる状態

■ LockSkByStateAndAction

説明

プロシージャ LockSkByStateAndAction は，引数 state によって指定された状態へと導く，また指定した引数 action と同一の ReturnString（状態行列からのアクション）を持つ，現在の状態のソフトキーすべてをロックします。

用途

特定のアクションに帰着し，特定の状態に変更するソフトキーを限定してロックすることができます。

構文

Sub LockSkByStateAndAction(ByVal state As Integer,ByVal action As String)

引数

表 7.25 LockSkByStateAndAction の引数

引数	データタイプ	説明
state	整数	ソフトキーの操作に対してロックされる状態
action	ストリング	ロックされるアクション

■ Lock_Softkey

説明

プロシージャ Lock_Softkey により，個々のソフトキー，完全な水平ソフトキーバーか垂直ソフトキーバーのいずれか（あるいは両方）を明示的にロックすることができます。

H_Lock，V_Lock，HV_Lock の数値は，モジュール ALDECL.BAS で定義されます。

ソフトキーは，新しいソフトキーバーが同じ場所に置かれたときに自動的にロックを解除されます。

ロックされたソフトキーは，Windows 定数 GRAY_TEXT=&H8000 0011 のグレー化（使用不可）テキストにより，（Windows と同じように）背景色が薄いグレーになることによって表されます。

Lock_Softkey，Unlock_Softkey は，アクションが構成されたソフトキーでのみ有効です。次のようなソフトキーでは有効ではありません。

- テキストが空，またはテキストがスペース文字
- アクションに対して構成されなかった
- AlEnableSkAction によって明示的に使用不可になった

用途

このプロシージャは，指定した状態のソフトキーまたはソフトキーバーをロックします。

構文

Sub Lock_Softkey(skIndex As Integer)

引数

表 7.26 Lock_Softkey の引数

引数	データタイプ	説明
skIndex	整数	H_Lock： 水平ソフトキーバーをロックする V_Lock： 垂直ソフトキーバーをロックする HV_Lock： 水平および垂直ソフトキーバーをロックする 0 ～ MaxSoftKeys： 指標 SoftkeyId を使用してソフトキーをロックする

注：状態が前のものに切り替わった後，以前にロックされていたソフトキーのロックは解除されます。

■ UnlockSkByAction

説明

プロシージャ UnlockSkByAction は，指定した引数 action と同一の ReturnString（状態行列からのアクション）を持つ，現行の状態のソフトキーのロックをすべて解除します。

用途

特定のアクションに帰着するソフトキーのロックを解除することができます。

構文

Sub UnlockSkByAction(ByVal action As String)

引数

表 7.27 UnlockSkByAction の引数

引数	データタイプ	説明
action	ストリング	保留解除されるアクション

■ UnlockSkByState

説明

プロシージャ UnlockSkByState は，引数 state によって指定された状態へと導く現在の状態のソフトキーのロックをすべて解除します。

用途

特定の状態に変更するソフトキーのロックを解除することができます。

構文

Sub UnlockSkByState(ByVal state As Integer)

引数

表 7.28 UnlockSkByState の引数

引数	データタイプ	説明
state	整数	ソフトキーに応じて保留解除される状態

■ UnlockSkByStateAndAction

説明

プロシージャ UnlockSkByStateAndAction は，引数 state によって指定された状態へと導く，また指定した引数 action と同一の ReturnString（状態行列からのアクション）を持つ，現在の状態のソフトキーのロックをすべて解除します。

用途

特定の状態に変更するソフトキーのロックを解除することができます。

構文

Sub UnlockSkByStateAndAction(ByVal state As Integer,ByVal action As String)

引数

表 7.29 UnlockSkByStateAndAction の引数

引数	データタイプ	説明
state	整数	ソフトキーに応じて保留解除される状態
action	ストリング	保留解除されるアクション

■ Unlock_Softkey

説明

プロシージャ Unlock_Softkey は，個々のソフトキー，または完全な水平，垂直のソフトキーバーのロックをキャンセルします。

用途

ソフトキーのロックを解除できます。

構文

Sub Unlock_Softkey(skIndex As Integer)

引数

表 7.30 Unlock_Softkey の引数

引数	データタイプ	説明
skIndex	整数	H_Lock： 水平ソフトキーバーのロックを解除します。 V_Lock： 垂直ソフトキーバーのロックを解除します。 HV_Lock： 水平および垂直ソフトキーバーのロックを解除します。 0 ～ MaxSoftKeys： 指標 skIndex を使用してソフトキーのロックを解除します。

7.8.5 ソフトキーテキスト関数

■ ALGetDLLEntries

説明

シーケンス制御には，言語 DLL から同時にいくつかのテキストを読み取るための関数 ALGetDLLEntries が備えられています。テキストは（言語 DLL に 1 度アクセスして）フォームごとに読み取ることができ，またフォームに一致するラベルを割り当てることができます。

用途

言語 DLL からいくつかのテキストを読み取るために常に使用すべきです。これは各テキストを個々に読み取るよりもはるかに効率的です。

構文

Function ALGetDLLEntries (ByVal hLang As Integer, ByVal t_begAs Integer, ByVal t_end As Integer, ByVal t_max As Integer, dlltAs DLLEntry_Type) As Integer

引数

表 7.31 ALGetDLLEntries の引数

引数	データタイプ	説明
hLang	整数	*hLanguageLibHandle は言語DLL を指定し, DLL のロード時に戻される*
t_beg	整数	テキスト領域の開始のマークを付ける
t_end	整数	テキスト領域の終了のマークを付ける
t_max	整数	テキストの最大数
dllt	DLL-Entry_Type	テキストが書き込まれる構造を指定する。この構造はシーケンス制御のタイプ宣言として提供される。

戻り値

関数は実際に読み取られたテキストの数を整数として戻します。

言語 DLL からのいくつかのテキストの読み取り

関数 ALGetDLLEntries は 10 のテキストを言語 DLL からロードし，それぞれの Visual Basic コントロールに割り当てます。

■ 例 7-11 いくつかのテキストの言語 DLL からの読み取り

```
Sub Form_Load (
Dim g_tDI_DlgText() As DLLEntry_Type, nRet As Integer
Const TEXTSTART = 26
Const TEXTNUM = 10
ReDim g_tDI_DlgText(0 To TEXTNUM – 1
nRet = ALGetDLLEntries(g_hLanguageLibHandle, TEXTSTART, TEXTSTART +_
TEXTNUM, TEXTNUM, g_tDI_DlgText(0))
Label1.Caption = g_tDI_DlgText(0).DLLText
Label1.Caption = g_tDI_DlgText(1).DLLText
End Sub
```

■ ALGetDLLEntriesEx

説明

シーケンス制御には，言語 DLL から同時にいくつかのテキストを，終了指標なしで読み取るための関数 ALGetDLLEntries が備えられています。

この関数は，指定した位置から開始して，言語 DLL の Index-gap かまたは指定した数に達するまで，言語 DLL からテキストエントリを読み取ります。戻り値は，実際に読み取られたテキストの数です（このコンテキストでは，Index-gap は，指定した範囲にテキスト番号が割り当てられなかったため，C 関数 LoadString がこのギャップを終了用に用いることを意味します)。

用途

いくつかのテキストが，終了指標なしで言語 DLL から読み取られるときはいつも使用するべきです。これはテキストを個々に読み取るよりもずっと高速です。

構文

Function ALGetDLLEntriesEx (ByVal hLang As Integer, ByValt_beg As Integer, ByVal t_max As Integer, dllt As DLLEntry_Type)As Integer

引数

表 7.32 ALGetDLLEntriesEx の引数

引数	データタイプ	説明
hLang	整数	*hLanguageLibHandle は言語 DLL を指定し，DLL のロード時に戻される*
t_beg	整数	テキスト領域の開始のマークを付ける
t_max	整数	テキストの最大数
dllt	DLL-Entry_Types	テキストが書き込まれる構造を指定する。この構造はシーケンス制御のタイプ宣言として提供される。

戻り値

戻り値は実際に読み取られたテキストエントリの数（整数）です。

言語 DLL のいくつかのテキストの読み取り

ALGetDLLEntriesEx 関数は，10 のテキストを言語 DLL からロードし，それぞれの Visual Basic コントロールに割り当てます。

■ 例 7-12 いくつかのテキストの言語 DLL からの読み取り

```
Sub Form_Load ()
Dim g_tDI_DlgText() As DLLEntry_Type, nRet As Integer
Const TEXTSTART = 26
Const TEXTNUM = 10
ReDim g_tDI_DlgText(0 To TEXTNUM - 1)
nRet = ALGetDLLEntriesEx(g_hLanguageLibHandle, TEXTSTART,
TEXTNUM,g_tDI_DlgText(0))
Label1.Caption = g_tDI_DlgText(0).DLLText
Label1.Caption = g_tDI_DlgText(1).DLLText
End Sub
```

■ AL_GetSKState

説明

この関数は，ソフトキーがロックされているかどうかを検出します。

用途

ソフトキーの状態がロックされているかロックされていないかを照会できます。

構文

Function AL_GetSKState(sk As Integer) As Integer

引数

表 7.33 AL_GetSKTextByIndex の引数

引数	データタイプ	説明
sk	整数	ソフトキー番号

戻り値

戻り値は，ソフトキーがロックされている場合は FALSE，ロックされていない場合は TRUE です。

■ AL_GetSKTextByIndex

説明

プロシージャ AL_GetSKTextByIndex により，現在の状態とは関係なく，アプリケーションは個々のソフトキーテキストを読み取ることができます。

用途

現在の状態とは関係なく，実行時にソフトキーテキストを読み取ることができます。

構文

Function AL_GetSKTextByIndex(ByVal hv As Integer,ByValIndex As Integer)As String

引数

表 7.34 AL_GetSKTextByIndex の引数

引数	データタイプ	説明
hv	整数	真： 水平ソフトキー 偽： 垂直ソフトキー
Index	整数	言語 DLL 内の位置番号

戻り値

プロシージャはストリングとしてソフトキーテキストを戻します。

■ AL_SetSKTextByIndex

説明

AL_SetSKTextByIndex により，アプリケーションは，各ソフトキーのテキストを，引数 sktext に含まれるテキストで上書きすることができます。この上書きはテキスト DLL では実行されません。これが有効なのはアプリケーションが終了するまでの間だけです。

用途

このプロシージャは，実行時に現在の状態に依存してソフトキーテキストを修正する場合（たとえば，INI ファイルを含むソフトキーテキストを表示する場合など）に役

立ちます。

構文

Sub AL_SetSKTextByIndex(ByVal hv As Integer,ByVal Index As Integer, ByVal sktext As String)

引数

表 7.35 AL_SetSKTextByIndex の引数

引数	データタイプ	説明
hv	整数	真：水平ソフトキー▼偽：垂直ソフトキー
Index	整数	言語 DLL 内の位置番号
sktext	ストリング	ソフトキーに割り当てられるテキスト

■ AL_GetSKTextByState

説明

関数 AL_GetSkTextByState は，state，Action，sk によって指定されたソフトキーテキストを提供します。

用途

この関数は現在の状態に依存して，ソフトキーテキストを読み取ります。

構文

Function AL_GetSkTextByState(ByVal state As Integer, ByValaction As Integer, ByVal sk As Integer) As String

引数

表 7.36 AL_GetSKTextByState の引数

引数	データタイプ	説明
state	整数	状態
action	整数	言語 DLL 内の位置番号
sk	整数	ソフトキー番号

戻り値

戻り値はストリングとしてのソフトキーテキストです。

■ AL_SetSKTextByState

説明

ルーチン AL_SetSkTextByState により，引数 state，Action，sk によって指定されたソフトキーテキストを変更することができます。

用途

現在の状態に依存したソフトキーテキストを書き込むことができます。

構文

Sub AL_SetSkTextByState(ByVal state As Integer, ByVal action AsInteger, ByVal sk As Integer, ByVal sktext As String)

引数

表 7.37 AL_SetSKTextByState の引数

引数	データタイプ	説明
state	整数	状態
action	整数	言語 DLL 内の位置番号
sk	整数	ソフトキー番号
sktext	ストリング	ソフトキーに割り当てられるテキスト

■ Change_SkText

説明

プロシージャ Change_SkText により，アプリケーションは実行時に SoftkeyId として指定された単一ソフトキーのテキストを，SkText のテキストで上書きすることができます。この上書きはテキスト DLL では実行されません。これが有効なのはアプリケーションが終了するまでの間だけです。

用途

実行時にソフトキーテキストを変更することができます。たとえば，.INI ファイルからのソフトキーテキストを表示させなければならない場合などに実行できます。

構文

Sub Change_SkText (SoftkeyId As Integer, SkText As String)

引数

表 7.38 Change_SkText の引数

引数	データタイプ	説明
SoftkeyId	整数	ソフトキーの番号
SkText	ストリング	ソフトキーに割り当てられるテキスト

■ Change_SkTextOnScr

説明

ルーチン Change_SkTextOnScr は，画面上で直接，引数 index で指定したソフトキーテキストを変更します。ただし内部データ構造（シーケンス制御）では変わりません。

用途

このルーチンは，状態が変わるまでプログラムのソフトキーを変更する場合に役立ちます。

構文

Sub Change_SkTextOnScr (SoftkeyId As Integer, SkText AsString)

引数

表 7.39 Change_SkTextOn Scr の引数

引数	データタイプ	説明
SoftkeyId	整数	ソフトキー番号
SkText	ストリング	ソフトキーに割り当てられるテキスト

■ SK_Highlight

説明

プロシージャ SK_Highlight は，パラメータ指標によってアドレッシングされたソフトキーを強調表示します
(一方，プロシージャ SK_HighlightUn は対応するソフトキーを即時に強調表示します)。

構文

Sub SK_HighLight(ByVal Index As Integer)

■ SK_HighlightUn

説明

プロシージャ SK_HighlightUn は，パラメータ指標でアドレッシングされたソフトキーを，背景色を青くして即時に強調表示します（一方，関数 SK_Highlight はアクションが終了するまでは対応するソフトキーを強調表示しません)。

構文

Sub SK_HighlightUn(ByVal Index As Integer)

7.8.6 テキストの表示

■ Write_Dialog

説明

プロシージャ Write_Dialog は，引数 Text で指定した ASCII テキストをダイアログ行に書き込みます。

用途

この関数は，たとえばダイアログフィールドのダイアログ行を表示する場合に役立ちます。

構文

Sub Write_Dialog(ByVal DText As String)

引数

表 7.40　Write_Dialog の引数

引数	データタイプ	説明
DText	ストリング	ダイアログ行のテキスト

7.8.7 モーダルウィンドウ関数

概説

新規フォームのモーダルダイアログは，状態マトリックスによってではなく，ModalDialog を呼び出すことによって処理されるようになりました。モーダルウィンドウの内容を書き込むための Write-ModalDlg プロシージャは，もはや必要なくなります。

注 : 以前と同様に，アプリケーションモーダルダイアログ（ModalDialog か UsrModalDialog のいずれか）は，1 度に 1 つしかオープンできません。

この新しいプロシージャは，主に，モーダルダイアログの使用を単純化し，また特にモーダルダイアログの認識処理を単純化します。加えて，アプリケーションの状態の数を削減します。

モーダルフォームは，（ソフトキーをクリックすることによって）モーダルダイアログの終了時に，ユーザに通知される前にアンロードされます。そのため，エントリフィールドの Change イベント，およびフォームの Unload イベントにそれぞれ現行値が一時的に保管される場合，エントリフィールドを持つモーダルダイアログは，この新しい関数によってのみ実現することができます。エントリフィールドを持つモーダルダイアログは，もはやモーダルダイアログとしてではなく，状態転換として実現できることになりました。

■ ModalDialog

説明

関数 ModalDialog により，アプリケーションはアプリケーションモーダルダイアログをオープンすることができます。アプリケーションは，ダイアログが終了するか，または関数 ModalDialogEnd -1 により中断されるまで，待ちループ状態のままとなります。モーダルダイアログもまた MDIChild であり，シーケンス制御に追加されなければなりません（7.22 と 7.23 を参照してください）。

- ソフトキーを表示します。hsk，vsk それぞれの値が -1 であるということは，

対応するソフトキーバーにソフトキーがラベル付けされないことを意味します。少なくとも hsk，vsk のうちのいずれかの値が -1 以外の値でなければなりません。

- モーダルウィンドウをオープンし，モーダルタイトル，モーダルテキストを表示します。
- ソフトキーによる終了を待ちます。

たとえば，ソフトキー 15 をクリックします。

- モーダルウィンドウをクローズします。
- ソフトキーバーとフォーカスを復元します。
- 仮想ソフトキー 15 が使用されるとすれば，戻り値 =15 となります。
- ユーザが戻り値を評価します。

用途

この関数を使用して，照会とメッセージダイアログを表示することができます。

構文

Function ModalDialog(ByVal Title As String, ByVal midx As Inte-ger,ByVal hsk As Integer, ByVal vsk As Integer, ByVal mtext AsString) As Integer

引数

表 7.41 ModalDialog の引数

引数	データタイプ	説明
Title	ストリング	モーダルダイアログのタイトルバー
midx	整数	g_frmFormName のモーダルウィンドウの指標
hsk	整数	水平ソフトキーバー。-1 の場合，ソフトキーなし。
vsk	整数	垂直ソフトキーバー。-1 の場合，ソフトキーなし。
mtext	ストリング	モーダルウィンドウのテキスト。

戻り値

関数は，モーダルダイアログを認識するソフトキーの指標か，またはダイアログが関数 ModalDialogEnd -1 によって中断された場合の -1 のいずれかを戻します。

ダイアログのオープン

水平ソフトキーなしで ”delete” というタイトルのダイアログをオープンします。垂直ソフトキーは，言語 DLL の指標 8 で始まります。

■ 例 7-13 ダイアログのオープン

```
ModalDialog ("delete", Get_FormIndex("ApplModal1"), -1, 8,"are you sure, you want to delete ?")
```

■ ModalDialogEnd

説明

関数 UsrModalDialog または ModalDialog によりオープンしていたアプリケーションモーダルダイアログを終了します。

用途

この関数を使用して，照会とメッセージダイアログを終了することができます。

構文

Sub ModalDialogEnd(ByVal Index As Integer)

引数

表 7.42 ModalDialog の引数

引数	データタイプ	説明
Index	整数	値 -1 はダイアログを終了する

ダイアログのクローズ

アプリケーションモーダルダイアログをクローズします。

■ 例 7-14 ダイアログのクローズ

```
ModalDialogEnd -1
```

■ ModalDialogInfo

説明

この関数は，モーダルダイアログがオープンしているかどうかを指し示します。

用途

モーダルダイアログがオープンしているかどうかを指し示すのに役立ちます。

構文

Function ModalDialogInfo() As Integer

引数

なし

戻り値

この関数は，モーダルまたはシステムモーダルウィンドウがオープンしている場合，TRUE の値を戻します。

■ UsrModalDialog

説明

関数 UsrModalDialog により，アプリケーションは，モーダルダイアログ用の待ち

ループを単独でプログラムすることができます。このため，モーダルウィンドウがオープンした後，アプリケーションは，次の引数を使用してメッセージ State_Changed により呼び出されます。

```
nRet = State_Changed (AL_USRMODALDLG, g_nCurrState,Val(AL_USRMODALDLG))
```

関数 UsrModalDialog は，モーダルダイアログを終了させるソフトキーの指標を戻します。この指標はグローバル変数に格納されます。この変数を関数 UsrModalDialogIndex によって照会することができます。

```
Index = UsrModalDialogIndex
```

この戻り値により，ユーザは待ちループから出ることができます。

モーダルダイアログは，次の関数を呼び出して，プログラムを終了することができます。

```
ModalDialogEnd -1.
```

この場合，UsrModalDialog の呼び出し元が自ら，この打ち切りに同意するかどうかを決定しなければなりません。

関数 UsrModalDialog により，アプリケーションは，アプリケーションモーダルダイアログをオープンすることができます。アプリケーションは，この関数では，ダイアログが認識されるか，または関数 ModalDialogEnd -1 によって終了されるまで，待ちループに入ったままです。ユーザは待ちループをプログラムする必要があります。

用途

この関数は，照会とメッセージダイアログを，それぞれの独自の待ちループと共にプログラムする場合に役立ちます。

構文

Function ModalDialog(ByVal Title As String, ByVal midx As Inte-ger,ByVal hsk As Integer, ByVal vsk As Integer, ByVal mtext AsString) As Integer

引数

表 7.43 UsrModalDialog の引数

引数	データタイプ	説明
Title	ストリング	モーダルダイアログのタイトル行
midx	整数	g_frmFormName のモーダルウィンドウの指標
hsk	整数	水平ソフトキーバー。-1 の場合，ソフトキーはラベル付けされない。
vsk	整数	垂直ソフトキーバー。-1 の場合，ソフトキーはラベル付けされない。
mtext	ストリング	モーダルウィンドウのテキスト。

戻り値

関数は，モーダルダイアログを認識するソフトキーの指標か，またはダイアログが関数 ModalDialogEnd -1 によって終了された場合の -1 のいずれかを戻します。

ダイアログのオープン

水平ソフトキーなしで ”delete” というタイトルのダイアログをオープンします。垂直ソフトキーは，言語 DLL の指標 8 で始まります。

■ 例 7-15 ダイアログのオープン

```
nRet = UsrModalDialog ("delete", 13, -1, 8, "MODAL-TEXT")
```

■ SysModalDialog

説明

関数 SysMododalDialog により，アプリケーションは，システムモーダルダイアログをオープンすることができます。

この関数の使用時には，モジュール ALMODAL.FRM を（フォーム ApplModal と共に）プロジェクトに組み込まなければなりません。ただし，フォームは必ずしもリストの状態マトリックス，MDI Chilid（Amain.bas，xxx.mdi）にリストされる必要はありません。

アプリケーションモーダルダイアログが関数 SysModalDialog を呼び出した時にオープンしている場合，システムモーダルダイアログを終了させた後にコントロールはアプリケーションモーダルダイアログに戻ります。

注 : 1 度にオープンできるのは，1 つのシステムモーダルダイアログのみです。

注 : 独自の ApplModal フォームを使用している場合，そのフォームには，title という名前のラベルと ModalText というテキストボックスが含まれていなければなりません。シーケンス制御によって提供されている formApplModal には，これら 2 つのコントロールがすでに含まれています。

アプリケーションは，この関数では，ダイアログが認識されるか，または関数 ModalDialogEnd -1 によって終了されるまで，待ちループに入ったままです。

- ソフトキーを表示する。hsk，vsk それぞれの値が -1 であるということは，対応するソフトキーバーにソフトキーがラベル付けされないことを意味します。少なくとも hsk，vsk のうちのいずれかの値が -1 以外の値でなければなりません。
- モーダルウィンドウをオープンし，モーダルタイトル，モーダルテキストを表示します。
- ソフトキーによる終了を待ちます。

--> たとえば，ソフトキー 15 をクリックします。

- モーダルウィンドウをクローズします。
- ソフトキーバーとフォーカスを復元します。
- ソフトキー 15 が使用されるように構成されたとすれば，戻り値 =15 になります。
- ユーザが戻り値を評価します。

用途

アプリケーションモーダルウィンドウをオープンするのに役立ちます。

構文

Function SysModalDialog(ByVal Title As String, ByVal hsk AsInteger, ByVal vsk As Integer, ByVal mtext As String) As Integer

引数

表 7.44 ModalDialog の引数

引数	データタイプ	説明
Title	ストリング	モーダルダイアログのタイトル行
hsk	整数	水平ソフトキーバー。-1 の場合，ソフトキーはラベル付けされない。
vsk	整数	垂直ソフトキーバー。-1 の場合，ソフトキーはラベル付けされない。
mtext	ストリング	システムモーダルウィンドウのテキスト。

戻り値

アプリケーションは，ダイアログが認識されるか，または関数 ModalDialogEnd -1 によって終了されるまで，この関数の待ちループに入ったままです。

システムモーダルダイアログのオープン

水平ソフトキーなしで ”warning” というタイトルのダイアログをオープンします。垂直ソフトキーは，言語 DLL の指標 8 で始まります。

■ 例 7-16 システムモーダルダイアログのオープン

```
nRet = SysModalDialog ("warning", -1, 8, "MODAL-TEXT")
```

7.8.8 アクション関数（状態マトリックスの動的変更）

■ ALDisableSKAction

説明

プロシージャ ALDisableSKAction により，状態行列 state および SKIndex で構成されたソフトキーアクションを，動的に（一時的に）使用不可とすることができます。すると，対応するソフトキーテキストはソフトキーバーから除去され，ソフトキー自体はロックされます。

用途

このプロシージャは，ソフトキーアクションを動的に使用不可にする場合に役立ちます。

構文

Sub ALDisableSKAction(ByVal state As Integer, ByVal SkIndex AsInteger)

引数

表 7.45 ALDisableSKAction の引数

引数	データタイプ	説明
state	整数	状態の番号
SkIndex	指標	ソフトキーの番号

■ ALEnableSKAction

説明

プロシージャ ALEnableSKAction により，引数 state および SKIndex により指定されたソフトキーアクション，また ALDisableSKAction により使用不可となったソフトキーアクションを，動的に使用可能にすることができます。すると，対応するソフトキーテキストは，ソフトキーバーに表示され，ソフトキーのロックは解除されます。

用途

このプロシージャは，ソフトキーアクションを動的に使用可能にする場合に役立ちます。

構文

Sub ALEnableSKAction(ByVal state As Integer,ByVal SkIndex As Integer)

引数

表 7.46 ALEnableSKAction の引数

引数	データタイプ	説明
state	整数	状態番号
SkIndex	指標	ソフトキー番号

■ ALNewActionEntry

説明

関数 ALNewActionEntry により，実行時に，現在の状況に対応する SkIndex 番号を使用して，アクションを変更することができます。こうして，実行時にアプリケーションの反応を再構成することができます。変更が現行の状態を参照する場合，水平，垂直それぞれのソフトキーバーは，必要に応じて再描画されます。

注 : アクション時にオープンされることになっている MDIchild は変更できません。

用途

この関数は，実行時にアプリケーションの状態変換を変更する場合に役立ちます。

構文

Function ALNewActionEntry(ByVal state As Integer, ByVal Skin-dexAs Integer, ByVal

AccessLevel As Integer, ByVal HskTextIdAs Integer, ByVal VskTextId As Integer, ByVal ReturnString AsString, ByVal Successor As Integer) As Integer

引数

表 7.47 ALNewActionEntry の引数

引数	データタイプ	説明
state	整数	状態番号
SkIndex	指標	ソフトキー番号
AccessLevel	整数	アクセスレベル
HskTextId	整数	水平ソフトキーテキストバー
VskTextId	整数	垂直ソフトキーテキストバー
ReturnString	ストリング	戻りストリング
Successor	整数	後継状態

戻り値

関数は 0 か，エラーの場合は -1 を戻します。

領域切り替えの追加

次の例では，内部状態マトリックスの状態 3 で後継状態 7 が入れられていますが，アクセスレベルは変更されていません。

■ 例 7-17 状態切り替えの追加

```
nRet = ALNewActionEntry (3, 8, -1, 0, 32, "abc", 7)
```

■ ALNewActionEntries

説明

ALNewActionEntries により，実行時に，引数 SkIndex（指標 0 ～ 16）により指定された現行の状態の状態マトリックスのエントリ（アクション）すべてを変更することができます。こうして，実行時にアプリケーションの反応を再構成することができます。

状態マトリックスの新しい値は，VB 構造の配列で渡されます。

```
Type Sk_MatEntry
SkIndex As Integer ‘ アクション
AccessLevel As Integer ‘ アクセスレベル
HskTextId As Integer ‘ 水平 SK テキストバー
VskTextId As Integer ‘ 垂直 SK テキストバー
ReturnString As String ‘ 戻りテキスト
Successor As Integer ‘ 後継状態
End Type
```

タイプ定義 Sk_MatEntry は，モジュール ALDECL.BAS にあります。

注 : アクション時にオープンされることになっている MDIchild は変更できません。

用途

この関数は，実行時にアプリケーションの状態変換を変更する場合に役立ちます。

構文

Function ALNewActionEntries(ByVal state As Integer, SkMatEn-try()As Sk_MatEntry) As Integer

引数

表 7.48 ALNewActionEntries の引数

引数	データタイプ	説明
state	整数	状態番号
SkMatEntry()	Sk_MatEntry	softkey number

戻り値

関数は 0 か，エラーの場合は -1 を戻します。

状態切り替えの追加

次の例では，状態マトリックスの状態 3 に，いくつかのアクションが入れられています。

■ 例 7-18 ダイアログのオープン

```
ReDim xxSkMatEntry(4) As Sk_MatEntry
xxSkMatEntry(0).SkIndex = 8
xxSkMatEntry(0). AccessLevel = -1
xxSkMatEntry(0). HskTextId = 0
xxSkMatEntry(0). HskTextId = 32
xxSkMatEntry(0).ReturnString = "abc"
xxSkMatEntry(0). Successor = 7
xxSkMatEntry(1).SkIndex = 10
xxSkMatEntry(1). AccessLevel = -1
xxSkMatEntry(1). HskTextId = 40
xxSkMatEntry(1). HskTextId = 48
xxSkMatEntry(1).ReturnString = "xyz"
xxSkMatEntry(1). Successor = 13
a.m.
nRet = ALNewActionEntries (3, xxSkMatEntry() )
```

■ ALNewSoftkeyAction

説明

ALNewSoftkeyAction 関数を使用して，実行時に，現行の状態（指標 0 ～ 16）の後継状態と，実行時にアプリケーションに戻される値の両方を変更することができます。

ReturnString が変更されるのは，引数 ReturnString が空ストリングでない場合だけです。

用途

この関数は，次に切り替えられるまでプログラム内のソフトキーアクションを変更する場合に役立ちます。

構文

Function ALNewActionEntry(ByVal state As Integer, ByVal SkIn-dexAs Integer, ByVal ReturnString As String, ByVal Successor AsInteger) As Integer

引数

表 7.49 ALNewActionEntry の引数

引数	データタイプ	説明
state	整数	状態番号
SkIndex	指標	ソフトキー番号
ReturnString	ストリング	戻りストリング
Successor	整数	水平ソフトキーテキストバー

戻り値

関数は 0 か，エラーの場合は -1 を戻します。

アクションの新規割り当て

後継状態 7 を状態 3 のソフトキー 8 に割り当てます。

■ 例 7-19 ダイアログのオープン

```
nRet = ALNewSoftkeyAction (3, 8, "abc", 7 )
```

7.8.9 動的グラフィック解像度の関数

概説

PC バージョンの MMC アプリケーションを，設定された画面サイズで表示するために，P5 のシーケンス制御はフォームとフォーム上に表示されるコントロールやテキストのサイズをスケール調整します。しかしこの処理は，シーケンス制御（SC）に認識されサイズ変更されるフォームとコントロールだけに実行できます。
SC でスケール調整できるのはコントロールのサイズだけです。内容に変更を加えることはできません（グリッドやリストボックスなどには SC で認識されない依存関係があります。リストボックスの行のスペーシングなどが該当します)。
この機能をアプリケーションに対して使用できるようにするには，グローバル INI ファイルである MMC.INI の中でこの機能をアクティブにする必要があり，このファイルのプログラムコードに以下のような多少の変更を加える必要があるかもしれません。

グローバル INI ファイル MMC.INI のセクション [CONTROL] 内のエントリ：

```
;==Resolution: 0=fixed (640x480), 1=variable, default:0
Resolution=1
;==BaseScreen: 640x480, 800x600, ... default: 640x480
```

BaseScreen=640x480

- Resolution=1（可変）は，PC に実際に設定されている画面解像度が使用されることを示します。
- BaseScreen=640x480 は，アプリケーションが開発された際の解像度を指定します。弊社の場合は，これまで常に（!!!）640x480（デフォルト設定）でした。つまり，普通はこのエントリに変更を加える必要はありません。

Resolution=1 の場合は，INI エントリの ScreenTwips（アプリケーション固有の INI ファイルにある）は無視されます。ScreenTwips=1 は，コントロールで設定されている値に基づく実際の条件とは無関係に，ピクセル当たりの twips の数を 15 に設定します。

シーケンス制御にはグローバルデータ構造が含まれており，アプリケーションに関係のある以下のデータがすべてここに保持されています。

```
Type AppRes_Info
HSize As Integer   水平解像度
VSize As Integer   垂直解像度（タスクバーを除く）
HFact As Single    BaseScreen-X を参照する X 係数
VFact As Single   （BaseScreen-Y-Taskbar）を参照する Y 係数
End Type

Global g_tAppRes As AppRes_Info
```

コントロールまたはフォームの水平解像度（垂直解像度）に，g_tAppRes.HFact（水平），g_tAppRes.Vfact（垂直）の数量がそれぞれ乗算されます。

プロシージャ subSetTCtrlAttr および subSetTFrmAttr には，オプションパラメータがあり，このパラメータは対応するコントロールのフォントサイズに係数 g_tAppRes.Hfactor を乗算するかどうかを指定します。パラメータを指定しなかったり FALSE にした場合は係数は 1 になり，それ以外は g_tAppRes.Hfact になります。

■ SubSetTFrmAttr

説明

プロシージャ subSetTCtrlAttr は，パラメータ Ctrl で指定された，形式 wType のコントロールの属性 FontName，Font-Size，および FontBold を設定します。

アプリケーションにより，FormLoad 中に，すべてのコントロールについてこのプロシージャを呼び出す必要があります。こうすると，アプリケーションのフォームの表示が一様になり，言語を転換しても正しい FontAttributes が得られるようにすることができます。

プロシージャ subSetTCtrlAttr(Ctrl As Control, wType As Integer,Optional ByVal size As Variant) には，2 つまたは 3 つのパラメータを指定できます（3 つめはオプション）。3 つめのパラメータを指定した場合，FontSize の設定値が再度計算されないことを示します。これは，高解像度（640x480 ～ 1024x768 など）の場合に特に重要です。高解像度の場合，フォームのロード時に計算が実行されるからです。

■ SubSetTCtrlAttr

説明

プロシージャ subSetTFrmlAttr は，パラメータ frm で指定されたフォームのヘッダ／ダイアログ行（typewType によって定義される）の属性 FontName，Font-Size，および FontBold を設定します。

アプリケーションにより，FormLoad 中に，すべてのコントロールについてこのプロシージャを呼び出す必要があります。こうすると，アプリケーションのフォームの表示が一様になり，言語を転換しても正しい FontAttributes が得られるようにすることができます。

プロシージャ subSetTFrmAttr(frm As Form, wType As Integer,Optional ByVal size As Variant) には，2 つまたは 3 つのパラメータを指定できます（3 つめはオプション）。3 つめのパラメータを指定した場合，FontSize の設定値が再度計算されないことを示します。これは，高解像度（640x480 ～ 1024x768 など）の場合に特に重要です。高解像度の場合，フォームのロード時に計算が実行されるからです。

7.8.10 INI ファイル項目の読み取り／書き込み関数

■ AlGetPrivateProfileString

説明

関数 **AlGetPrivateProfileString** によって、INI ファイル項目を読み取ることができます。'IpFileName' の後にファイル名のみを指定した場合、関数は 840DI 固有のファイル構造を想定します。つまり、ディレクトリ user、oem、add_on、mmc2、および Hmi_adv からこのファイル名を持つ INI ファイルの現在の項目を戻します（user ディレクトリからサーチを始めて、最初の検出でサーチを中止します）。
全パスを含むファイル名を入力すると、その INI ファイルのみがサーチされます。

用途

INI ファイル項目の読み取り

構文

Function AlGetPrivateProfileString (

ByVal IpApplicationName As String,
ByVal IpKeyName As String,
ByVal IpDefault As String,
IpReturnedString As String,
ByVal nSize As Long,
ByVal IpFileName As String) As Long

引数

表 7.50 AlGetPrivateProfileString の引数

引数	データタイプ	説明
IpApplicationName	String	セクション名
IpKeyName	String	キー名
IpDefault	String	デフォルト戻り値
IpReturnedString	String	戻される項目
nSize	Long	IpReturnedString のサイズ
IpFileName	String	INI ファイル名

戻り値

文字数が戻される

C++ での構文

```
long stdcall ALGetPrivateProfileString (
                                        LPCTSTR lpApplicationName,
                                        LPCTSTR lpKeyName,
                                        LPCTSTR lpDefault,
                                        LPCTSTR lpReturnedString,
                                        long nSize,
                                        LPCTSTR lpFileName);
```

■ AlGetPrivateProfileInt

説明

関数 **AlGetPrivateProfileInt** によって、INI ファイルの数字項目を読み取ることができます。'IpFileName' の後にファイル名のみを指定した場合、関数は 840DI 固有のファイル構造を想定します。つまり、ディレクトリ user、oem、add_on、mmc2、および Hmi_adv からこのファイル名を持つ INI ファイルの現在の項目を戻します（user ディレクトリからサーチを始めて、最初の検出でサーチを中止します)。
全パスを含むファイル名を入力すると、その INI ファイルのみがサーチされます。

用途

INI ファイルの数字項目の読み取り

構文

Function AlGetPrivateProfileInt (

ByVal lpApplicationName As String,
ByVal lpKeyName As String,
ByVal nDefault As Integer,
ByVal lpFileName As String) As Integer

引数

表 7.51 AlGetPrivateProfileInt の引数

引数	データタイプ	説明
lpApplicationName	String	セクション名
lpKeyName	String	キー名
nDefault	Integer	デフォルト戻り値
lpFileName	String	INI ファイル名

戻り値

指定した INI ファイルの中の、指定したキー名の後に続く文字列の数値。キーが見つからない場合は、デフォルト値が戻されます。値が '0' 未満の場合は '0' が戻されます。

C++ での構文

UINT long stdcall ALGetPrivateProfileInt (

LPCTSTR lpApplicationName,
LPCTSTR lpKeyName,
INT nDefault,
LPCTSTR lpFileName);

■ AlWritePrivateProfileString

説明

関数 **AlWritePrivateProfileString** によって、INI ファイルに項目を書き込むことができます。'IpFileName' の後にファイル名のみを指定した場合、関数は 840DI 固有のファイル構造を想定します。つまり、ディレクトリ user 内のこの名前を持つ INI ファイルの現在の項目に書き込みます。

全パスを含むファイル名を入力すると、その特定の INI のみで書き込みが実行されます。

用途

INI ファイル項目の書き込み

構文

Function AlWritePrivateProfileString (

ByVal IpApplicationName As String,
ByVal IpKeyName As String,
ByVal IpString As String,
ByVal IpFileName As String) As Long

引数

表 7.52　AlWritePrivateProfileString の引数

引数	データタイプ	説明
IpApplicationName	String	セクション名
IpKeyName	String	キー名
IpString	String	キーの設定値
IpFileName	String	INI ファイル名

戻り値

`'0'（成功した場合）、'0'（失敗した場合）

C++ での構文

BOOL stdcall ALWritePrivateProfileString (

LPCTSTR IpApplicationName,
LPCTSTR IpKeyName,
LPCSTR IpString,
LPCTSTR IpFileName);

7.9 シーケンス制御のグローバル変数

全般

関数やプロシージャの他に，シーケンス制御には，メモリースペースを節約し，効率を上げるグローバル変数が備えられています。

■ g_chNCDDEServiceName

説明

NCDDE サーバのリンクトピックが含まれます。例，NCDDE|NCU840D

■ g_chMBDDEServiceName

説明

アラームサーバのリンクトピックが含まれます。例，mbdde|alarme

■ g_chGlobalProfile

説明

ファイル MMC.INI のパスとファイル名が含まれます。例，c:¥mmc2¥MMC.INI

■ g_chLocalProfile

説明

OEM アプリケーションの初期化ファイルのパスとファイル名が含まれます。例，c:¥oem¥OEMBSP1.INI

■ g_chMMCPATH

説明

システムの MMC2 パスが含まれます。例，c:¥mmc2

■ g_hLanguageLibHandle

説明

現在使用されている言語 DLL にアクセスするためのハンドルが含まれます。

■ g_nAccessLevel

説明

システムの現在のアクセスレベルが含まれます（表 7-9 を参照）。

■ g_nAppRunning

説明

MMC 上で現在アクティブなアプリケーション／タスクの番号が含まれます。

■ g_nHelpInfo

説明

ヘルプが使用可能か（真），使用不可か（偽）を示します。

■ g_chHelpContext

説明

ヘルプファイルのある場所を示します。

8 MMC ⇔ NCK/PLC 間のインタフェース

概説

NCDDE サーバは，MMC 103 と 840DI の間をつなぐインタフェースです。MMC 103 はこのインタフェースを使用して，NC，PLC，およびドライブのすべてのデータにアクセスします。

NCDDE サーバは，アプリケーションの開発者に以下の 3 つの機能を提供します。

- 変数サービス：NC，PLC，およびドライブデータへのアクセス
- ドメインサービス：MMC から NCK へ，またはその逆へのファイルのコピー
- PI サービス：NC のプログラム呼び出しサービスの開始

8.1 全般

アプリケーションと NCDDE サーバとの間の通信は，WINDOWS DDE（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，各 WINDOWS 開発環境で使用できます。

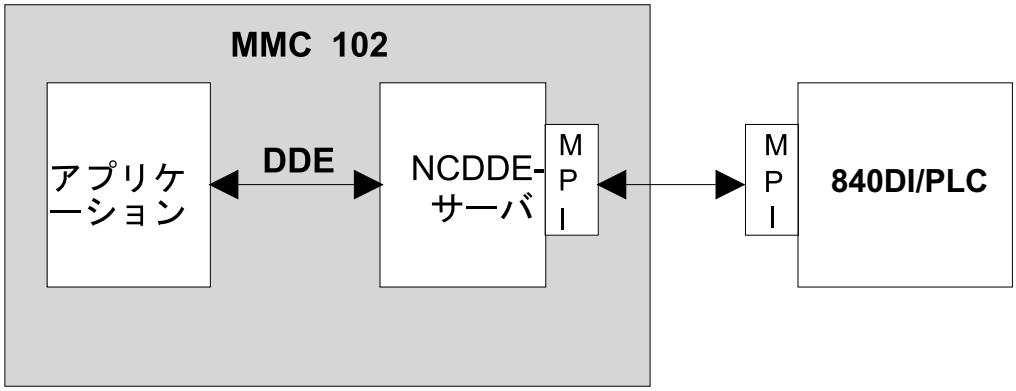


図 8.1 概要

NCDDE サーバは，初期設定ファイルを使用して構成し，特定の開発環境に合うように調整することができます。テストのために使用できるコントロールはあるでしょうか？データにアクセスしたい 1 つまたは複数の NCU がありますか？

注 : Windows 環境を使用するために，NCK との通信には，リアルタイムの動作に制限があります。そのため，インプリメント機能の方を使用してください。これは，OEM パッケージ NCK を使用する NCU において，リアルタイムの反応に直接依存しています。

8.2 DDE の基本

概説

Windows オペレーティングシステムは動的データ交換（DDE）をサポートしています。これにより，アプリケーション開発者は，ある 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 の特長

Windows の DDE には，以下の特長があります。

- DDE は Windows アプリケーション間の通信です。
- DDE はクライアントサーバモデルに従う 2 つのプロセスで実行されます。
- 一方のプロセスがクライアントの役割を果たします。このプロセスはサーバからのデータを要求します。
- もう一方のプロセスがサーバの役割を果たします。このプロセスはクライアントにデータを送ります。
- 接続はクライアントにより確立されます。
- 1 つのプログラムがクライアントと同時にサーバの役割も果たす場合があります。
- 通信は内部 Windows プロトコルに従って指定されます。

DDE 接続の確立

DDE サーバへの接続を確立するには，クライアント開発者は以下の用語に精通していなければなりません。

- Link Server：DDE サーバの名前
- Link Topic：主題
- Link Item：アクセスされるデータ項目
- Link Mode：接続のタイプ

DDE のリンクモード

- Request：クライアントは 1 回だけデータを照会します。
- Warmlink：サーバは変更されたデータについてクライアントに知らせます。その後，クライアントはそのデータ項目にアクセスできます。
- Hotlink：データが変更された場合，サーバは自動的に現在のデータ値をクライアントに送ります。
- Poke：データ項目を書き込むようにクライアントがサーバに指示します。
- Execute：コマンドを実行するようにクライアントがサーバに指示します。

注 : DDE 接続の詳細については，以下の文献を参照するようにお勧めします。
Microsoft Press: Charles Petzold, Programmierung unterWINDOWS 3.1
Microsoft Press: Charles Petzold, WINDOWS 95 Programmierung

8.3 NCDDE サーバの構成

8.3.1 初期設定ファイル MMC.INI

説明

NCDDEサーバの初期設定は, ファイルMMC.INIのセクション[GLOBAL]を使用して実行されます。
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。ここでは，Link Server および Link Topic を定義します。ローカルの NCDDE サーバはこれらを使用して，接続を確立しなければなりません。
Link Server と Link Topic という用語については，8.2 章で説明されています。

ファイル MMC.INI のセクション [GLOBAL] 内の 4 つの行を編集することにより，NCDDE サーバを以下の 4 つの基本的な方法で構成することができます。

- 1 つの NC に対する接続を確立する（デフォルト）
- 1 つまたは複数の NC に対する接続を確立する（M:N - 機能については 8.3.3 章を参照）。
- PC 上のローカル操作モード。
  開発者は，NC に接続しなくても，自分の PC 上でローカルに自分のアプリケーションをテストできます。この場合のために NCDDE サーバには置換値が備えられています。この値はコマンド "NEW"（8.8）を使用して定義し，コマンド "ANIMATE"（8.8）を使用して変更を加えて，アクティブな NC をシミュレートできます。
- NC シミュレータを使用した PC 上のローカル操作モード。開発者は，NC に接続しなくても自分の PC 上で自分のアプリケーションをローカルにテストできます。NC シミュレータを使用すると，本物の CNC に極めて近い動作を実現できます。

注 : これらの 4 つの基本構成のうち，アクティブにできるのは 1 つだけです。

注 : 行の先頭のセミコロン（;）は，その行がコメントであることを示しています。

NcddeServiceName

NCDDE サーバの DDE リンクサービス名。デフォルト名は常に "ncdde" です。

注 : 8 章のすべての例では，以下のことを前提としています。
”NcddeServiceName = ncdde"
これ以外の場合，例を変更して，使用できるようにする必要があります。

Ncdde-MachineName

ここには，標準アプリケーションの NCU 名が入力されます。
ここに "MachineSwitch" が入力されていれば，複数の NCU の間で切り替えることができます（M:N 機能については，8.8.3 を参照）。

NcddeDefault-MachineName

これは，M:N 機能を初期設定します。つまり，この NCU は MMC のスタートアップ時に接続されます。

Ncdde-MachineNames

ここには，接続できる NCU の名前が入力されます。ここに入力する各 NCU 名は，ファイル MMC.INI に同じ名前のセクションがなければなりません。

NCDDE サーバの開始時にロードされる NSK ファイル（8.3.2 章）。ここには，変数定義が入れられます。

この属性は，ファイル MMC.INI のセクション [GLOBAL] に指定し，インストールされている NC シミュレータの特徴を示します。NC シミュレータがインストールされていない場合は，このエントリには意味がありません。

以下の例では，NC および NC シミュレータを使用せずに，PC 上にインストールする場合のファイル MMC.INI の設定を示しています。

■ 例 8-1 ファイル MMC.INI の抜粋

```
[GLOBAL]
; for using M:N function set NcddeMachineName=MachineSwitch
; for working without NC set NcddeMachineName=local
; for working with SIMNC set NcddeMachineName=SIM1
; for connecting to a NC set NcddeMachineName=NCU840D
NcddeMachineName=local
; for using M:N function set NcddeDefaultMachineName=net:NCU_1
; for working without NC set NcddeDefaultMachineName=local
; for working with SIMNC set NcddeDefaultMachineName=SIM1
; for connecting to a NC set NcddeDefaultMachineName=NCU840D
NcddeDefaultMachineName=local
; for using M:N function set NcddeMachineNames=net,NCU840D
; for working without NC set NcddeMachineNames=
; for working with SIMNC set NcddeMachineNames=SIM1
; for connecting to a NC set NcddeMachineNames=NCU840D
NcddeMachineNames=
; for using M:N function set NcddeStartupFile=ncdde5.nsk
; for working without NC set NcddeStartupFile=ncdde202.nsk
; for working with SIMNC set NcddeStartupFile=sim1dde5.nsk
; for connecting to a NC set NcddeStartupFile=ncdde5.nsk
NcddeStartupFile=ncdde202.nsk
```

ネームスペース

LOCAL モードでは，NCDDE サーバは変数の ' ネームスペース ' を識別しません。' ネームスペース ' は，TOPIC 間の区別に使用されるものです。TOPIC LOCAL 用の変数が作成され，TOPIC Sim0 にも同じ変数が作成された場合，NCDDE では両者を区別しません。このことにより，領域アプリケーション MACHINE で現行のブロック表示による画面が選択されていると，シミュレーションモードでは現行のブロック表示が使用できないなどの影響が生じることがあります。この場合，プログラムにより ' シミュレーション変数 ‘ を上書き定義するローカル変数が作成されます。

8.3.2 NCDDE サーバのコマンドファイル

NSK ファイル

コマンドファイル（拡張子 NSK）には，NCDDE 接続で参照される Link-Item などが含まれています。これらのファイルには，8.8 章で説明されているコマンドを含めることができます。

これらのファイルでは，アクセスできるデータ（Link Item）を記述することができます。さらに，CALL 命令を使用して NSK ファイルを含めることもできます。これにより，構造化が可能です。例 8-2 では，Link Item（LastError）と，CALL 命令を使用した MMC 103 のグローバル変数の構造化を示しています。

注 : CALL 命令で独自の NSK ファイルをロードすることができます。MAP 機能（8.6.3 章）を使用して，NSK ファイルを作成することができます。

■ 例 8-2 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 STATELINK("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
```

8.3.3 複数の NC への接続

M :N 機能

この機能を使用すれば，複数の MMC を複数の NCU に接続することができます。たとえば，2 つの NCU に入っているデータに，1 つの MMC 103 からアクセスすることができます。ファイル NETNAMES.INI（ファイル MMC.INI のセクション [net] で定義されている）が解釈されて，この基本構成に使用されます。

接続先

セクション [conn MMC_1] では，MMC が接続できるパートナを指定します。

ネットワークパラメータ

セクション [param network] では，以下のようにバス構成に応じて，転送速度が設定されます。

- BTSS 1,5 Mbit
- MPI 187,5 Kbit

バス参加者

セクション [param NCU_n] では，NCU 名のほかに，NC および PLC のバスアドレスが設定されます。MMC103 は NCU をアドレッシングする場合に，これらの名前を使用しなければなりません。記述は NCU ごとに行わなければなりません。

■ 例 8-3 ファイル NETNAMES.INI

```
; Owner TECHNICAL reference to the bus addresses
; computer-specific
[own]
owner= MMC_1
; Description of possible connections
[conn MMC_1]
conn_1= NCU_1
conn_2= NCU_2
; Description of significant net-parameters
; btss =1,5Mbit
; mpi =187,5 Kbit
[param network]
bus= btss
; Bus addresses for all bus participants
[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_Maschine
```

8.4 DDE 接続の確立

概説

この項では，Visual Basic および Visual C++ を使用して，NCDDE サーバへの DDE 接続を確立する方法について説明します。

注 : 以降の例では，DDE 接続が使用するのは，標準 Visual Basic コントロール（VBX）の "LABEL" だけです。ただし，OEM アプリケーションは OEM Visual Basic コントロール（たとえば，当社が提供している DCTL など）を，DDE 通信に使用しなければなりません。

以降の例を使用するには，以下の要件を満たしていなければなりません。

開発環境

- MS Visual Basic 4.0_16 を使用するようにお勧めします。
- 例をテストするための，PC と 840DI の間の MPI 通信，
  さらに NC 操作のための NCDDE サーバの構成
  840DI を使わずに NCDDE サーバを使用した場合，アクセスできないデータもあります。
- NCDDE サーバ（C:¥MMC2¥NCDDE.EXE）を開始していなければなりません（たとえば，エクスプローラーや［スタート］メニューを使用して）。

8.4.1 Visual Basic による DDE 接続の確立

DDE クライアント接続を確立できる標準 Visual Basic コントロール（VBX）の例を以下に示します。

- ラベル
- テキストボックス
- ピクチャ

Link - Service と Link - Topic は，プロパティ（属性）"LinkTopic" に結合されています。これらはパイプシンボル " | " によって区切られています（たとえば，LinkTopic = "ncdde | local"）。

1 度だけの変数の読み取り

以下の例では，ワーク座標系を参照する最初のチャネル内の最初の軸の実際の位置を 1 度だけ読み取ります。以下の例では，NCDDE サーバをローカル操作モード用に構成し，NcddeServiceName を ncdde に設定することが必要です。つまりこの場合，NCK はアクセスされません。このタイプの読み取りの場合，LinkMode は 2 に設定されていなければなりません。

注 : 値を 1 度だけ読み取る場合，LinkMode を 2 に設定します。それから，最初のチャネルはメソッド "Link Request" を使用して，値を要求します。

■ 例 8-4 1 度だけの変数の読み取り

```
Sub Form_Load ()
Label1.LinkTopic = "ncdde|local"
```

```
Label1.LinkItem="/Channel/GeometricAxis/actToolBasePos[u1,1]"
Label1.LinkMode = 2
Label1.LinkRequest
End Sub
```

注 : チャネル ID "u1" が指定されていない場合，デフォルトで最初のチャネルにアクセスします。

変更時の更新

以下の例では，"label1" の機械座標系を参照する 2 番目のチャネル内の 3 番目の軸の実際の位置を，自動的に更新（ホットリンク）します。つまり，この軸の現在の実際の位置が表示されます。

注 : ホットリンクの場合，"LinkMode" は 1 に設定しなければなりません。

■ 例 8-5 変更時の更新

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

変更時の通知

この例では，最初の PLC バイトが変更されたときに，NCDDE サーバはアプリケーション／クライアントに通知を行います（Warmlink）。それから，Label1 の "SubLinkNotify" が自動的に実行します。そこでは，データを入手するために，"LinkRequest" を呼び出さなければなりません。そのため，データが表示される前に検査を行い，変更または変換することができます。

注 : 変更時に通知する（Warmlink）場合，"LinkMode" は 3 に設定しなければなりません。

■ 例 8-6 変更時の通知

```
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 データの書き込み

この例では，クライアントは最初のチャネルの最初の R パラメータ R[1] に，値 "4" を書き込みます。

注 : データを書き込む（Poke）場合，"LinkMode" は 2 に設定しなければなりません。LinkPoke がこの値を書き込みます。

■ 例 8-7 NC データの書き込み

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

PLC データの書き込み

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

■ 例 8-8 PLC データの書き込み

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

コマンドの実行

実行されるコマンドについては，?.?.? 章で説明されています。以下の例では，MMC から NCK へのファイル "test.mpf" の転送を開始します。

注 : コマンド実行（Execute）の場合，"LinkMode" は 2 に設定しなければなりません。LinkExecute でコマンドを実行します。

■ 例 8-9 コマンドの実行

```
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
```

8.4.2 Visual C/C++ による DDE 接続の確立

概説

C/C++ は DDE インタフェースのすべての機能を使用できます。特に，DDE インタフェースの非同期呼び出しを使用することができます。これは，DCTL などの OEM Visual Basic コントロールを使用すれば，Visual Basic でも可能です。

注 : C/C++ での DDE を推奨されているのは，Windows での C プログラミングに習熟している OEM ユーザと，OEM パッケージのシーケンス制御を部分的にしか（またはまったく）組み込む必要のない OEM ユーザだけです。

C / C++ での DDE アクセス

この例では，以下の間での Hotlink（Advise）接続（確認付き）を確立する方法を示し

ます。

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

変数の値の変更は，DDEML に公表されているコールバックルーチンにより，XTYP_ADVDATA トランザクションに受け入れられます。

■ 例 8-10 C レベルの Hotlink

```
WORD idInst;// created with DdeInitialize
HSZ hszService, hszTopic, hszItem; // string handles
HCONV hConv; // conversation handle
hszService = DdeCreateStringHandle ( idInst , "ncdde" , NULL );
hszTopic = DdeCreateStringHandle ( idInst , "local" , NULL );
hszItem = DdeCreateStringHandle ( idInst ,
_ "/Channel/GeometricAxis/toolBaseDistToGo[1]" , NULL );
hConv = DdeConnect(idInst,hszService,hszTopic, NULL);
// establishing a connection to the server
// hotlink follows
if ( DdeClientTransaction ( (LPBYTE)NULL ,0 ,hConv , hszItem ,
_ CF_TEXT ,XTYP_ADVSTART|XTYP_ACKREQ , 1000 , NULL )
_ ==TRUE) {...} // hotlink successfully established ...
```

注 : C/C++ DDE プログラミングの詳細については，以下の文献を参照するようにお勧めします。

- Microsoft Press: Charles Petzold, Programmierung WINDOWS 3.1
- Microsoft Press: Charles Petzold, WINDOWS 95 Programmierung

8.4.3 MS Excel からの DDE 接続の確立

概説

Excel で，セル形式を使用する変数用の，NCDDE サーバのインタフェースへの Advise（Hotlink）接続を確立することができます。

EXCEL セル内の構文

セル内で以下の定義が必要です。

= NcddeServiceName|NcddeMachineName!Variables

Excel 内の PLC ビットの表示

以下の例では，Excel（ドイツ語版）のセルから，データブロック 100 のバイト 9 の中の 3 番目のビットへの，Advise（Hotlink）接続が示されています。

変数の名前は，"/Plc/DataBlock/Bit[c100,9.3]" です。NCDDE サーバが接続するマシンの名前は "ncu840D" です。

■ 例 8-11 MS Excel での PLC ビットの表示

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

1	1

セル計算式をご覧ください。右側には結果が示され，継続的に表示が更新されます。

8.5 変数サービス

概説

NCDDE サーバの変数サービスでは，以下の 2 つの種類のデータアクセスが提供されています。

- 単一変数アクセス
- 配列変数アクセス

必要であれば，Link Item で，追加データフォーマットおよび配列の範囲により，変数を指定することができます。これにより，NCDDE サーバからのデータを要求することができます（ほとんどの場合，それ以上の変換を必要ありません）。

注 : 11 章「参照データ」および変数のオンラインヘルプで，アクセスできる変数について詳しく説明されています。

NCDDE 変数のフォーマット

NCDDE 変数のフォーマット命令が，Link Item の最後に追加されています。データの内部準備により，タイプが整数，浮動小数点数，およびテキストのフォーマットを設定することができます。

フォーマットは，高水準プログラミング言語である C の，拡張された ‘printf’ フォーマットとして指定されます。NCDDE フォーマットの構文を，以下に示します。

フォーマット : "" < 引数 > <‘printf’ フォーマット >
引数 : ‘!’ ‘b’ < 引数 > // ビットストリングに変換
‘!’ ‘d’ < 引数 > // d（64 ビット浮動小数点数としての double）
‘!’ ‘l’ < 引数 > // l（32 ビット整数としての long）
‘!’ ‘t’ < 引数 > // t（ストリングとしてのテキスト）
!’ ‘#’ < 引数 > // #（32 ビット整数としての
// 変数アクセスの指標）

対応する DDE 変数のデータタイプについては，11 章または "NCDDE 変数 " ヘルプを参照してください。

注 : データ選択のタイプと実際に読み取られる変数のタイプが異なる場合，データ形式の変換が自動で実行されることはありません。つまり，間違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，2 番目の軸の実際の位置が NC から読み取られ，整数部分は最大 11 桁，小数部分は最大 3 桁で表示されます。フォーマット設定を行わないと，小数点以下 3 桁のみになります。

■ 例 8-12 整数部分最大 11 桁，小数部分最大 3 桁のフォーマット設定

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

16 進数形式への変換

この例では，フラグバイト 5 が PLC から読み取られてから，先行ゼロの後に表示されます。

■ 例 8-13 16 進形式への変換

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

ビットストリングへの変換

この例では，フラグバイト 5 が読み取られてから，32 ビットストリングとして表示されます。

■ 例 8-14 ビットストリングへの変換

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

結果 : 10101010101010101

PLC からのストリングの読み取り

この例では，バイト 20 から始まるデータモジュール 81 から 10 バイトが読み取られ，最後が 0 になるストリングとして表示されます。

■ 例 8-15 PLC からのストリングの読み取り

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

結果 : 例，Hello

8.5.1 単一変数アクセス

シーケンス制御（7 章を参照）で作業している場合，グローバル変数

"g_chNCDDEServiceName" を LinkTopic として使用できるという利点があります。この変数には，ファイル MMC.INI 内に入力されている，NCDDEServiceName，および NcddeMachineName が入れられます。それらはパイプシンボル（"|"）で区切られています。

3 つの変数への単一アクセス

以下の例では，最初の 3 つの形状軸の名前が読み取られます。

■ 例 8-16 3 つの変数への単一アクセス

```
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
chsname(2).LinkItem = "/Channel/MachineAxis/name[3]"
achsname(2).LinkMode = 2
achsname(2).LinkRequest
End Sub
```

PLC ビットアクセス

以下の Link Item を使用すれば，入力バイト 2 のビット 4 にアクセスできます。

/Plc/Input/Bit[2.4]

PLC バイトアクセス

以下の Link Item を使用すれば，出力バイト 4 にアクセスできます。

/Plc/Output/Byte[4]

PLC ワードアクセス

以下の Link Item を使用すれば，レジスターワード 4 にアクセスできます。

/Plc/Memory/Word[8]

他の変数にアクセスする方法については，11.1.5 章で説明されています。

8.5.2 配列変数アクセス

用途

配列変数は，領域から複数のデータを読み取る場合に便利です。そのため，単一変数の複数アクセスと比べて，NCDDE サーバの計算負荷が軽減されます。例 8-16 はこの点に関しては悪い例といえます。

ヒント：配列アクセスは，通信に必要な時間が大幅に軽減されるので，データアクセスのスピードだけでなく，システム全体のスピードも上がります。

構文

はじめに，配列の領域の構文を簡単に示します。

Variable name[c, u, StartIndex, [EndIndex]]

パラメータ

表 8.1 配列データのアクセス用のパラメータ

名前	説明
Variable name	NCK/PLC 変数の名前（11 章を参照）
c	NCK 変数にアクセスする（11 章を参照）場合，列指標 c は列を表し，多次元配列のみに適用されます。PLC 変数にアクセスする場合，c はアクセスされるデータモジュールを特定します。
u	チャネルなどの装置の指標（NCK 変数の場合のみ）:u は装置を表します。
StartIndex	読み取られる変数の指標。配列へアクセスする場合に，読み取られる最初の値をここで指定します。
EndIndex（オプション）	（配列アクセスの場合のみ）読み取られる値の番号を指定します。

軸名配列へのアクセス

以下の例では，NCK から最初の 3 つの軸名が読み取られます。結果は，それらの軸名を含んでいるストリング（たとえば，X1，Y1，Z1 を含んでいる "X1Y1Z1" というフォーマット）になります。Visual Basic の関数 "Trim$" および "Mid$" を使用すれば，この結果ストリングから軸名を取り出すことができます。

■ 例 8-17 軸名配列へのアクセス

```
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
'extracting single names from the text array
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))
```

軸名配列へのアクセス

以下の例では，軸 3 で始まる，2 番目のチャネルにある 2 つの軸の軸名が読み取られます。軸 3 および軸 4 の名前が読み取られます。

以下の行以外は，例 8-17 と同じです。

■ 例 8-18 チャネル 2 内の軸名配列へのアクセス

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

PLC 配列データへのアクセス

以下の例では，バイト 2 から始まる DB 8 の 3 つのバイト（バイト 2，3，および 4）を，16 進形式の 2 桁の番号として PLC から読み取ります。それから，Visual Basic の関数 ”Trim$” および ” Mid$” を使用して，そのバイトを抽出します。

■ 例 8-19 PLC 配列データへのアクセス

```
Label1.LinkTopic = "ncdde|ncu840d"
Label1.LinkItem = "/PLC/Datablock/Byte[c8,2,4](""!I%02Ix"")"
Label1.LinkMode = 1 'hotlink
'extracting single bytes in hexadecimal format
byte_1 = Trim$(Mid$(Label1.Caption,1,2))
byte_2 = Trim$(Mid$(Label1.Caption,3,2))
byte_3 = Trim$(Mid$(Label1.Caption,5,2))
```

番号を指定した PLC アクセス

以下の例では，ワード 2 から始まる DB 8 の 5 つのワードを，4 桁の 16 進数として PLC から読み取ります。ワードは "_" で区切られます。

■ 例 8-20 指定された番号の PLC 配列データの読み取り

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

R パラメータ配列へのアクセス

この例では，以下の値を 3 つの R パラメータ（R3,
R4，および R5）に書き込みます。
R3 = 2.2
R4 = 3.5
R5 = 4.9

■ 例 8-21 R パラメータ配列へのアクセス

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

8.6 ファイル転送サービス（ドメインサービス）

概説

ドメインサービスを使用して，領域（ドメイン）MMC および NCK ／ PLC 間でファイルを転送できます。

MMC と NCK ／ PLC の間のファイル転送に使用できるコマンドは，全部で 5 つあります（表 8.2）。ファイル転送はバックグラウンドで実行されます。

SW リリース 3.3 以降，領域間の拡張コピー機能を使用できるようになりました。この機能は，特に NC でのプログラム編集に適しています。8.6.2 に，この新しい機能の詳細が説明されています。

表 8.2 ドメインサービスのコマンド

コマンド	説明
COPY_TO_NC	MMC から NCK へのコピー，追加情報を含む
COPY_TO_NC_BINARY	MMC から NCK へのコピー，追加情報を含まない
COPY_FROM_NC	NCK から MMC へのコピー，追加情報を含む
COPY_FROM_NC_BINARY	NCK から MMC へのコピー，追加情報を含まない
MAP_ACC_NC	ACC ファイルの NCK からのロード, および DDE インタフェースでの使用準備

特殊な状態変数を使用して，データ転送の状態をモニタできます。

8.6.1 MMC と NC ／ PLC の間のデータ転送

説明

この機能を使用して，MMC と NC ／ PLC の間でデータ／データファイルを転送できます。

用途

パートプログラムや工具データを NC に転送する場合や，S7 プログラムや C プログラムを PLC に転送する場合などに，この機能を使用できます。

この機能に拡張子 "BINARY" を指定しないと，パートプログラムなどのファイルを NC に転送できます。NCDDE サーバにより，ブロックヘッダがデータに追加されます。このヘッダには，ブロックのサイズと日付，および NCK ファイルシステムのパスが含まれています。

注 : NC へのデータ転送に使用します。

注 : 必ず NC ブロックヘッダがデータストリームに追加されるので，PLC へのファイル転送には使用できません。

BINARY 機能

この機能に拡張子 "BINARY" を指定すると，パートプログラムなどのファイルを NC に転送できるだけでなく，PLC モジュールを PLC に転送することもできます。

NCDDE サーバにより，NC ブロックヘッダを追加せずにファイルが転送されます。

注 : PLC および NC へのファイル転送に使用できます。

注 : PLC モジュールは，必ず PLC の受動ファイルシステムにコピーされます。この時点ではまだモジュールはアクティブではありません。受動モジュールをアクティブにする必要は依然としてあります（例 8-33 と比較）。

構文

コピー機能は，以下の構文に従ったストリングとして記述する必要があります。

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)

引数

表 8.3 コマンド COPY_TO ／ FROM_NC（_BINARY）の引数

名前	説明
WinFile	MMC 領域内の情報のソースまたは宛先
NcFile	NCK ／ PLC 環境のファイル名
TransferState	転送状態の特徴を表す変数

引数 WinFile

MMC 側の情報のソースと宛先を記述します。先頭文字でタイプを指定します。

このパラメータは，WINDOWS 環境のデフォルトのファイル名になります。このパラメータには，ドライブの指定，パス，およびファイル名が含まれている必要があります。たとえば，"C:¥NCest.MPF" のようになります。

引数 WinFile によるパイピング

WinFile の先頭文字が @ 文字の場合，引数はパイプ名として解釈されます。
COPY_TO_NC 機能と一緒に，‘ パイプを介したコピー ’ サービスを実行できます。

注 : 最大 500 バイトまでのサイズのブロックの読み取りや書き込みに適しています。このサイズより大きいブロックは NCDDE サーバにより拒否されます。

NCK ／ PLC への転送（ダウンロード）時には，DDE-poke がパイプラインを埋めるので，NCK ／ PLC への直接転送を実行できるようになります。Poke が空の場合は，転送が終了したことを示します（例 8-27 を参照）。

NCK ／ PLC からの転送（アップロード）時には，DDE-request によりパイプラインは空にされ，転送が進行するにつれて埋められることになります。

引数 WinFile による共用メモリアクセス

要求によって空のデータが戻された場合は，転送が終了したことを示します。WinFile の先頭文字が # 文字で，その後に 16 進数形式の数値が続いている場合は，グローバルヒープに割り当てられた WINDOWS 共用メモリとして解釈されます。

WINDOWS の GlobalAlloc 機能により割り振られたメモリは，以下の構造で初期設定しなければなりません。このヘッダの後に，使用可能データが付加されます。次の例は，Visual Basic での使用例を示しています。

■ 例 8-22

```
struct NCDDE_DOMAINMAP_HEADER {
unsigned short handle; // buffer handle (HGLOBAL) (is preset by the client)
unsigned short header_size; // header length (is preset by the client)
unsigned long shared_size; // usable length of the data area
// (is preset by the client)
unsigned long fill_count; // number of valid bytes in the data area
// (is preset by the client during download
// and set by the Server during upload)
unsigned long state; // corresponds to the transfer state variable of
// the transfer command
// < 100: transfer is running,
// "state" approximately reflects the percentage of
// the file, that has already been transferred
// ==100: transfer successfully terminated
// > 100: transfer was stopped with error,
// "state" shows the NCDDE error code
// (is set by the server)
unsigned long file_mod_time; // file modification time
// (is preset by the client during download
// and set by the server during upload)
unsigned long server_private; // server-specific data (is set by the server)
unsigned long client_private; // client-specific data (is set by the client)
unsigned long magic; // signature for an additional type check
// the value is always NCDDE_MAGIC = 0xF6F7F8F9
// (preset by the client)
};
```

引数 NcFile

引数 "NcFile" は NCK ／ PLC 環境のパス名です。この名前は構成可能な部分（関係する CNC のアドレッシングに必要）と，CNC 環境のドメインパスから成り立っています。

NCK のドメインは, NCDDE サーバによって NC ファイル名を使用してアドレッシングされます。

/NCPLC か NC のどちらかの領域

/_N_MPF_DIRNC のパスの指定

/_N_WS03_MPF ファイル名

引数 TransferState

引数 TransferState は，バックグラウンドで実行される転送の状態を戻すのに使用される，サーバのローカル変数の名前です（変数タイプ : 固定）。この引数を指定しないと，この変数はサーバから作成されます。

変数 TransferState は，ファイル転送の状態を特定表します。

表 8.4 転送状態の特定

転送状態	数値	意味
転送の開始	0	CNC に対する開始プロトコルが処理されています。
転送を実行中	1 ～ 98	転送を実行中です。すでに転送済みのファイルのおおよそのパーセント比率が，この数値により示されます（注を参照）。
転送の終了	99	CNC に対する終了プロトコルが処理されています。
転送の成功	100	エラーが起きずにジョブが実行されました。
エラーコードで転送停止	100 より大きい	転送は停止しました。TransferState に，報告されたエラーコード（11.7）が含まれています。

値の範囲は決められているので，100 以下の値は通常の状態を示し，100 より大きい値はすべてエラー状態を示しています。

注 : 変数の値が 0 ～ 99 の間（過渡状態）にあるうちは，その変数を使用してさらにファイル転送を実行することはありません。

ファイル転送の停止

実行中のファイル転送を停止するには，適切なエラーコードで転送変数を上書きしなければなりません。つまり，"LONG"（4 バイト）として定義されている転送変数の個々のバイトの値を，0 以外にしなければなりません。

適切なエラーコードの例 : 16909060

視覚化

転送状態を視覚化する場合に，たとえば Advise ／ Hotlink 接続を介したバー表示などに TransferState を使用できます。

注 : BINARY モードでの転送の際やパイプを使用した転送の際には，ファイルサイズに関する情報は利用できません。したがって，変数 TransferState を使用しても現行の転送データのパーセント比率は示されません。常に 50% として示されます。

ファイルが非常に小さい場合，表示が 1 から 99 にジャンプすることがあります。これは Hotlink の本質的な問題で，この問題があるため短時間に NCDDE サーバからクライアント／アプリケーションへのデータ呼び出しを完全に実行することはできません。

パートプログラムのアップロード

以下の例では，パートプログラム "BSP.MPF" をディレクトリ "C:¥NC" 内のファイル "test.mpf" にコピーします。ファイル "test.mpf" が新しく作成されます。パートプログラム "BSP.MPF" が NCK 内になければなりません。

■ 例 8-23 パートプログラムのアップロード

```
Sub Form_Load ()
Label1.LinkTopic = "NCDDE|ncu840d"
Label1.LinkMode = 2
Label1.LinkExecute "COPY_FROM_NC(""C:¥NCest.MPF"",
_ /NC/_N_MPF_DIR/_N_BSP_MPF,trans)"
End Sub
```

共用メモリアクセス

以下の例（例 8-24 と例 8-25）では，パートプログラム "TEST.MPF" を共用メモリ領域にコピーし，この領域から "String" タイプの Visual Basic 変数 FILE$ にコピーします。Memoryread および Memorywrite 機能を使用するだけで，VB からこれらのデータにアクセスできます。C++ を使用する方がこの機能をより活用できます。

注 : Visual Basic から初期化を実行する際には，Intel 形式（つまり高位バイトと低位バイトが逆になっている）の NCDDE_DOMAINMAP_HEADER 構造内のストリングにエレメントをすべて書き込まなければなりません。

■ 例 8-24 共用メモリその 1: " 一般的な宣言 "

```
'WINDOWS SDK functions
Declare Function GlobalAlloc Lib "Kernel" (ByVal wFlags As Integer, ByVal dwBytes
AsLong) 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
AsAny, ByVal dwcb&) As Long
Declare Function MemoryRead Lib "toolhelp.dll" (ByVal wSel%, ByVal dwOffset&, lpvBuf
AsAny, ByVal dwcb&) As Long
Const HeaderSize = 32
```

■ 例 8-25 共用メモリその 2

```
Sub Form_Load ()
Dim ncdde_global_memory$
Static byte(1 To HeaderSize) As Integer
Dim ptrToBuffer, ergPtr As Long
LenOfFile$ = Space$(5)
'Hotlink, so the NC connection is established when Copy_FROM_NC is called
Label2.LinkTopic = "NCDDE|NCU840D"
Label2.LinkItem = "/Bag/State/resetActive"
Label2.LinkMode = 1
' allocating the shared memory
handleglobal = GlobalAlloc(&H2102, 1024)
ptrToBuffer = GlobalLock(handleglobal)
'build up NCDDE_DOMAINMAP_HEADER
'handle
byte(1) = handleglobal Mod 256
byte(2) = handleglobal ¥ 256
'header length
byte(3) = HeaderSize
byte(4) = 0
'number of usable data area
byte(5) = &HE0
byte(6) = &H3
byte(7) = 0
byte(8) = 0
'initialize the next 20 bytes with 0
for i=9 to 28
byte(i) = 0
next I
'initialize the NCDDE-Magic also
byte(29) = &HF9
byte(30) = &HF8
byte(31) = &HF7
byte(32) = &HF6
'build up string
For i =1 ToHeaderSize
ncdde_global_memory$ = 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$(byte(1)) +
_ ",/NC/_N_MPF_DIR/_N_TEST_MPF,trans)"
Label1.LinkExecute execCommand
'reads data area from Shared Memory
nBytes& = MemoryRead(GlobalHandleToSel(handleglobal),8, ByVal LenOfFile$, 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
```

パートプログラムのダウンロード

以下の例では，"test.mpf" という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" にコピーします。NC ではパートプログラムの名前は "BSP.MPF" です。

■ 例 8-26 パートプログラムのダウンロード

```
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
```

パイプを使用したパートプログラムのダウンロード

以下のファイルでは，パイプメカニズムのアプリケーションを示します。ファイル PIPE1.MPF が NC で作成され，NC ブロック￥GO1F11111 x5555￥”がそのファイルに書き込まれます。

■ 例 8-27 パイプを使用したパートプログラムのダウンロード

```
Sub Form _Load ()
'start Pipe
Label1.LinkTopic =  “NCDDE|ncu840d“
Label1.LinkMode = 2
Label1.LinkExecute  “COPY_TO_NC ( @pipe
/NC/_N_MPF_DIR/_N_PIPE1_MPF,trans ) “
’describe Pipe
Label2.LinkTopic =  “NCDDE|NCU840D“
Label2.LinkMode = 2
Label2.LinkItem =  “@pipe“
Label2.Caption = "G01 F11111 X5555"
Label2.LinkPoke
’finish Pipe
Label2.Caption = ""
Label2.LinkPoke
End Sub
```

S7 モジュールの PLC へのダウンロード

モジュール "OB1.PLC" を PLC の受動ファイルシステムに転送します。

注 : PLC モジュールは，必ず PLC の受動ファイルシステムにコピーされます。この時点ではまだモジュールはアクティブではありません。受動モジュールをアクティブにする必要は依然としてあります（例 8-33 を参照）。

■ 例 8-28 S7 モジュールの PLC へのダウンロード

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

8.6.2 MMC と NC ／ PLC の間の拡張コピー機能

説明

この機能を使用すると，NC ／ PLC と MMC の間でファイルを転送できます。

用途

この機能は，シングルブロックのパートプログラムのセクションを転送する場合や，NC に格納されているパートプログラムを編集する場合に特に適しています。

注 : 正規の機能と "BINARY" を付けて変形した機能の違いについては，8.6.1 章で説明されています。

構文

拡張コピー機能は，以下の構文に従ったストリングとして記述する必要があります。

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)

引数

表 8.5 に引数の説明があります。

表 8.5 COPY_TO/FROM_NC コマンドの引数

名前	説明
WinFile	MMC 領域の情報のソースまたは宛先
NcFile	NC ／ PLC 環境のファイル名
seekPos	シークポインタ : コピープロシージャの始点。識別子はブロックの場合は B，文字の場合は C。
seekLen	ウィンドウサイズ : 転送される領域。識別子はブロックの場合は B，文字の場合は C。
compareString	検索ストリング。最大長は 32 文字。
skipCount	検出された検索ストリングをスキップする回数。

コマンドのサブコマンドがすべて完全に処理されると，そのコマンドは戻ります。このコマンドの実行中にエラーが起きたら，変数 LastError を使用してそのエラーを分析できます。

以下の例では，この新しいコマンドの代表的な用途を示します。

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" の先頭 1024 バイトをディレクトリ "C:¥NC" のファイル "test.dat" にファイル転送します。

■ 例 8-29 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:¥NCest.dat,
_ /NC/_N_MPF_DIR/_N_TP1_MPF,1 ,1024 , ,0 )"
End Sub
```

シングルブロックの転送

ブロック 2 ～ 4 をパートプログラム X.MPF にパイプ転送します。既存のブロックは上書きされます。

■ 例 8-30 シングルブロックの転送

```
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
```

1 つのブロックの転送

テキストをパートプログラムの 2 番目のブロックに転送します（テキストの最大長は 200 バイト）。2 番目のブロックは上書きされます。

■ 例 8-31 ブロックの転送

```
Sub Form_Load ()
Label1.LinkTopic = "NCDDE|NCU840D"
Label1.LinkMode = 2
Label1.LinkExecute "COPY_TO_NC ( ""!This is the content of the
_ 2nd block"" , /NC/_N_MPF_DIR/_N_TEST_MPF, B2 ,
_ 1 , ,0 )"
End Sub
```

8.6.3 メイン間の MAP 機能

■ MAP_ACC_NC

説明

この機能を使用すると，グローバルユーザデータ（GUD）と NCK マシンデータを NCDDE サーバに通知できます。これらのデータは拡張子が ACC のファイルに格納されます。これらのファイルは NCK にあり，変数のアクセス記述が含まれています。

用途

コマンド MAP_ACC_NC を使用すると，ACC ファイルを NCK から読み取り，DDE インタフェースで使用するために準備できます。つまり，これらのファイルに対応する接続が NCDDE サーバに作成されます。

ヒント : ユーザが NCDDE サーバに新しい NCK データを通知できるようにしてください。そうしないと，この種の変数／データにアクセスできません。

このコマンドは，拡張子を指定したコマンド COPY_FROM_NC と同じ働きをします。

ACC ファイルからの情報はデコードされ，DDE インタフェースに渡せる状態にされます。

構文

この呼び出しは以下の構文に従います。

MAP_ACC_NC_ (WinFile, NcFile, TransferState, Area, DataBlock, Timeout, Prefix)

引数

表 8.6 には，引数に関する詳細が記述されています。最初の 3 つの引数は，他のドメインサービスの引数と同じです（8.6.1 を参照）。概要を示すために以下の表に含められています。

表 8.6 コマンド MAP_ACC_NC の引数

名前	説明
WinFile	MMC 領域の情報のソースまたは宛先
NcFile	NC ／ PLC 環境のファイル名
TransferState	転送の状態を特定する変数
Area	11.1.1 章の表 11-1 で説明されている，ACC データの領域アドレス 。ここに，完全なリストを示します。 領域　領域アドレス NCK　0 モードグループ　1 チャネル　2 軸　3 ツールオフセット　4 送りドライブ　5 メインの主軸ドライブ　6 予約済み　7
DataBlock	変数サービスのモジュール形 : 11.3.1 章で説明されている，範囲 00 ～ FF の 16 進数。以下はその例（抜粋）です。 モジュール識別　モジュール番号 システム状態データ（Y）　10 グローバルユーザデータ（GUD）　17 OEM ツールデータ（TU）　24 マガジンディレクトリ（TMV）　2B
Timeout	NC-MMC トランザクションの実行時間を秒単位でモニタします。
Prefix	ACC 変数の前に挿入されるストリング

注 : 引数 WinFile を拡張子 .NSK のファイルにすると，ドメインサービスにより ACC ファイルに加えて NSK ファイルも作成されます。NSK ファイルには対応する LINK コマンドが入ります。

ACC ファイル

```
/NC/_N_NCK_GD2_ACC ; global NCK user variable MGUD
/NC/_N_CH02_GUD_ACC ; global user variable in the 2nd channel
/NC/_N_AX_SEA_ACC ; axis specific setting data
/NC/_N_CH_TEA_ACC ; channel specific NC machine
```

ドライブマシンデータの接続の確立

MAP_ACC_NC	コマンドのヘッダ
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	領域。この例の数値 5 は " ドライブ " の領域アドレスを表します。
7F	DataBlock。この例のアドレス 7F は モジュール " ドライブのサービス値 " を表します。
10	モニタする時間。この例では 10 秒。
/ACC/driveVSA/MD/	接頭部。この例では，後でデータのアクセスに 使用されるストリング。

■ 例 8-32 ドライブマシンデータの接続の確立

```
Sub Form_Load ()
Label1.LinkTopic = "NCDDE|NCU840D"
Label1.LinkMode = 2
abel1.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
```

すでに確立済みの接続へのアクセス

すでに前述の例で確立されたリンクに，以下の構成要素でアクセスします。:

/ACC/driveVSA/MD/ 前述の例で MAP コマンドを呼び出した結果に基づく接頭部

$MD_TORQUE_THRESHOLD_X[1] 先頭が $ のマシンデータの名前。

■ 例 8-33 確立済みの接続へのアクセス

```
Sub Form_Load ()
Label1.LinkTopic = "NCDDE|NCU840D"
Label1.LinkMode = 2
Label1.LinkItem= "/ACC/driveVSA/MD/$MD_TORQUE_THRESHOLD_X[1]"
Label1.LinkRequest
End Sub
```

グローバルユーザ変数へのアクセス

8.13 には，グローバルユーザ変数にアクセスする方法が示されています。

MAP_ACC_NC コマンドのいくつかの例

パラメータ WinFile および NcFile の後に，コンマ 1 つとスペース文字 1 つがあるか確認してください。

■ 例 8-34 MAP_ACC_NC コマンドの例

すべてのマシンデータ：
MAP_ACC_NC(c:mp¥c.nsk, /NC/_N_COMPLETE_TEA_ACC, trans,0,1A,10,/MD/)
すべての NCK マシンデータ：
MAP_ACC_NC(c:mp¥nc.nsk, /NC/_N_NC_TEA_ACC, trans,0,1A,10,/NC/)
チャネル 1 のチャネルマシンデータ：
MAP_ACC_NC(c:mp¥ch1.nsk, /NC/_N_CH1_TEA_ACC, trans,2,1A,10,/CH1/)
すべての軸固有マシンデータ：
AP_ACC_NC(c:mp¥ax.nsk, /NC/_N_AX_TEA_ACC, trans,3,1A,10,/AX/)
すべての NC グローバル設定データ：
MAP_ACC_NC(c:mp¥sea.nsk, /NC/_N_NC_SEA_ACC, trans,0,16,10,/SEA/)
すべての軸固有設定データ：
MAP_ACC_NC(c:mp¥axs.nsk, /NC/_N_AX_SEA_ACC, trans,3,16,10,/AXSEA/)
すべての NC グローバルユーザデータ：
MAP_ACC_NC(c:mp¥gud.nsk, /NC/_N_NC_GUD_ACC, trans,0,17,10,/GUD/)
すべてのチャネル固有ユーザデータ：
MAP_ACC_NC(c:mp¥gud.nsk, /NC/_N_CH_GUD_ACC, trans,2,17,10,/GUD/)
すべての NC グローバルユーザデータ 1（=SGUD）：
MAP_ACC_NC(c:mp¥gd1.nsk, /NC/_N_NC_GD1_ACC, trans,0,17,10,/GUD1/)
すべてのチャネル固有ユーザデータ 1（=SGUD）：
MAP_ACC_NC(c:mp¥gch1.nsk, /NC/_N_CH_GD1_ACC, trans,2,17,10,/GUD1/)
すべての NC グローバルユーザデータ 2（=MGUD）：
MAP_ACC_NC(c:mp¥gd2.nsk, /NC/_N_NC_GD2_ACC, trans,0,2D,10,/GUD2/)
すべてのチャネル固有ユーザデータ 2（=MGUD）：
MAP_ACC_NC(c:mp¥gd2.nsk, /NC/_N_CH_GD2_ACC, trans,2,2D,10,/MGUD/)
すべての NC グローバルユーザデータ 3（=UGUD）：
MAP_ACC_NC(c:mp¥gd3.nsk, /NC/_N_NC_GD3_ACC, trans,0,2E,10,/GUD3/)
すべての NC グローバルユーザデータ 4（=GUD4）：
MAP_ACC_NC(c:mp¥gd4.nsk, /NC/_N_NC_GD4_ACC, trans,0,2F,10,/GUD4/)
すべての NC グローバルユーザデータ 5（=GUD5）：
MAP_ACC_NC(c:mp¥gd5.nsk, /NC/_N_NC_GD5_ACC, trans,0,30,10,/GUD5/)
すべての NC グローバルユーザデータ 6（=GUD6）：
MAP_ACC_NC(c:mp¥gd6.nsk, /NC/_N_NC_GD6_ACC, trans,0,31,10,/GUD6/)
すべての NC グローバルユーザデータ 7（=GUD7）：
MAP_ACC_NC(c:mp¥gd7.nsk, /NC/_N_NC_GD7_ACC, trans,0,32,10,/GUD7/)
すべての NC グローバルユーザデータ 8（=GUD8）：
MAP_ACC_NC(c:mp¥gd8.nsk, /NC/_N_NC_GD8_ACC, trans,0,33,10,/GUD8/)
すべての NC グローバルユーザデータ 9（=GUD9）：
MAP_ACC_NC(c:mp¥gd9.nsk, /NC/_N_NC_GD9_ACC, trans,0,34,10,/GUD9/)

8.7 PI サービス

概説

PI サービスを使用して，コマンドを NC と PLC に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あります。

NCDDE サーバの PI サービスは以下のとおりです。

PI_START コマンドを実行するよう NCK に指示します。
PI_START_BINARY コマンドを実行するよう PLC に指示します。
PI_STOP コマンドの実行を停止するよう NCK に指示します。
PI_STOP_BINARY コマンドの実行を停止するよう PLC に指示します。
PI_RESUME 停止した実行を再開するよう NCK に指示します。
PI_RESUME_BINARY 停止した実行を再開するよう PLC に指示します。

■ PI_START(_BINARY)

説明

この機能を使用すると，指示を MMC から NCK に送信できます。

用途

この機能は，NCK でジョブを開始する場合に適しています。

非バイナリ転送は，NCK に転送する場合に適しています。

バイナリ転送は，データを PLC，NC，およびドライブに転送する場合に適しています。

構文

PI サービスを呼び出すためのコマンド行は，以下の構文に従います。

PI_START(server name, argument 1, argument 2 ... argument n, PIname)

PI_START_BINARY (server name, argument, PI name)

NC の場合の PI name の先頭は _N_ で，その後に 6 文字続きます。PLC に適用される規則は多少違っています。

引数

引数の機能は，それらが使用される PI サービスによって異なります。したがって，引数の詳細については，オンラインヘルプに説明されています。

パートプログラムの選択

この例では，PI サービス "SELECT"（チャネル内で実行されるプログラムを選択する）により，パートプログラム "BSP.MPF" が選択される様子を示しています。NC ファイルパスではなく領域パスをこのコマンドに入力しなければならないことに注意してください。

■ 例 8-35 パートプログラムの選択

```
Sub Form_Load ()
Label1.LinkTopic = "NCDDE|NCU840D"
Label1.LinkMode = 2
Label1.LinkExecute "PI_START(/NC,201,/_N_MPF_DIR/
_ _N_BSP_MPF, _N_SELECT)"
End Sub
```

OB 1 の活動化

受動ファイルシステムにすでに格納されている OB1 をアクティブにします。

■ 例 8-36 OB 1 の活動化

```
Sub Form_Load ()
Label1.LinkTopic = "NCDDE|NCU840D"
Label1.LinkMode = 2
Label1.LinkExecute "PI_START_BINARY( /PLC, mailto:""@1d1@1d0@@0800001P"",
_ _INSE)"
End Sub
```

パートプログラムの選択の停止

この例では，パートプログラム "BSP.MPF" に関する PI サービス "SELECT"（チャネル内で実行されるプログラムを選択する）が停止される様子を示しています。

■ 例 8-37 パートプログラムの選択の停止

```
Sub Form_Load ()
Label1.LinkTopic = "NCDDE|NCU840D"
Label1.LinkMode = 2
Label1.LinkExecute "PI_STOP(/NC,201,/_N_MPF_DIR/
_ _N_BSP_MPF, _N_SELECT)"
End Sub
```

OB 1 の活動化の停止

■ 例 8-38 OB 1 の活動化の停止

```
Sub Form_Load ()
Label1.LinkTopic = "NCDDE|NCU840D"
Label1.LinkMode = 2
Label1.LinkExecute "PI_STOP_BINARY( /PLC
_ ""@1d1@1d0@@0800001P"", _INSE)"
End Sub
```

8.8 NCDDE サーバのその他のコマンド

概説

表 8.7 に，NCDDE サーバのその他のコマンドがリストされています。

表 8.7 NCDDE サーバのその他のコマンド

コマンド	意味
NEW	ローカル変数を作成します。
FREE	変数を削除します。
ANIMATE	ローカル変数に継続的に変更を加えます。
CALL	ファイル内の NCDDE コマンドを実行します。
PLC_MEMORYRESET	PLC メモリをリセットします。

■ NEW

説明

NCDDE サーバでローカル／内部変数を作成します。作成後，その変数にアクセスできるようになります。

用途

コマンド NEW を使用すると，NCDDE サーバのローカル変数が新しく作成されます。この変数へのアクセス時には，NCK とは通信されません。VarName の名前の変数が既存の場合，その変数が削除されてから新しい変数が作成されます（8.8 の FREE コマンドと似ています)。

構文

NEW (VarName ,value)

引数

表 8.8 NEW の引数

引数	構文	説明
VarName	<string>	作成する変数の名前
value	<number>	変数の初期設定値

内部変数の作成

変数 "test" を NCDDE サーバに作成し，値 10.0 で初期設定します。

■ 例 8-39 内部変数の作成

```
Sub Form_Load ()
Label1.LinkTopic = "NCDDE|NCU840D"
Label1.LinkMode = 2
```

```
Label1.LinkExecute " NEW ( test , 10.0 )"
End Sub
```

■ FREE

説明

NCDDE サーバにある変数を削除します。

用途

コマンド”FREE” は, コマンド”NEW” やコマンド”LINK" で作成された変数を削除します。変数がファイル転送サービス（8.6.1）により状態変数として使用中になっている場合は，コマンド”FREE” は拒否されます。変数に対する Advise Link（Hotlink）があると，それらのリンクは除去されます。NCK および PLC との他のトランザクションは打ち切られます。

構文

FREE (VarName)

引数

表 8.9 FREE の引数

引数	構文	説明
VarName	< String >	削除する変数の名前

内部変数の削除

NCDDE サーバにある変数 "test" を削除します。

■ 例 8-40 内部変数の削除

```
Sub Form_Load ()
Label1.LinkTopic = "NCDDE|NCU840D"
Label1.LinkMode = 2
Label1.LinkExecute " FREE(test)"
End Sub
```

■ ANIMATE

説明

"NEW" を使用して作成されたローカル変数の値が，NCDDE サーバにより継続的に変更されるようにします。値は約 1 秒のサイクルで大きくなります。

用途

アプリケーションのテストに使用します。

構文

Animate (VarName)

引数

表 8.10 Animate の引数

引数	構文	説明
VarName	< String>	変更を加える変数の名前

内部変数の変更

NCDDE サーバにある変数 "test" の値に継続的に変更を加えます。

■ 例 8-41 内部変数の変更

```
Sub Form_Load ()
Label1.LinkTopic = "NCDDE|NCU840D"
Label1.LinkMode = 2
Label1.LinkExecute " ANIMATE(test)"
End Sub
```

■ CALL

説明

コマンドファイルを解釈します。

用途

コマンド CALL を使用すると，ファイルに記録されている NCDDE コマンドが実行されます。ファイルの個々の行が，コマンドとして NCDDE サーバの構文解析プログラムに渡されます。ファイルにはコメントやスペース行を含めることができます。すべての NCDDE コマンドファイルには拡張子 .NSK が使用されます。

ヒント : 用途に合わせて NCDDE サーバをカスタマイズできます。

構文

CALL (FileName)

引数

表 8.11 CALL の引数

引数	構文	説明
FileName	< string >	NCDDE コマンドファイルの名前

例

ファイル "¥MMC2¥ NCDDE311.NSK" を参照してください。

■ PLC_MEMORYRESET

説明

NCDDE サーバのコマンド PLC_MEMORYRESET は，PLC メモリをリセットします。領域アドレスとして /PLC を指定してください。

用途

PLC メモリをリセットします。

構文

PLC_MEMORYRESET(AreaAddr)

引数

表 8.12 PLC_MEMORYRESET の引数

引数	構文	説明
AreaAddr	< String >	領域アドレス

注 : PLC を停止する方法については，11.13.8 に説明されています。

PLC のリセット

PLC をリセットします。しかしその PLC は，すでに停止しているものでなければなりません。

■ 例 8-42 PLC のリセット

```
Sub Form_Load ()
Label1.LinkTopic = "NCDDE|NCU840D"
Label1.LinkMode = 2
Label1.LinkExecute "PLC_MEMORYRESET(/PLC)"
End Sub
```

8.9 OEM-Visual Basic コントロール（VBX ファイル）

概説

このコントロールを使用すれば，Visual Basic の標準コントロールを使用した場合の DDE 通信の欠点のいくつかが解決されます。

注 : OEM アプリケーションを開発する場合，これらの OEM-Visual Basic コントロールを使用して，NCDDE サーバにアクセスするようお勧めします。

" ラベル " や " テキストフィールド " などの形態の標準コントロールは，DDE 通信を提供します。しかし，この通信には以下のようないくつかの欠点があります。

- イベントが失われる場合がある。
  Linkmode = 1 で DDE 変数の値が変更された場合，VB プログラムの変更プロシージャが呼び出されないことがあります（唯一の解決方法は，タイマコントロールを使用して，値をポーリングすることです）。
- DDE 機能がネストされない。
  DDE 変更プロシージャ内で，コントロールの DDE 機能が活動化できなくなる場合があります。（同様に唯一の解決方法は，タイマなどを使用することです。）
- 同期トランザクションしか認識されない。
  インストールされた Hotlink および Request の応答時間のインターバルが非常に長くなります。これは特に，これらのアクションに複数の CPU が関係している場合に当てはまります（NCK，PLC など）。
- たくさんのリソースを必要とする。
  DDE を使用する制御インスタンスごとに，DDE 通信が導入されます。この通信は 2 つの WINDOW ハンドルを使用するので，希少なユーザリソースを妨害します。
- NCDDE で LastError を簡単に処理できない。
  NCU との通信を NCDDE を介して行う場合，NCDDE サーバは，エラーの詳細な分析を行う DDE 変数 "LastError" を提供しています。この変数は DDE 通信ごとに指定されますが，有効になるのは，DDE に DDE_FNOTPROCESSED が戻された場合だけです。

8.9.1 ファイル DCTL.OCX

概説

DCTL OCX コントロールは，拡張された DDE 機能を備えたグラフィカルコントロールです。これは標準コントロールのラベルに似ていますが，さらに以下の利点が加わっています。

- Windows リソースの必要量が最小限に抑えられている。
  "DDE Request"，"DDE Poke"，および "DDE Execute" は，リソースを一時的にしか使用しません。コントロール DCTL.OCX を使用する Windows プロセスの "DDE Hotlink" は，すべてが 1 つの Windows ハンドルを使用します。
- NCDDE サーバとの緊密な連携。
  たとえば，トランザクションが正常に処理されなかった場合に，"LastError" が

送られます。
- 速度が速い。
  複数のトランザクションを並列的に処理できるので，アプリケーションの速度が向上します。
- 出力速度の向上。
  画面出力の最適化および索引フィルタ操作により，画面表示の速度が向上しています。さらに，BASIC プログラミングがより簡単になりました。
- 副次作用の回避。
  Visual Basic コントロールの典型的な副次作用（プログラムされた接続を Esc キーを押すことにより打ち切るなど）を，回避することができます。

この章では，最初にこの新しいコントロールの特性について，次に追加されたイベントについて説明していきます。この章には，DCTL.OCX の使用方法を示す，いくつかの例が含まれています。

プロパティ

DCTL.OCX コントロールのほとんどの特性は，Visual Basic の標準コントロールの特性に対応しています。以下の特性が該当します。

- Style 特性
- Color 特性
- Base 特性
- Drag 特性
- Font 特性

Visual Basic の他のコントロールとは異なる，以下のような DCTL.OCX コントロールもあります。

- DDE 特性
- HorAlignment 特性
- VertAlignment/Multiline 特性
- WordBreak 特性
- TabSize 特性
- LastError 特性
- Data 特性
- DataToCaption 特性
- LinkCmd 特性
- LinkNext 特性
- LinkFilter 特性

DDE 特性

DDE 特性は以下のとおりです。
LinkItem
LinkTopic（デフォルトで事前設定されている NCDDE）
LinkTimeout（LinkCmd を同期化するため）

HorAlignment 特性

この特性は，キャプション表示のテキスト水平位置調整を制御します。

表 8.13 テキスト水平位置調整

値	特性
0	左寄せ（デフォルト）
1	右寄せ
2	中央そろえ

VertAlignment/Multiline 特性

この特性は，表題表示のテキスト垂直位置調整を制御します。テキスト垂直位置調整の代わりに，複数行表示を選択することもできます。その場合，Word-BreakProperty が折り返しを判別します。

表 8.14 テキスト垂直位置調整

値	特性
0	垂直方向に中央そろえ（デフォルト）
1	上寄せ
2	下寄せ
3	複数行

WordBreak 特性

特性 VertAlignment/Multiline で Multiline（値 = 3）が設定されている場合，特性 WordBreak は，折り返しを以下のように判別します。

表 8.15 折り返しのタイプ

値	特性
False	CR/LF（復帰／改行シーケンス）での語折り返し
True	ワードが行に収まらない場合に，自動的に語折り返しを行います。改行シーケンスも行の折り返しを行います。

TabSize 特性

各タブに使用するスペース文字の数を指定します。デフォルト値は 8 です。許可されている最大数は 255 です。

LastError 特性

この特性を使用することにより，エラーメッセージを送信できます。

この値は，サーバとの DDE トランザクションが開始されたときに，0 にリセットされます。トランザクション中にエラーが発生した場合，および DCTL コントロールがこのエラーを検出した場合，詳細なエラーコードが求められます。これは，LastError 特性を使用して入手できます。

注 : DCTL コントロールは，NCDDE サーバからデータ（'#' 文字やスペースを含む）として送られたエラーのデコードは行いません。

変数 LastError については，11.7.1 で説明されています。

Data 特性

Data 特性は，以下の DDE ランザクションの引数として使用します。

表 8.16

DDE トランザクション	引数
Request	要求される変数値（"DataToCaption" が FALSE に設定されている場合）
Advise Link	更新される値（"DataToCaption" 特性が FALSE に設定されている場合）
Poke	送られる値
Execute	実行されるコマンド

DataToCaption 特性

DataToCaption 特性は，DDE トランザクションが受け取るデータの宛先を判別します。

表 8.17 テーブルの宛先

値	意味
True	データの宛先は Caption 特性です
False	データの宛先は Data 特性です

LinkCmd 特性

特性 LinkCmd を変更すると，DCTL コントロールの DDE 活動が開始されます。活動がなければ，LinkCmd は 0 です。

表 8.18 LinkCmd 特性

番号	変更後	DDE 活動	終了方法
1	Advise Link	AdviseLink を確立します。AdviseLink が確立されてから，戻されます。AdviseLink は Stop コマンドで削除できます。	Stop
2	Advise Link_NotifyData	"1 - AdviseLink" の DDE 活動に加えて, DDE データを受け取った時点でアクション（1）が行われます。	Stop
3	Advise Link_NotifyData WhenVisible	"1 - AdviseLink" の DDE 活動に加えて, DDE データを受け取った時点でアクション（2）が行われます。	Stop
4	Advise LinkAsync	AdviseLink を確立します。AdviseLink が確立される前に，戻されます。AdviseLink は Stop コマンドで削除できます。	Stop

5	Advise LinkAsync_NotifyData	"4 - AdviseLinkAsync" の DDE 活動に加えて，DDE データを受け取った時点でアクション（1）が行われます。	Stop
6	Advise LinkAsync_NotifyDataWhenVisible	"4 - AdviseLinkAsync" の DDE 活動に加えて，DDE データを受け取った時点でアクション（2）が行われます。	Stop
7	Stop	AdviseLink を削除します。AdviseLink が削除されてから，戻されます。	それ自体
8	StopAsync	AdviseLink を削除します。AdviseLink が削除される前に，戻されます。	Sync
9	StopAsync_Notify	"8 - StopAsync" の DDE 活動に加えて，削除が完了した時点でアクション（1）が行われます。	Sync
10	StopAsync_NotifyWhenVisible	"8 - StopAsync" の DDE 活動に加えて，削除が完了した時点でアクション（2）が行われます。	Sync
11	Request	DDE 変数を読み取ります。読み取りが完了してから，戻されます。	それ自体
12	RequestAsync	DDE 変数を読み取ります。読み取りが完了する前に，戻されます。	Sync
13	RequestAsync_Notify	"12 - RequestAsync" の DDE 活動に加えて，読み取りが完了した時点でアクション（1）が行われます。	Sync
14	RequestAsync_NotifyWhenVisible	"12 - RequestAsync" の DDE 活動に加えて，読み取りが完了した時点でアクション（2）が行われます。	Sync
15	Execute	コマンドをサーバに送ります。コマンドの実行が完了してから，戻されます。	それ自体
16	ExecuteAsync	コマンドをサーバに送ります。コマンドの実行が完了する前に，戻されます。	Sync
17	ExecuteAsync_Notify	"16 - ExecuteAsync" の DDE 活動に加えて，コマンドの実行が完了した時点でアクション（1）が行われます。	Sync
18	ExecuteAsync_NotifyWhenVisible	"16 - ExecuteAsync" の DDE 活動に加えて，コマンドの実行が完了した時点でアクション（2）が行われます。	Sync
19	Poke	DDE 変数を書き込みます。書き込みが完了したら，戻されます。	それ自体
20	PokeAsync	DDE 変数を書き込みます。書き込みが完了する前に，戻されます。	Sync
21	PokeAsync_Notify	"20 - PokeAsync" の DDE 活動に加えて，書き込みが完了した時点でアクション（1）が行われます。	Sync
22	PokeAsync_NotifyWhenVisible	"20 - PokeAsync" の DDE 活動に加えて，書き込みが完了した時点でアクション（2）が行われます。	Sync
23	Sync	同期コマンドのように，非同期コマンドを終了させます。非同期コマンドが実行されていなかった場合，特に行われる操作はありません。	それ自体

アクション

前述の表で使用されているアクション（1）と（2）について，以下に説明します。

アクション（1）

DdeNotify イベントプロシージャの呼び出しを試行します。Visual Basic がこの時点でイベントプロシージャを読み出さない場合，またはイベントプロシージャのパラメー

タが変更されていない場合，DCTL コントロールは，DdeNotify イベントプロシージャの引き数が変更されない限り，1 秒間に 10 回ずつこのイベントの送信を試行します。

アクション（2）

DCTL コントロールは，Windows からペイントメッセージを受け取った時点で，DdeNotify イベントを呼び出します。これらのペイントメッセージが，Windows によって作成されたことを確証するために，DdeNotify の引数が変更されない限り，コントロールの左上隅のピクセルは無効のままです。実際のところ，コントロールが見えない場合，この機構が表示を抑止しています。

注 : 新しい DDE 活動を開始する前に，必ず前の DDE 活動を終了させなければなりません。これは，前述の表の右端の欄（終了方法）にある引数を使用して行えます。

同じ Windows プロセス内のすべての DCTL コントロールの Hotlink が，同じ "LinkTopic" 特性を使用する場合，それらは 1 つの DDE 接続を共有します。他の活動（ホットリンクを除く）の DDE 接続は，動的に作成され，削除されます。上記の理由に加えて，DCTL コントロールに専用のウィンドウがないので，必要な Windows リソースは非常に少なくて済みます。

注 : LinkCmd 特性を変更すると，LinkTopic，LinkTimeout，および LinkItem 特性が評価されます。その結果，LinkCmd 特性が変更されていれば，これらの特性に関するいくつかのエラーが報告されます。そのため，それらの評価はこの時点で行われなければなりません。

LinkNext 特性

オプション特性 LinkNext は別の DCTL コントロールの名前を保持します。またその索引を保持するようにもできます。

LinkNext 特性が空ではない場合，DCTL コントロールは，AdviseLink から送られたストリングを調べて，NCDDE 索引指定（5 桁と末尾に ':'）を探します。ストリングを索引付きのサブストリングに分けて，LinkNext 特性が作成した DCTL コントロールのチェーンリストに従って，それらを送ります。索引に一致する LinkFilter 特性を持つコントロールが，対応するサブストリングを受け取ります。この方法で入手されないサブストリングは，失われます。

LinkFilter 特性

LinkFilter 値の範囲は 0 ～ 65535 です。この特性の使用方法については，LinkNext に関する段落で説明されています。

DCTL.OCX のイベント

DCTL コントロールのほとんどのイベントは，以下のような，他の VisualBasic 標準コントロールとまったく同じです。

- Click
- DblClick
- MouseDown
- MouseMove
- MouseUp

- DragDrop
- DragOver

イベント DdeNotify

特に DDE 通信の場合，イベント DdeNotify が認識されます。これは新しい AdviseLink データが受け取られたことや，非同期 DDE トランザクションが終了したことを示します。この使用方法については，LinkCmd に関する段落（アクション（1）および（2））で説明されています。

構文

Sub ctlname_DdeNotify (Index As Integer , Flag As Integer)

引数

Index は，コントロールがコントロール配列内にある場合に，そのコントロールを一意的に識別します。また Flag は，イベントが実際に基本レベルで到着したことを DCTL コントロールに示します。
Flag の値は，イベントプロシージャの呼び出しごとに変更されなければなりません。これは，フラグ引数が変更されるまでに，DCTL コントロールが DdeNotify イベントを開始するためです。変更されない場合，連携的な活動がいつまでも行われる結果になり，システムに不必要に負荷がかかります。

8.9.2 DCTL.OCX の採用

変数の読み取りと表示

DDE 変数を読み取って，読み取られた値をすぐに使用し，その値を画面に表示する場合を想定します。
この場合，DCTL コントロール（名前は，たとえば Dctl1）を，変数値を表示する画面に配置します。以下のようなコードを使用します。

■ 例 8-44 変数の読み取りと表示

```
Sub Form_Load ()
Dctl1.LinkItem = "/Channel/Parameter/R[1]" ' the variable's name
Dctl1.DataToCaption = TRUE ' that's default, can be omitted
Dctl1.LinkCmd = 11 ' commands the reading
' here ctl1.Caption holds the value of the DDE variable
End Sub
```

Data 特性での DDE 変数の読み取り

DDE 変数を読み取って，読み取られた値をすぐに使用するものの，その値を画面には表示しない場合を想定します。この場合，ラベルを付ける DCTL コントロール（名前は，Dctl1 など）の 1 つをフォームで使用します。以下のようなコードを使用します。

■ 例 8-45 Data 特性での読み取り

```
Sub Form_Load ()
Dctl1.LinkItem = "/Channel/Parameter/R[1]" ' the variable's name
Dctl1.DataToCaption = FALSE ' routing data to the Data property
Dctl1.LinkCmd = 11 ' commands the reading
```

```
' here Dctl1.Data holds the value of the DDE variable
End Sub
```

DDE 変数の書き込み

DDE 変数を書き込む場合を想定します。

この場合，ラベルを付ける DCTL コントロール（名前は，Dctl1 など）の 1 つをフォームで使用します。以下のようなコードを使用します。

■ 例 8-46 変数の書き込み

```
Sub Form_Load ()
Dctl1.LinkItem = "/Channel/Parameter/R[1]" ' the variable's name
Dctl1.Data = 12 ' the value
Dctl1.LinkCmd = 19
' commands the writing
' here the NC variable is already successfully set to 12
End Sub
```

DDE コマンドの実行

DDE コマンドをサーバに送る場合を想定します。

この場合，ラベルを付ける DCTL コントロール（名前は，DCTL1 など）の 1 つをフォームで使用します。以下のようなコードを使用します。

■ 例 8-47 コマンドの実行

```
Sub Form_Load ()
Dctl1.Data = "Pi_start(/NC,001,_N_SET_OF)" ' the command
Dctl1.LinkCmd = 15 ' sends the command
' here the command is already successfully executed
End Sub
```

DDE ホットリンクの表示

DDE 変数の値を画面に表示する場合を想定します。

この場合，DCTL コントロール（名前は，Dctl1 など）を，変数値を表示する画面に配置します。さらに，DCTL コントロールのバックグラウンド活動に作成タスクを作成する，ホットリンク作成の開始のみを選択します。以下のようなコードを使用します。ただし，設計時のコード化された特性設定を実行することもできます。

■ 例 8-48 DCTL へのホットリンク

```
Sub Form_Load ()
Dctl1.LinkItem = "/Channel/Parameter/R[1]" ' the variable name
Dctl1.DataToCaption = TRUE ' that's default, can be omitted
Dctl1.LinkCmd = 4 ' initiates the creation of a hotlink
End Sub
```

並列アクションによる DDE の速度向上

フォームをロードし，いくつかの独立 DDE 活動を実行しなければならない場合を想定します。この場合，フォームがロードされる速度に関心があるので，DCTL コントロールに関しては，DDE 活動を並列に実行するのが最も適しています。以下のコードで例を示します。

■ 例 8-49 並列実行による速度向上

```
Sub Form_Load ()
' start reading variable 1
Dctl1.LinkItem = "/Channel/Parameter/R[1]" ' the variable name
Dctl1.LinkCmd = 12 ' initiates the reading
' start reading variable 2
Dctl2.LinkItem = "/Channel/Parameter/R[2]" ' the variable name
Dctl2.LinkCmd = 12 ' initiates the reading
' start reading variable 3
Dctl3.LinkItem = "/Channel/Parameter/R[3]" ' the variable name
Dctl3.LinkCmd = 12 ' initiates the reading
' start a hotlink into display
Dctl4.LinkItem = "/Channel/Parameter/R[4]" ' the variable name
Dctl4.DataToCaption = TRUE ' that's default, can be omitted
Dctl4.LinkCmd = 4 ' creates the hotlink
' start a command execution
Dctl5.Data = "Pi_start(/NC,001,_N_SET_OF)" ' the command
Dctl5.LinkCmd = 16 ' commands execution
' here the tree variable accesses, the hotlink creation and the
' command are working in parallel. You can not be sure that any
' of them has completed.
Dctl1.LinkCmd = 23 ' wait until variable 1 read
Dctl2.LinkCmd = 23 ' wait until variable 2 read
Dctl3.LinkCmd = 23 ' wait until variable 3 read
Dctl5.LinkCmd = 23 ' wait until command executed
' here the variable accesses and the command have completed,
' the hotlink will show it's value on screen as soon as possible.
End Sub
```

テキスト配置による DDE の速度向上

たくさんのデータ項目を高い頻度で表示しなければならない場合を想定します。この場合，プログラムは表示タスクに BASIC 言語を入力すべきではありません。さらに，転送されるデータの量は最小限にとどめなければなりません。NCDDE 側では，配列アクセスと，配列アクセスおよび "Field" データ準備の組み合わせにより，この要件をサポートしています。DCTL コントロールは，これらの機能に向いている，複数行表示および索引フィルタを提供しています。

■ 例 8-50 テキスト配置機能を使用した速度向上

```
'NCDDE array access with "Field" data preparation-DCTL indexfiltering:
 high frequency display of 5 values in 5 different controls
Dctl1.LinkItem = "/Channel/Parameter/R[1,5](!""!d%12.5g"")" ' variable
Dctl1.LinkFilter = 1 ' index of accepted data
Dctl1.LinkNext = "Dctl2" ' linkage to the next control
Dctl2.LinkFilter = 2 ' index of accepted data
Dctl2.LinkNext = "Dctl3" ' linkage to the next control
Dctl3.LinkFilter = 3 ' index of accepted data
Dctl3.LinkNext = "Dctl4" ' linkage to the next control
Dctl4.LinkFilter = 4 ' index of accepted data
Dctl4.LinkNext = "Dctl5" ' linkage to the next control
Dctl5.LinkFilter = 5 ' index of accepted data
Dctl1.LinkCmd = 4 ' initiates the creation of a hotlink
NCDDE array access - Dctl multiline display:
' high frequency display of 5 values in a column
```

```
Dctl1.LinkItem = "/Channel/Parameter/R[1,5](""!d%12.5g"
Dctl1.LinkItem = Dctl1.LinkItem + Chr$(13) + Chr$(10)+""")"
Dctl1.DataToCaption = TRUE ' that's default, can be omitted
Dctl1.VertAlignment = 3 ' multiline selection
Dctl1.LinkCmd = 4 ' initiates the creation of a hotlink
```

通知機能の使用

画面表示レイアウトが，DDE がアクセスする変数に依存している場合を想定します。この場合，DCTL へのホットリンクをこの変数に作成することができます。また，変数値の変更時に，DCTL 通知機能を使用して，レイアウトを再配置することができます。再配置は時間がかかるタスクなので，再配置の実行は，フォームが画面上に表示されている場合にのみ考慮してください。

■ 例 8-51 通知機能の使用

```
Sub Form_Load ()
' basic code that creates a hotlink with notification "when visible"
ctl1.LinkItem = "/Channel/Parameter/R[1]" ' the variable name
Dctl1.LinkCmd = 6 'initiates the creation of a hotlink
' handler for the notification event
End Sub
Sub Dctl1_DdeNotify ( Index As Integer , Flag As Integer )
Flag = Flag + 1 ' Flag MUST change
... ' rearrangement to be done
End Sub
```

エラー処理

読み取り，書き込み，および実行に関する典型的なエラー処理

■ 例 8-52 エラー処理

```
On Error Goto TypicalErrorHandling
Dctl1.LinkCmd = 11 ' a DDE activity
...
TypicalErrorHandling:
Select Case Dctl1.Lasterror / 16777216 ' selection by error source
Case 2 ' MPI level error
.. ' e. g. no connection to NC
ase 3 , 5 ' NC/PLC level error
... ' e. g. non existing variable
Case 7 ' DCTL level error
Select Case Dctl1.Lasterror MOD 256 ' selection by error code
Case 7 ' DCTL level time-out occurred
...
 Else ' other DCTL level errors
...
End Select
Case Else ' other error sources
...
End Select
...
```

8.10 NCDDE アクセスの診断機能

8.10.1 NCDDE サーバのテスト機能

概説

特に，NCDDE サーバのテスト機能は，ファイル作成時に NCDDE サーバで宣言されていたローカルおよび外部変数に関する情報を提供します。以下の手順で呼び出します。

(1) プログラムグループ SINUMERIK MMC-OEM にある NCDDE サーバを開始します。
(2) ALT+TAB を押して，NCDDE プログラム（つまり，NC 通信 DDE サーバ）に移動します。アイコンが作成されます。
(3) そのアイコンをクリックします。以下のウィンドウが表示されます。

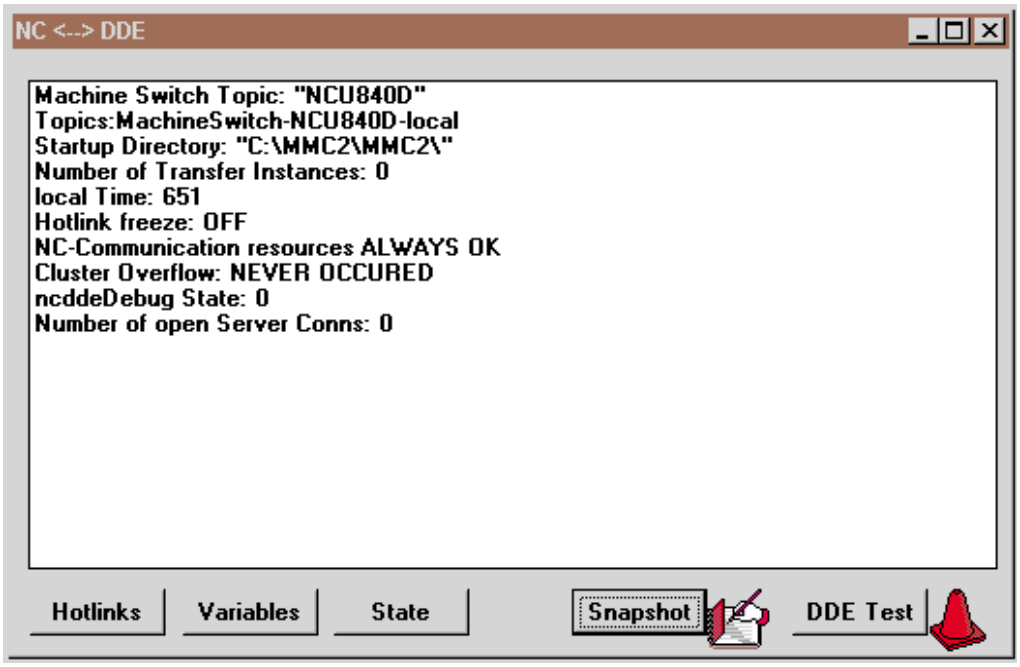


図 8.2 NCDDE サーバの標準表示

これらの機能は，主に NCDDE サーバの環境でのデバッグ用です。

[Hotlinks]

すべての既存の Advise Link（Hotlink および Warmlink）が入ったリストが作成されます。この表には 5 つの列があり，それぞれには以下の意味があります。

表 8.19 ホットリンク

列	情報	内容
1	PDU 参照	内部値 :NCU と PLC との通信に使用できる PDU 参照
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：外部 Advise Link が別のジョブに追加された

3	更新時刻	NCDDE サーバの内部時刻装置内の最新のリフレッシュPDUの時刻
4	LastError 変数	本書の 11.14 章に従う LastError 指定。サーバの DDE インタフェースで報告されている値と同じである必要はありません。これは，ある接続に対応するいくつかのトランザクションの最新のエラーが示される場合があるためです。
5	変数名	11 章に従う変数名。

[Variables]

すべての変数，NCDDE サーバの接続先，およびそれらがあるデバイス（"LOCAL" または "PLC/NC"）を含むリストが作成されます。

[Snapshot]

このボタンを押すと，"NCDDE_X.TXT" というファイルが作成されます。このファイルには，NCDDE サーバの状況，ホットリンク，および変数が含まれています。

[DDE Test]

このボタンを押すと，テストプログラム "DDETEST.EXE" が実行されます。これには，以下の機能があります。

表 8.20 DDE テストコマンド

コマンド	アクション	意味
Passive	なし	状況がリセットされる。アクティブになる機能はなし
Hotlink	開始	Advise Link の確立
Request	実行	変数の読み取り
Poke	実行	変数の書き込み
Execute	実行	サービスの実行

インストールされている NC を Service|Topic に指定します（たとえば，NCDDE|NCU840D）。"DEFAULT_NC" では，ファイル "MMC.INI" からデフォルト設定が読み取られます。
5 個の選択項目をいずれかをクリックすると，機能 Command が切り替わります。
エラーメッセージ LastError については，11.7 で説明されています。

8.10.2 接続の状態

変数 NcState

サーバは，ローカル変数 NcState を介する CNC への接続の状態を示します。この変数はサーバが開始された直後に作成されます。これは，DDE インタフェースでは変更できないという事実だけが，サーバのほかのローカル変数とは異なります。

この変数は，以下の状態のいずれかを示します。

表 8.21 変数 NcState の状態

値	意味
0	正常運転
1	一部の接続で CNC が切断されている
2	すべての接続で CNC が切断されている
3	起動ファイルの解釈中
4	サーバの初期設定

8.10.3 障害追及

NCK からのエラーメッセージ

エラー条件（リソース不足，アクセス違反，操作モードの間違いなど）には，トランザクションの確認を介して，NCK から通信します。NCDDE サーバがこれらのエラー条件を処理できない場合，エラー状態が示されて（アプリケーションが結果を入手しないなど），DDE インタフェースのそれに応じたトランザクション Request，Poke，および Execute が終了します。

変数 LastError

詳細な診断が，最新のトランザクションに関する情報を保持する変数 LastError によって提供されます。これは，Link Item の LastError によって読み取ることができます。読み取り後，この変数は 0 に設定されます。必ず NCDDE サーバに登録されている最新のエラーが示されます。

変数 LastError は 4 バイトから成り立っています。以下のエラーグループは，バイトの降順で（高位バイトから下位バイトへ）示されます。

- 上位のエラークラス，エラーソース
- エラー領域
- エラークラス
- エラーコード

さまざまなエラーコードの意味については，11.7 の NCDDE エラーメッセージに関する段落で説明されています。

NCK との接続の切断

接続が切断されている場合，NCDDE サーバはアクティブトランザクション Request，Poke，および Execute に ' 否定応答 ' します。接続が切断されている限り，以降のトランザクションの実行は拒否されます。同時に，サーバは切断されている CNC との接続の再開を試行します。接続の状態は，サーバのローカル変数 NcState によって示されます。

Advise Link の処理

Advise Link 接続が切断されている場合，NCDDE サーバによって戻される値は，文字 '#' になります。Advise Link は，接続の再確立後に NCK に復元されます。

NCDDE サーバ内のリソースの不足

NCDDE サーバで使用するリソースが不足している場合，エラーコードが示されて，DDE インタフェースの影響を受けているトランザクションが終了します。

8.11 ネットワーク経由のアクセス用の NCDDE サーバの構成方法

ベータ版 :

これは，開発環境でのみ使用できる機能です。製品リリースでは使用できません。

概説

MMC 領域では，オペレーティングシステム Windows95 が使用されています。このため大抵は，PC ネットワークに接続したどの Windows PC からも NCK のデータにアクセスすることが可能です。

注 :Windows95 では，次の点に注意してください。
フォルダ "AUTOSTART" に，プログラム "¥WINDOWS¥NETDDE.EXE" へのリンクが含まれている必要があります。

840DI は，ハードウェアとソフトウェアにより，Windows ネットワークに組み込まれていると想定されます（図 8.3）。

これは以下のステップにより行われます。

- ISA スロットに挿入されたネットワークアダプタ，またはシリアルポート上のネットワークアダプタに物理的に接続する
- Windows を経由してネットワークに接続する
- Network DDE Share Manager を使用して，840DI のファイル SYSTEM.INI のセクション［DDEShares］にエントリを追加する
- Windows PC のファイル MMC.INI，REGIE.INI を適切に変更する
- 840DI を開始する
- Windows PC を開始する
- 接続をテストする

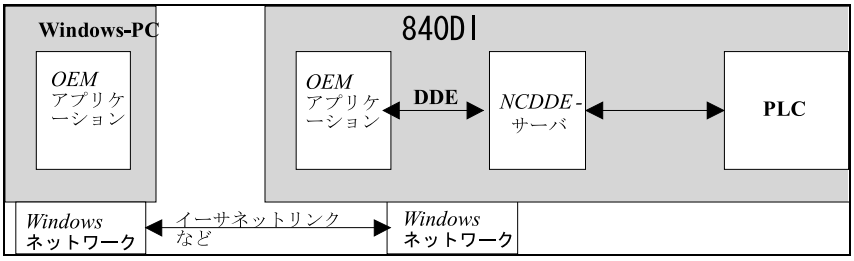


図 8.3 840DI のネットワーク

用途

PC 上で，840DI ユーザインタフェースを実行できます。また，独自のアプリケーションを実行したり，"NETDDE" 経由で NCK のデータにアクセスすることもできま

す。

840DI の構成

NCDDE サーバは，コマンド "New Share" と表 8.23 の項目を使用して，"Network DDE Share Manager" "¥MMC2¥DDESHARE.EXE" 経由で Windows ネットワークに宣言されなければなりません。プログラム " DDESHARE.EXE" は，OEM パッケージで提供されています。

次のような構成を使用します。

表 8.22 DDEShare のネットワーク構成

ID	エントリ	意味
共用名	NCU840D$	840DI の ID。この ID によりネットワークで識別される。
アプリケーション名	ncdde	ネットワークアクセスを経由してシステムで使用可能となるアプリケーションの名前。
トピック名	NCU840D	DDE 接続を確立するための名前の一部。（PC のファイル MMC.INI 内の NcddeMachineName で定義されたものと同じでなければならない）
アクセスタイプ	Full	パスワードロックなしの書き込み／読み取りアクセス。

注 : 次の例は Windows 3.x にのみ適用されます。Windows 95 では，これはプログラム "DDESHARE.EXE" により，レジストリに入力されます。

Share Manager が終了すると，以下の行がファイル "SYSTEM.INI" のセクション［DDEShares］に入力されます。

DDEShares

ファイル SYSTEM.INI のセクション［DDEShares］のエントリの例を示します。

■ 例 8-53 DDEShares

```
[DDEShares]
MMC2HW0$=ncdde,NCU840D,,15,,0,,0,0,0
```

注 : エントリ " 共用名 " はオプションです（この場合は NCU840D$）。エントリ " アプリケーション名 " と " トピック名 " は，840DI のファイル MMC.INI のセクション［GLOBAL］にあるエントリ "NcddeMachineName"，"NcddeServiceName" と同一でなければなりません。

840DI を再始動すると，ネットワーク経由での NCDDE サーバへのアクセスが可能になります。

Windows PC の構成

Windows PC では，NCK データは Windows ネットワークを経由して，実行中の MMC OEM アプリケーションと通信すると予期され，ファイル

MMC.INI

REGIE.INI

は，次のように変更しなければなりません。

MMC.INI のエントリ

ファイル MMC.INI のセクション［GLOBAL］に，次のエントリを作成する必要があります。

■ 例 8-54 ファイル MMC.INI のエントリ

```
[GLOBAL]
NcddeMachineName=NCU840D$ ; this is the Share Name
NcddeServiceName=¥¥SIN840D¥NDDE$
; this is the computer's name
```

REGIE.INI のエントリ

以下のエントリ

Startup2 = name := ncdde

が，Windows PC 上のファイル REGIE.INI のセクション［StartupConfiguration］でコメント化されている場合，独自の NCDDE サーバは，必ず開始が妨げられてしまいます。なぜなら，840DI システムの NCDDE サーバ（独自のものでない）は，ネットワーク経由で使用されるはずだからです。

■ 例 8-55 REGIE.INI でのスタートアップエントリのコメント化

```
[StartupConfiguration]
; do not start the own NCDDE Server
; Startup1 = name := ncdde, Timeout := 20000
```

サーバのシーケンスの開始

ネットワーク経由で NCDDE 通信が開始される前に，クライアントシステム（この場合 840DI）上にある NCDDE サーバが開始されなければなりません。その後，Windows PC を再始動すると，ファイル MMC.INI と REGIE.INI に加えられた変更が有効になります。これに続いて，ローカル NCDDE サーバアクセスに関する同じ命令構文を使用することができます。

接続のテスト

プログラム ‘NCDDE test’ を使用して，外部 Windows PC から 840DI の NCDDE サーバへの正確なアクセスをチェックすることができます。これを行うには，エントリ "Service/Topic" を "¥¥SIN840D¥NDDE$|NCU840D$" に設定しなければなりません（そして当然，NCDDE サーバを 840DI で実行していなければなりません）。

ネットワーク内の複数の MMC

いくつかの 840DI がネットワーク経由でアドレッシングされる場合，これらには，DDESHARE.EXE を使用して，別々の共用名を割り当てなければなりません。

8.12 NCDDE サーバの拡張

8.12.1 複数変数サービス

概説

複数変数サービスを使用すると，1 つの NCDDE ジョブ内の複数の変数にアクセスできます。そのため，複数ある特定の変数のアクセス速度が上がります。これは読み取りアクセスおよび書き込みアクセス専用です（ホットリンクは不可です）。

項目を指定する際には，対応するアクセス先の特定変数／配列の項目と同様に，'|' で区切ります。読み取りアクセスして得られるデータは，配列にアクセスする際に密に圧縮されます。形式を指定して配列にアクセスする場合や，新しいアクセス変更を指定して（8.12.3 を参照）アクセスする場合などは，その前に区切り文字をパラメータ化する必要があります。書き込みアクセスの際に，書き込まれるデータの先頭文字は，さまざまなデータブロックの区切り文字として解釈されます。

制限

- 個々のジョブには，最大 8 つの密に圧縮された PDU が含まれます。これにより，普通は 1 つのジョブ内の 100 を超える変数にアクセスできます（正確な数は試行して判別できます）。
- PDU は宛先アドレスに送信されます。したがって，PLC アクセスと NC アクセスが 1 つのジョブに混在することはありません。さらに，別々のチャネル内のチャネル固有変数へのアクセスも混在しません（NC の要件）。ドライブ固有の変数に対するアクセスにも同じことが当てはまります。
- 複数変数サービスでは，実変数だけをアドレッシングできます（BTSS インタフェース／ PLC-BUB）。日付，時刻，システム状態のリスト，ディレクトリ情報などはアドレッシングできません。
- DDE 項目のサイズは 255 文字に限定されていることに注意してください。項目のストリングがこの限度を超えている場合は，間接的に指定する必要があります（下記参照）。

複数変数サービスを使用した書き込みと読み取りの例　項目 :

channel/parameter/r[1,2](|"!l%ld")|/channel/parameter/r[10](|)

データ例 : |1|2|10.000000

8.12.2 間接的に項目を指定する

間接的に項目を指定すると，255 文字より長い項目を使用できます（最大 4KB）。NCDDE ローカル変数の内容は，DDE アクセス用の項目として使用されます。この場合，ローカル変数の名前は，'>' 文字を前に付けた項目として指定しなければなりません。

R10 アクセスの場合の例 :

Exec: NEW(x,"/channel/parameter/r[10]")

項目 :>x

データ例 : 10.000000

注 : NCDDE サーバでの変数書き込み用やコマンド実行用のデータ長は，4KB に限定されています。この値を超えると，エラー 0X01050414 になります。

8.12.3 新しいアクセス変更

以下の文字を括弧で囲んで項目ストリングに追加できます。

'|' ICF_TEXT 読み取りアクセスに関する個々の単一項目の前に '|' シンボルを挿入します。

これは書き込みアクセスの場合は評価されません。8.12.1 にある複数変数サービスの例を参照してください。

'^' このタグが付いた変数には，ホットリンクの非活動化は無効です（DEBA/DEBR）。

8.13 グローバルユーザ変数 GUD，SGUD，MGUD，UGUD，GD3 ～ GD9 へのアクセス

概説

グローバルユーザ変数は，NCK でも，各チャネルでも使用可能です。

NCK 固有のグローバルユーザ変数が，コントロールにつき 1 つのインスタンスに存在します。それらの変数は，チャネルから独立した設定と，チャネル間のプログラム調整で使用されます。

チャネル固有のグローバルユーザ変数が，各チャネルに存在します。それらの変数は，チャネル固有の設定と，1 つのチャネルで実行している異なるプログラム間のデータ転送で使用されます。

同じことがローカルユーザデータについても当てはまります。

まず初めに，ユーザ変数を定義し，アクティブにします。その後，NCDDE サーバはそれらの変数にアクセスできます。変数をクラスタ化する場合は，対応する NSK ファイルを作成し，組み込む必要があります。これを行うには 5 つのステップがあります。

(1) 定義ファイルを作成する
(2) この定義ファイルを NCK のディレクトリ /_N_DEF_DIR にコピーする
(3) INITIAL.INI ファイルをロードして，ユーザデータを *.ACC ファイルとしてアクティブにする
(4) MAP コマンドを使用して *.NSK ファイルを作成する
(5) 作成した *.NSK ファイルを NCDDE サーバの NSK ファイルに追加する

定義ファイル :

グローバルユーザ変数は，定義ファイル（モジュール）の中で，次の 5 つの名前を使用して定義しなければなりません。

- _N_GUD_DEF （GUD の場合）
- _N_SGUD_DEF （GD1 = SGUD の場合）グローバルデータ，当社用
- _N_MGUD_DEF （GD2 = MGUD の場合）グローバルデータ，工作機械メーカー用
- _N_UGUD_DEF （GD3 = UGUD の場合）グローバルデータ，ユーザ用
- _N_GUD4_DEF ～ _N_GUD9_DEF （GD4 ～ GD9）

これらのファイルは，NCK のディレクトリ /_N_DEF_DIR に保管しなければなりません。
グローバルデータを定義するファイルの総数は，対応するマシンデータ 18118（MM_NUM_GUD_MODULES）に依存します。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

グローバルデータは次のものにより定義されます。

- 定義ヘッダー： DEF

- 領域： NCK または CHAN
- タイプ： REAL，INT など
- 変数名： LIFTOFF_DIST など
- 寸法： 大括弧で囲む
- コメント： セミコロンで始まる任意指定テキスト

詳細は，840D Programming Guide を参照してください。

定義ファイルの作成

定義ファイルは，NCK または MMC で作成することができます。

NCK での作成 :
NCK のパートプログラムレベルで作成できるグローバル変数の定義ファイルは，ディレクトリ /_N_DEF_DIR になければなりません。それには次のものが含まれます。

- 最初の行に，プログラム ID
- パス指定したコメント行（評価される）
- 定義
- 終了命令 M02，M17，または M30

■ 例 8-56 NCK でのグローバル変数の定義

```
%_N_MGUD_DEF
; $PATH=/_N_DEF_DIR
DEF NCK REAL LIFTOFF_DIST ; defining a global variable
DEF CHAN INT TABLE[100] ; defining a channel-specific array
DEF CHAN REAL BLF_OFFS_X ; defining a channel-specific variable
M17 ; terminate this line with RETURN
```

MMC での作成 :

MMC で作成できる，MGUD.DEF というファイル名のグローバル変数の定義ファイルは，ディレクトリ C:¥TMP にあり，次のもので構成されます。

- 定義
- 終了命令 M02，M17，または M30

■ 例 8-57 MMC でのグローバル変数の定義

```
DEF NCK REAL LIFTOFF_DIST ; defining a global variable
DEF CHAN INT TABLE[100] ; defining a channel-specific array
DEF CHAN REAL BLF_OFFS_X ; defining a channel-specific variable
M17 ; terminate this line with RETURN
```

注 :MMC は，ドメインサービス COPY_TO_NC を使用して，NCK のディレクトリ /_N_DEF_DIR にこのファイルを転送しなければなりません。
COPY_TO_NC(C:¥TMP¥MGUD.DEF,/NC/_N_DEF_DIR/_ _N_MGUD_DEF,trans)

ユーザデータの活動化

INITIAL.INI ファイルを NCK にコピーすることにより，ユーザデータをアクティブにします。

ファイルは，非常に短いこともあります。M17 の後に RETURN を入力するだけで十分だからです。次の例はディレクトリ C:¥TMP にあるファイル INITIAL.INI に当てはまります。

COPY_TO_NC(C:¥TMP¥INITIAL.INI, /NC/_N_INITIAL_INI,_ trans)

これにより，NCK に次の 2 つの ACC ファイルが生成されます。

_N_NCK_GD2_ACC ：グローバルユーザ変数用
_N_CH_GD2_ACC ：チャネル固有のユーザ変数用（MGUD = GD2 を使用した上記の例に当てはまる）

注 : これによって NCK の静的メモリが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，すべてのプログラム，フレーム，マシンデータのバックアップを取ってください。

NCK 用の NSK ファイルの作成

MAP コマンドを呼び出すと，ACC ファイルから，グローバルユーザ変数用の対応する NSK ファイルが作成されます。これらのファイル名は，ACC ファイルの名前と同じです。

例は Visual Basic での呼び出しを示しています。

"MAP_ACC_NC" コマンドの呼び出し

C:¥MMC2¥MGUD_NCK.NSK: Windows 環境のファイル名
/NC/_N_NCK_GD2_ACC : NC ドメイン
trans : 変数 TransferState
0: 領域 NCK
2D : モジュール形 MGUD
10: トランザクションの時間制限 : 10s
/ACC/NCK/MGUD : 変数名の接頭部として使用される任意のストリング

■ 例 8-58 NCK-GUD 用の NSK ファイルの作成

```
Sub Form_Load ()
Label1.LinkTopic = "NCDDE|MMC2HW0"
Label1.LinkMode = 2
Label1.LinkExecute "MAP_ACC_NC(C:¥MMC2¥MGUD_NCK.NSK,
_ /NC/_N_NCK_GD2_ACC, trans, 0,
_ 2D , 10, /ACC/NCK/MGUD/)"
End Sub
```

チャネル用の NSK ファイルの作成

MAP コマンドを呼び出すと，ACC ファイルからチャネル固有のグローバルユーザ変数用の対応する NSK ファイルが作成されます。これらのファイル名は，ACC ファイルの名前と同じです。

例は Visual Basic での呼び出しを示しています。

"MAP" コマンドの呼び出し
C:¥MMC2¥MGUD_CH.NSK: Windows 環境のファイル名
/NC/_N_CH_GD2_ACC : NC ドメイン
trans : 変数 TransferState
2 : 領域チャネル
2D : モジュール形 MGUD
10 : トランザクションの時間制限 : 10s

/ACC/CH/MGUD : 変数名の接頭部として使用される任意のストリング

■ 例 8-59 チャネル固有 GUD 用の NSK ファイルの作成

```
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），ASCII フォーマット（*.NSK）の両方の形式で生成されます。

NSK を NCDDE サーバのファイルにマージする

この例のファイル MGUD_NCK.NSK，MGUD_CH.NSK を，NCDDE サーバ NCDDE311.NSK の NSK ファイルにマージするには，以下のようにします。

```
REM IMPORT ADDITIONAL USER VARIABLES
CALL(MGUD_NCK.NSK)
CALL(MGUD_CH.NSK)
REM
```

NCK ユーザ変数へのアクセス

以下の例では，NCK ユーザ変数 LIFTOFF_DIST が NCK からどのように読み取られるかが示されています。

NCK ユーザ変数 LIFTOFF_DIST の読み取り

■ 例 8-60 NCK ユーザ変数の読み取り

```
Sub Form_Load
CtlName1.LinkTopic = g_chNCDDEServiceName
CtlName1.LinkItem = "/acc/nck/mgud/LIFTOFF_DIST"
CtlName1.LinkMode = 2
CtlName1.LinkRequest
CtlName1.LinkMode = 0
End Sub
```

チャネル固有のユーザ変数へのアクセス

以下の例では，チャネル固有のユーザ変数を読み取る方法が示されています。

第 2 チャネルの BLF_OFFS_X

■ 例 8-61 チャネル固有のユーザ変数の読み取り

```
Sub Form_Load
CtlName.LinkTopic = g_chNCDDEServiceName
CtlName.LinkItem = "/acc/ch/mgud/BLF_OFFS_X[u2]" '2nd channel
CtlName.LinkMode = 2
CtlName.LinkRequest
CtlName.LinkMode = 0
End Sub
```

注 : ユーザデータを作成し，適用する方法についての詳細は，Installation and Start-up Guide /IAD/ と Programming Guide /PA/ を参照してください。

8.14 変数のオンラインヘルプ

概説

変数に関するオンラインヘルプは，OEM プログラマが NCK 領域からデータを選択し，定義するのをサポートします。その構造は，Windows の他のすべてのヘルプファイルと同じで，同じ特長があります。変数に関するオンラインヘルプは OEM パッケージ MMC から独立しており，ディレクトリ MMC2 にヘルプファイル BTSS_VAR.HLP として保管されています。

ターゲットシステム

変数に関するオンラインヘルプの使用は，840DI の OEM プログラマだけに限られません。このオンラインヘルプは，MMC 100 や PLC プログラミング環境の NC-Var Selector をカスタマイズする際にも使用できます。

特長

変数に関するオンラインヘルプには，11 章にリストされている NCK 変数，またリストブック /LIS/ で詳細が説明されている NCK 変数すべてに関する情報が提供されています。

いくつかの記述レベルを使用して，特殊変数に関する情報に到達することができます。
データ領域から開始する :
データエリア，モジュール，変数，例
モジュールを使用してアルファベット順に開始する :
モジュール，変数，例
または，関数 SEARCH（FIND）を使用してキーワードを検索する

キーワードは次のとおりです。
変数の短い記述：spindle type など
変数の名前：variable spindleType など
モジュールの短い記述：SSP（主軸状態データ）など

データのコピー

表示されたヘルプトピックから一部をコピーし，それを他のファイルにマージすることができます。これは特に，変数に関するオンラインヘルプの例を独自の OEM プログラムに挿入する場合に役立ちます。
これを行うには，次のようにします。
メニュー［Edit］を選択する
項目［Copy］を選択する
マウスを使用して，必要なテキストを選択する
［Copy］をクリックする
別のアプリケーションに切り替える
テキストを挿入する

他の特長

変数に関するオンラインヘルプを使用して，次のことも行えます。

- トピックを印刷する
- 独自のコメントを各トピックに挿入する
- 最も頻繁に参照する情報を早く見つけるためにブックマークを定義する

注 : 変数に関するオンラインヘルプのコメントは，ファイル BTSS_VAR.ANN（ANN は annex の略）に保管され，ブックマークは Windows ディレクトリにあるファイル WINHELP.BMK（BMK はブックマークの略）にあります。

8.15 トラブルシューティング

次の節では，操作時に起きる可能性のある問題を説明し，それらの解決方法を示します。

NCK/PLC の接続が切れる

- 接続ケーブルをチェックする
- MPI ドライバのインストールをチェックする
- MMC.INI をチェックする
- WINSTART.BAT
- S7DPMPI.INI

DDE-Initiate に応答しなかった

- LinkTopic をチェックする
- Linkitem をチェックする
- 変数は宣言されているか（特に PLC アクセス），またデータモジュールは宣言されているか。

ホットリンクをたくさん確立したためフォームのロードが長くかかる

- DCTL コントロールを使用する
- 非同期ホットリンクを確立する

最初の実行コマンドが機能しない

理由

いくつかのコマンドに対して，NCDDE サーバが NC への既存の接続を予期している。

解決方法

まず NC 変数へのホットリンクを確立する。

9 アラームの処理

概説

アラーム DDE サーバ（mbdde.exe）は，DDE プロトコルを介して，MMC に現在のアラームおよびシステムメッセージを提供します。これには，独自のグラフィカルユーザインタフェースはありません。アラーム／メッセージボックスの構成を行うなどの方法で，アラームを表示することができます。

システムアラーム／メッセージ

以下のアラーム／メッセージがシステムで示される場合があります。

- NCK アラーム
- ドライブアラーム
- サイクルプログラムアラーム
- PLC アラーム
- コンパイルサイクルアラーム
- PLC メッセージ
- MMC アラーム

特長

DDE アラームサーバは，メッセージに対して次のようなレジストリ機能を備えています。

- アラームの登録
- アラームの確認
- 照会機能 :
  - 最優先アラーム
  - 2 番目に優先されるアラーム
  - アクティブアラームの数
  - 発生したアラームの数
  - アクティブアラームのリスト
- ログファイルの記録
- 選択した言語でのアラームテキストの表示

9.1 通信

クライアント - サーバ

Windows アプリケーション（DDE クライアント）は，Windows が提供している DDE インタフェースを介して，アラームサーバと通信します。図 9.1 は，アラームサーバの機能の概要を示しています。

通信ジョブごとに，以下の指定が必要です。

- Service　：ドレッシングされる DDE サーバの名前
- Topic　：接続のトピック
- Item　：アクセスするデータ
- Mode　：接続のタイプ

DDE 通信の基本情報については，8.1 を参照してください。
詳細については，OEM 文書の 8.3 を参照してください。

Service および Topic は，初期設定ファイル mbdde.ini のセクション［mbdde］に入力しなければなりません。

以下の設定が DDE アラームサーバに適用されます。

Service = mbdde（mb はレジストリモジュールの短縮形）
Topic = alarms
Item = <string>（特定のサービスを識別する文字ストリング）

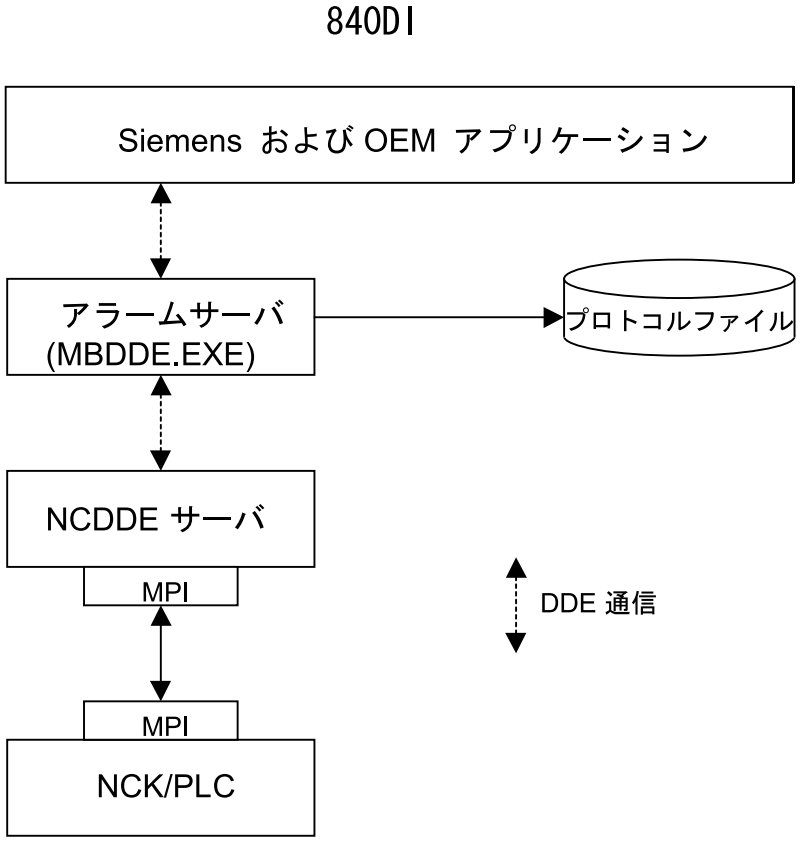


図 9.1 レジストリモジュールとの通信

VB でのプログラミング

Visual Basic では，Service および Topic の名前は特性 "LinkTopic" で結合されています。そこでは，パイプシンボル "|" で区切られた形になっています（たとえば，"mbdde|alarms"）。

DOS ボックスでのアラーム出力

ドライバは，DOS プログラムのアラーム出力に使用できます。これは，DOS ボックス内で実行され，ファイルにアラームメッセージを書き込みます。詳細については，6.7.5 を参照してください。

9.2 サービスのタイプ（DDE - Linkmode）

DDE アラームサーバでは，3 つのタイプのサービスが提供されています。これらは表 9.1 にリストされています。

表 9.1 アラームサーバのサービスのタイプ

リンクモード	意味
Execute	サーバからのサービスを要求する（たとえば，アラームメッセージの生成）
Advise（Hotlink）	特定のデータをモニタするようにサーバに指示する データが変更された場合，クライアントに通知されるか，データが自動的に更新される。
Request	サーバがあとで送信しなければならないデータを要求する

9.3 アラーム DDE インタフェースのサービス

概説

表 9.2 アラームサーバのサービス

章	内容
9.3.1	アラームサーバのコマンド
9.3.2	アラームサーバの Advise 変数
9.3.3	アラームサーバの Request 変数

9.3.1 アラームサーバのコマンド

これらのコマンドを実行する際，以下の点に注意してください。

- Visual Basic ラベル特性 LinkMode は，“手動を意味する 2“に設定しなければなりません。
- コマンド自体は，LinkExecute を使用して実行しなければなりません。

■ AlarmFree

説明

このコマンドは，MMC からアラームを消去します。

用途

アラームは，対応する確認変数が認識されていれば，どのようなアプリケーションからでも消去できます。

構文

AlarmFree (Quitvar)

引数

表 9.3 AlarmFree の引数

引数	構文	説明
Quitvar	<String>	確認変数の名前

例 9-1 確認変数テストに対応するすべてのアラームの消去（AlarmFree）

```
Sub Form_Load ()
    Label1.LinkTopic = "mbdde|alarms"
    Label1.LinkMode = 2
    Label1.LinkExecute "AlarmFree ( test )"
End Sub
```

■ AlarmMsg

説明

アラームを設定します。

用途

アプリケーションは，接続されている任意の NCU にアラームを設定できます。

構文

AlarmMsg(No,[Prio],[Var1...Var4],[TimeDate],Type,[NCU],Quitvar)

引数

表 9.4 AlarmMsg の引数

引数	説明
No	設定されるアラームの識別番号
Prio	アラームメッセージの優先順位（オプション）。メッセージはこの優先順位でリストに追加されます。各メッセージタイプの優先順位は，filembdde.ini で設定できます。
Var1 ...Var4	表示されるアラームストリングに挿入される変数の内容。最大 4 つまでの変数（%1 ～ %4）を指定できます（オプション）。
TimeDate	任意の時刻／日付（オプション）。引数 Time が指定されている場合，その値は，アラームが発生した時刻とみなされます。それ以外の場合，mbdde.ini が時刻を判別します。
Type	アラームのタイプ。1：PowerOn 2：Reset 3：Cancel 4：NC-Start
NCU	NCU の名前
Quitvar	認識変数

例 9-2 優先順位 100 のアラーム 1019 の設定（AlarmMsg）

```
Sub Form_Load ()
    Text1.LinkTopic = "mbdde|alarms"
    Text1.LinkMode = 2
    Text1.LinkExecute "AlarmMsg (1019,100,1 2 3 4,
                                        _ 26.07.99 ,1,NCU_1,quit1)
End Sub
```

9.3.2 アラームサーバの Advise 変数

タイプがストリングのこれらの変数は，サービス Advise および Request によってアクセスすることができます。

Advise サービスを使用する場合，以下の点に注意してください。

- ラベル特性（Visual Basic）：LinkMode は，1（自動）または 3（通知）に設定しなければなりません。

- 1 では，変数の値（表示されているものなど）が自動的に更新されます。
- 3 では，Warmlink はクライアントに対してデータの変更の通知のみを行います。新しい値を入手するには，クライアントは LinkRequest を使用してデータを読み取らなければなりません。

Request サービスを使用する場合，以下の点に注意してください。

- ラベル特性（Visual Basic）：LinkMode は，2（手動）に設定しなければなりません。
- LinkRequest を使用して変数の値を読み取ります。

■ AlarmList

説明

現在のアラームのリストが入ります。最初の '#' 文字の前の文字は，アクティブアラームの番号です。例 9-5 には，残りの部分が示されています。

用途

アプリケーションは，アラームリストにアクセスして，表示や編集を行うことができます。

構文

AlarmList

引数

なし

例 9-3 ラベル 1 のアラームリストの表示（AlarmList）

```
Sub Form_Load ()
    Label1.LinkTopic = "mbdde|alarms"
    Label1.LinkItem ="AlarmList"
    Label1.LinkMode = 1
End Sub
```

■ DialogAlarm

説明

該当するアラームがアクティブな場合に，ダイアログボックスアラーム（表示域 1）が入ります。

用途

アプリケーションは，現在アクティブなダイアログボックスアラームを表示することができます。

構文

DialogAlarm

引数

なし

例 9-4 ラベル 1 のダイアログアラームの表示（DialogAlarm）

```
Sub Form_Load ()
    Label1.LinkTopic = "mbdde|alarms"
    Label1.LinkItem = "DialogAlarm"
    Label1.LinkMode = 1
End Sub
```

■ FirstAlarm

説明

現在最も優先順位の高いアラームを入れます。

用途

アプリケーションは，Request を使用して最も優先順位の高いアラームにアクセスしたり，そのようなアラームを自動的に更新したりすることができます。

構文

FirstAlarm

引数

なし

例 9-5 Request を使用した最も優先順位の高いアラームの読み取り（FirstAlarm）

```
Sub Form_Load ()
    Label1.LinkTopic = "mbdde|alarms"
    Label1.LinkItem = "FirstAlarm"
    Label1.LinkMode = 2
    Label1.LinkRequest
End Sub
```

結果 :

12080#100#NCU_1: channel 1 block syntax error in text eal h #03.01.97 _ 20:57:08#0#12#NEW#5#NCU_1#hlp\alarm_GR.hlp#12080”

12080	アラーム番号
100	アラームの優先順位
NCU_1: channel...	NCU の名前およびアラームテキスト
03.01.97 ...	日時
0	表示域 “ メッセージ行 ”
12	確認変数 “12”
NEW	確認変数の内部値
5	アラームタイプ “NC-Start”
NCU	アラームの送信側 : “NCU_1”
hlp\alarm_GR.hlp	ヘルプファイルのパスおよび名前

12080	アラーム番号（再掲）

MZ1

説明

メッセージ行 1 の内容が入ります。

用途

アプリケーションは，メッセージ行 1 の現在の内容を表示することができます。

構文

MZ1

引数

なし

例 9-6 メッセージ行 1 の内容の表示（ラベルは**メッセージ行**（MZ1））

```
Sub Form_Load ()
    Meldezeile.LinkTopic = "mbdde|alarms"
    Meldezeile.LinkItem = "MZ1"
    Meldezeile.LinkMode = 1
End Sub
```

■ MZ2

説明

メッセージ行 2 の内容が入ります。

用途

アプリケーションは，メッセージ行 2 の現在の内容を表示することができます。

構文

MZ2

引数

なし

例

例 9-6 を参照

■ NrOfAlarm

説明

アクティブアラームの番号が入ります。

用途

アプリケーションは，インプリメントするアクティブアラームの番号を読み取ります（この番号の持続時間表示など）。アクティブなアラームがなければ，‘0’ が戻されます。

構文

NrOfAlarm

引数

なし

例 9-7 サービス Advise-Notify を使用した，ラベル “An-zAlarm” のアラームの番号の表示（NrOfAlarm）

```
Sub Form_Load ()
    AnzAlarm.LinkTopic = "mbdde|alarms"
    AnzAlarm.LinkItem = "NrOfAlarm"
    AnzAlarm.LinkMode = 3 'Advise-Notify
End Sub
Sub AnzAlarm_LinkNotify ()
    AnzAlarm.LinkRequest 'read with LinkRequest
    Print AnzAlarm.Caption
End Sub
```

9.3.3 アラームサーバの Request 変数

タイプがストリングのこれらの変数は，アラームサービス Request よってアクセスすることができます。

Request サービスを使用する場合，以下の点に注意してください。

- Bisual Basic の LinkMode は 2（手動）に設定しなければなりません。
- 変数の内容は，LinkRequest を使用して読み取ることができます。

■ AlarmSeqNr

説明

最後に電源投入したとき以降に発生したアラーム（確認済みも含む）の数が入ります。

用途

アプリケーションは，MMC の最後の電源投入以降に発生したアラームの数にアクセスすることができます。

構文

AlarmSeqNr

引数

なし

例 9-8 アラームの合計数の読み取り（AlarmSeqNr）

```
Sub Form_Load ()
    Label1.LinkTopic = "mbdde|alarms"
    Label1.LinkItem = "AlarmSeqNr"
    Label1.LinkMode = 2
    Label1.LinkRequest
End Sub
```

■ AlarmTextForID

説明

特定のアラーム識別番号に対応するアラームテキストの読み取り

用途

アプリケーションは，指定したアラーム番号に対応するアラームテキストを調べることができます。

構文

AlarmTextForID(alarmNo)

引数

表 9.5 AlarmTextForID の引数

引数	説明
alarmNo	対応するアラームテキストが戻されるアラームの識別番号

例 9-9 アラーム番号 “12080” に対応するアラームテキストの照会（AlarmTextForID）

```
Sub Form_Load ()
    abel1.LinkTopic = "mbdde|alarms"
    Label1.LinkItem = "AlarmTextForID(12080)"
    Label1.LinkMode = 2
    Label1.LinkRequest
End Sub
```

結果 :

012080 0 0 "channel %1 block %2 syntax error in text %3 "

012080	アラーム番号
0	表示域（ここでは，メッセージ行）
0	ヘルプフィールド（ここでは，File0）
“channel ...”	アラームテキスト

■ HelpForID

説明

アラーム番号に対応するヘルプファイルの番号が入ります。

用途

アプリケーションは，特定のアラーム番号に対応するヘルプファイルを見つけることができます。

構文

HelpForID(alarmNo)

引数

表 9.6 HelpForID の引数

引数	説明
alarmNo	ヘルプファイルを判別したいアラームの番号

■ 例 9-10 アラーム番号 1080 に対応するヘルプファイルの名前の読み取り（HelpForID）

```
Sub Form_Load ()
Label1.LinkTopic = "mbdde|alarms"
Label1.LinkItem = "HelpForID(1080)"
Label1.LinkMode = 2
Label1.LinkRequest
End Sub
```

9.4 アラームサーバの初期設定

ファイル

アラームサーバ（レジストリモジュール）は，以下のファイルを使用します。

MBDDE.INI
NETNAMES.INI
アラームテキストファイル

9.4.1 ファイル MBDDE.INI

説明

ファイル MBDDE.INI には，特に以下のセクションが含まれています。

表 9.7 ファイル MBDDE.INI のセクション

セクション	意味
Alarms	アラームリストに関する一般情報。 例 : レジストリエントリの日時の形式

TextFiles	アラームテキストリストのパスおよびファイル名。 例 : MMC=..\dh\mb.dir\alm_（ディレクトリ mb（レジストリモジュール）内の MMC アラームテキストの場合）
Helpcontext	ヘルプファイルの名前およびパス。 例 : File0=hlp\alarm_
DEFAULTPRIO	さまざまなアラームタイプの優先順位に関するデフォルト定義。 例 :POWERON=100
PROTOCOL	ログファイルの特性。（例 :file=.¥proto.txt，ログファイルの名前およびパス。）
KEYS	アラームの消去に使用できるキーに関する情報。 （例 :Cancel=+F10，キーの組み合わせ Shift+F10 によるアラームの消去）

■ セクション［Alarms］

説明

このセクション内の設定で，アラームリスト内のさまざまな特性を決定します。

例 9-11 ファイル MBDDE.INI 内のセクション［Alarms］

```
[Alarms]
TimeFormat=%d.%m.%y %H:%M:%S
MaxNr=50
ORDER=LAST
PLCTIME=5000
```

TimeFormat

ここでは，日時の表示に使用される形式を入力します。これは，Microsoft Foundation Classes（詳細については，"Microsoft Reference Volume I,Class Library Reference For the Microsoft Foundation Class Library" を参照）用に定義されている，CTime::Format に従っています。

MaxNr

アラームリストの最大サイズを決定します。

ORDER

アラームがアラームリストに挿入される順序を決定します。オプション FIRST を指定すると，新しいアラームがリストの上部に置かれます。オプション LAST を指定すると，最新のアラームが下部に挿入されます。

PLCTIME

3.2 より前の PLC ソフトウェアリリースに使用される，当社内部項目

注 : PLCTIME のエントリは変更しないでください。

■ セクション［TextFiles］

説明

このセクションでは，アラームに使用するテキストファイルの名前およびパスを設定します。サーバはこれらのリストから，言語依存のヘルプテキストを読み取ります。

テキストリストは，以下のように記述します。

message source_language.com

例 : 英語の MMC メッセージは，ファイル alm_gb.com に入っています。

識別子である alm_（INI ファイルで指定する）は，選択した言語に従って，レジストリモジュールにより自動的に拡張されます。対応するファイルがオープンされます。

■ 例 9-12 ファイル MBDDE.INI 内のセクション［TextFiles］（デフォルト）

```
[TextFiles]
=¥dh¥mb.dir¥alm_ ;        MMC アラーム
NCK=¥dh¥mb.dir¥aln_ ;     NCK アラーム
PLC=¥dh¥mb.dir¥alp_ ;     PLC アラーム
ZYK=¥dh¥mb.dir¥alz_ ;     サイクルアラーム
CZYK=¥dh¥mb.dir¥alc_ ;    コンパイルサイクルアラーム
serMMC= ;                 ユーザ定義の MMC アラーム
UserNCK= ;                ユーザ定義の NCK アラーム
UserPLC= ;                ユーザ定義の PLC アラーム
UserZYK= ;                ユーザ定義のサイクルアラーム
UserCZYK=;                ユーザ定義のコンパイルサイクルアラーム
```

注 **:** UserMMC，UserNCK，UserPLC，UserZYK，および UserCZYK で指定されるファイル内のアラームテキストは，MMC，NCK，PLC，ZYK，および CZYK で定義されるファイルの対応するテキストを多重定義します。
そのため，ファイルの変更はシステムの必要に応じて，もっぱら UserMMC，UserNCK，UserPLC，UserZYK，および UserCZYK の方で行わなければなりません。

外部生成されたメッセージテキスト

DOS エディタを使用してメッセージやアラームテキストを作成すると，ä，ö，ü のような特殊文字に関する問題が生じることがあります。OEM と ANSI では文字セットが違うことが原因です。

Windows では通常，ANSI が使用されます。自動的に認識することはできません。

それでも DOS 生成ファイルをインポートする場合は，MBDDE.INI 内のテキストファイルの名前の後にブランク 1 個とストリング "DOS" を追加してください（大文字も小文字も使用できます）。すると，アラームサーバ MBDDE により，OEM から ANSI への変換が自動的に実行されます。

注 : パラメータ DOS の設定後やリセット後には，その影響を受けるテキストファイルの日付を更新しなければなりません（開いて保管するだけでよい）。更新しないと，システムによってパラメータの変更が通知されません。

例 9-13 ファイル MBDDE.INI 内のセクション［TextFiles］
（ユーザ定義のテキスト付き）

```
[TextFiles]

MMC=¥dh¥mb.dir¥alm_ ;     MMC アラーム
NCK=¥dh¥mb.dir¥aln_ ;     NCK アラーム
PLC=¥dh¥mb.dir¥alp_ ;     PLC アラーム
```

ZYK=¥dh¥mb.dir¥alz_ ;	サイクルアラーム
CZYK=¥dh¥mb.dir¥alc_ ;	コンパイルサイクルアラーム
UserMMC= ;	ユーザ定義の MMC アラーム
UserNCK= ;	ユーザ定義の NCK アラーム
UserPLC=	c:¥dh¥mb.dir¥myplc_ DOS ; ユーザ定義の PLC アラーム ;（DOS ファイル）
UserZYK= c:¥dh¥mb.dir¥mycyc_ ;	ユーザ定義のサイクルアラーム
UserCZYK= ;	ユーザ定義のコンパイルサイクルアラーム

■ セクション［Helpcontext］

説明

このセクションでは，WinHelp 形式のヘルプテキストが入っているヘルプファイルの名前およびパスを指定します。このリストには，最大 10 までのエントリを含めることができます。

ヘルプファイルはテキストリストと同じ方法で，以下のように記述します。

messagesource_language .hlp

例 : 英語の OEM メッセージのヘルプテキストは，ファイル oem_gb.hlp にあります。

識別子である hlp¥OEM_（INI ファイルで指定する）は，選択した言語に従って，メッセージモジュールにより自動的に拡張されます。

対応するファイルがオープンされます。

例 9-14 ファイル MBDDE.INI 内のセクション［Helpcontext］

```
[Helpcontext]
File0=hlp¥alarm_
File1=hlp¥oem_
...
File9=hlp¥xyz_
```

ファイル

ヘルプ ID およびアラーム番号に対応するヘルプファイルの名前は，‘HelpForAlNr’ 機能を使用して判別できます。

アラームテキストファイルには，以下の形式のデータが入ります。

100001 0 0 ‘Function has not yet been implemented !’

ここで，
アラーム番号 **: 100001**
表示位置 **: 0**
ヘルプファイル **ID: 0**
アラームテキスト **: ‘Function has not yet been implemented !’**

ヘルプファイル ID 0 は，ヘルプテキストを見つけるためにスキャンされるファイルの指標を指定しています。この場合は，File0 です。エントリ File0 には，ヘルプファイルを割り当てなければなりません。

例 9-14 では，ディレクトリ ‘hlp’ にあるファイル "alarm_gb.hlp" がそれに当たります（英語が選択されている場合）。

MBDDE サーバによって置換が行われます。呼び出しを行うアプリケーションは，ヘルプファイルの名前および HelpID（今後はアラーム番号と同じです）で指定されます。

■ セクション［DEFAULTPRIO］

説明

各優先順位の値はデフォルトで '100' に設定されています。

アラームタイプの優先順位の変更は，INI ファイルの以下のエントリで行うことができます。

注 : 優先順位を変更すると，アラームの優先順位が低くなる場合があります。たとえば，Reset アラームの方が重要であり，重大な結果を招く場合でも，NC-Start アラームが Reset アラームより前に表示される場合があります。

例 9-15 ファイル MBDDE.INI 内のセクション［DEFAULTPRIO］

```
[DEFAULTPRIO]
MMC=100
CANCEL=100
RESET=100
POWERON=100
NCSTART=100
PLC=100
PLCMSG=10
```

■ セクション［PROTOCOL］

説明

このセクションでは，特定のエラーメッセージの登録方法を指定するプロトコルのさまざまな特性を定義します。

例 9-16 ファイル MBDDE.INI 内のセクション［PROTOCOL］

```
[PROTOCOL]
File=.¥proto.txt
Filter=Mode!1
Records=20
RecLen=300
FlushTime=10
DiskCare=0
```

ファイル

ログファイルのパスおよび名前

フィルタ

記録されるアラームメッセージの選択基準

以下の表記を使用できます。

[IDENTIFIER][RELATION][CHARACTERISTICS][OPERATORS]

IDENTIFIER:

- No　アラーム番号
- Prio　優先順位
- Mode　メッセージ／アラーム行またはダイアログボックス
- Type　アラームタイプ（PowerOn，Cancel など）
- From　アラームの送信側
- Quitvar　確認変数

RELATION:

- ':'　等しい
- '<'　より小さい
- '>'　より大きい
- '!'　否定

CHARACTERISTICS:

- 番号
- ストリング

OPERATORS

- ','　コンマは，フィルタの内部でのみ論理 OR を表します。
- ' '　スペースは，異なるフィルタの間で論理 AND を表します。
- '|'　パイプは，異なるフィルタの間で論理 OR を表します。

例 : **Filter=Typ<3**
POWERON と RESET アラームだけが登録されます。

例 : **Filter=From:NCU_1**
NCU_1 アラームだけが登録されます。

例 : **Filter=From:NCU_1 Typ:1,3**
NCU_1 からの POWERON と CANCEL だけが登録されます。

Record

ログファイル内のエントリの数を表します。実際の数がこの値を超えると，サーバは古いエントリのオーバーライドを開始します（原則 : リングバッファ）。

RecLen

エントリの長さ（バイト単位）

FlushTime

バッファ付き入出力が使用されている場合，この期間（秒単位）が経過すると，バッファの内容がログファイルに送られます。

DiskCare

メッセージモジュール MBDDE の場合，現リリースではアラームプロトコルをハードディスク（ファイル mmc2\proto.txt）に書き込むかどうか，およびいつ書き込むかを構成できます。旧リリースでは，アラームが生じたり消えたりするたびに，アラームプロトコルがハードディスク上に書き込まれていました。現リリースでは，ファイル MBDDE.INI のセクション [PROTOCOL] 内のエントリ "DiskCare" により，プロトコルファイルがいつ書き込まれるかが制御されます。

以下のパラメータを設定できます。

DiskCare = -1（デフォルト）MBDDE サーバにより，メインメモリでアラームプロトコルが実行されます。プロトコルは，診断モードのソフトキーで指示された場合にのみ，ハードディスクに書き込まれます。これは，制御がオフにされると，それ以前にハードディスクに書き込まれていない限り，そのアラームプロトコルは使用できなくなるということも意味します。

DiskCare = 0 プロトコルファイルに変更を加えると，即時に格納されます（旧リリースの動作と同じ）。

DiskCare = n アラーム状態の変更内容は，変更が加えられなくなってから n 秒間経過するとプロトコルファイルに書き込まれます。

エントリ DiskCare はスタートアップ時だけ評価されます。

■ セクション［KEYS］

説明

以下のエントリを使用して，ファンクションキー（たとえば，Shift+F10）を取り消し機能に割り当てることができます。

例 9-17 ファイル MBDDE.INI 内のセクション［KEYS］

```
[KEYS]
Cancel=+F10
```

また，これにより，PC 開発システムで取り消しアラームを確認することもできます。

■ セクション［MmcAlarmDisable］

説明

ここに入力されている送信側の ID は使用不可にされ，AlarmMsg ジョブで受け入れられません。MBDDE.INI 内で，”PLC”，”NCU”，および ”MBDDE” は，デフォルトで使用不可です。

例 9-18 ファイル MBDDE.INI 内のセクション［MmcAlarmDisable］

```
[MmcAlarmDisable]
DisableSenderOfMmcAlarm0 = PLC
DisableSenderOfMmcAlarm1 = NCU
DisableSenderOfMmcAlarm2 = MBDDE
DisableSenderOfMmcAlarm3 = .....
```

エントリの最大数は 100（0 ～ 99）です。

OEM ユーザはこれらのエントリを適用して，特別な送信側の ID を使用不可にすることができます。

9.4.2 ファイル NETNAMES.INI

説明

ファイル NETNAMES.INI には，バスを使うすべてのもののののバスアドレスが含まれています。アラームサーバは，どの NCU が使用可能かを，それぞれの名前とともに判別する必要があります。そのため，アラームメッセージには NCU の名前を表示することができます。

例 9-19 NETNAMES.INI（抜粋）

```
[param NCU_1]
nck_address= 13
plc_address= 13
name=Standard Machine

[param NCU_2]
nck_address= 13
plc_address= 13
name=Test Maschine
```

9.4.3 アラームテキストファイル

説明

アラームテキストファイルの拡張子は必ず .COM です。以下に例を示します。

alm_gb.com ;　　MMC アラーム（英語）

これらのファイルは，データ管理ツリー \DH のディレクトリ \MB にあります。以下に例を示します。

```
C:¥DH¥MB¥alm_gb.com
```

言語固有のテキストファイルが，現在選択されている言語に応じてロードされます。

言語の省略形

言語 DLL およびアラームテキストファイルの名前では，言語は 2 文字の省略形で記述されます。

表 9.8 言語の省略形

言語	省略形
中国語（中華人民共和国）	CH
中国語（台湾）	TW
英語	GB
フランス語	FR
ドイツ語	GR
イタリア語	IT
韓国語	KO
スペイン語	SP

これらの省略形は，LanguageList の下のデータファイル MMC.INI のセクション

［LANGUAGE］にあります。

アラームテキストファイル

アラームテキストファイルは以下のような構造になっています。

表 9.9 アラームテキストファイルの内容

エントリ	意味
alarmNo	アラーム番号
display	0：ヘッダのアラーム／メッセージ行に表示 1：ダイアログボックスに表示
help-Id	アラームへのヘルプファイルの割り当て
text	ヘルプテキスト（最大 110 文字） ここには，ほかの任意のアラームの番号を入力することができます。その場合，そのアラームのヘルプテキストが表示されます。

アラームテキストファイルには，コメントを含めることができます。コメントの先頭には "\\" を付けなければなりません。

■ 例 9-20 ファイル ALN_GR.COM 内のエントリ

```
027011 0 0 "axis %1 Safe velocity exceeded "
```

9.5 840DI のアラーム領域

表 9.10 アラーム番号の領域

アラーム番号	アラーム領域	アラームテキストファイル
	NCK アラーム	
000 000 ～ 009 999	一般アラーム	ALN_xx.COM
010 000 ～ 019 999	チャネルアラーム	ALN_xx.COM
020 000 ～ 029 999	軸／主軸アラーム	ALN_xx.COM
030 000 ～ 039 999	機能アラーム	
040 000 ～ 059 999	予約済み	
060 000 ～ 062 999	サイクルアラーム（当社）	ALZ_xx.COM
063 000 ～ 064 999	予約済み	
065 000 ～ 067 999	サイクルアラーム（ユーザ）	
068 000 ～ 069 999	予約済み	
070 000 ～ 079 999	コンパイルサイクルおよびOEMアラーム	ALN_xx.COM ALC_xx.COM
080 000 ～ 099 999	予約済み	
	MMC アラーム／メッセージ	
100 000 ～ 109 999	MMC 100	
100 000 ～ 100 999	基本システム	
101 000 ～ 101 999	診断	
102 000 ～ 102 999	サービス	
103 000 ～ 103 999	マシン	
104 000 ～ 104 999	パラメータ	
105 000 ～ 105 999	プログラミング	
106 000 ～ 106 999	予約済み	
107 000 ～ 107 999	OEM	
108 000 ～ 109 999	予約済み	
110 000 ～ 119 999	MMC 101	
120 000 ～ 129 999	PCU50	ALM_xx.COM
130 000 ～ 139 999	OEM	
140 000 ～ 199 999	予約済み	
（200 000 ～ 299 999	MCU アラーム）	
300 000 ～ 399 999	ドライブアラーム	ALN_xx.COM
	PLC アラーム／メッセージ	
400 000 ～ 499 999	一般アラーム	ALP_xx.COM
500 000 ～ 599 999	チャネルアラーム	
600 000 ～ 699 999	軸／主軸アラーム	

アラーム番号	アラーム領域	アラームテキスト ファイル
700 000 ～ 799 999	ユーザ域	
800 000 ～ 899 999	実行チェーン／グラフ	ALP_xx.COM
810 000 ～ 810 009	PLC のシステムエラーメッセージ	ALP_xx.COM
900 000 ～ 999 999	予約済み	

xx = 言語の省略形

10 データ管理

概説

データ管理サーバ（DHServer.exe）を使うと，アプリケーションで Siemens データ管理へアクセスできます。ここでは，パートプログラムと工具類のデータが管理されています。Siemens の標準インタフェースでは，NCK のデータへアクセスするときも，このデータ管理を使います。

データ管理には，ファイルやディレクトリを扱うための，次のようなインタフェースが備えられています。

- 作成
- 削除
- コピー
- リスト表示
- ダウンロード（MMX から NXK へのコピー。ソースは削除される）
- アップロード（NXK から MMX へのコピー。ソースは削除される）
- アクセス許可の設定
- 補助機能の実行

10.1 一般情報

要約

データ管理は，システムでの実行時に読み込むことのできる，ハードディスクのデータ構造の記述（データスキーム）に基づいています。

データ管理サーバとの通信は，Windows - DDE（動的データ交換）- インタフェース（8-1 を参照）を経由して行われます。データ管理サーバは，NCK へのインタフェースとして NCDDE サーバを使います。

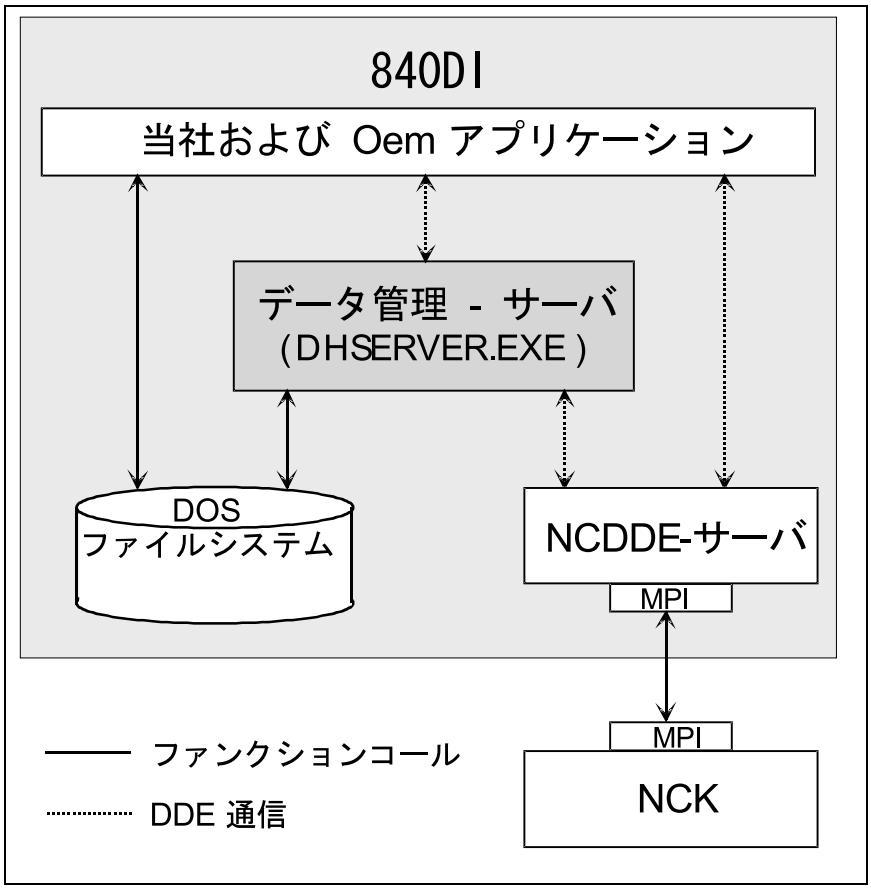


図 10.1 システム全体におけるデータ管理サーバ

呼び出しのタイプ

機能の複雑さに応じて，次の 3 タイプの呼び出しが実装されています。

- ジョブの要求 : データを戻す単純な要求
- 単純な実行ジョブ : データを戻さない単純な要求
- 複雑な実行ジョブ : データを戻す複雑な要求。これらのジョブは，長時間にわたって実行されることがある。

表 10.5 には，機能の概要が示されています。

10.2 データ管理のディレクトリ構造

ディレクトリツリー

MMC データ管理によって管理されるファイルはすべて，DOS ファイルシステムの中の特別なディレクトリツリーに編成されています。

このディレクトリツリーの構造は，データスキームで定義されています。DH.INI ファイルには，次のように，この構造の起点を示すパス（ROOT という）を入力する必要があります。

例 : C:¥DH

初期設定ファイル

DH.INI ファイルには特に，以下のセクションが含まれています（表 10.1）。

表 10.1 DH.INI ファイルのセクション

セクション	意味
DHSTART	ハードディスク上の MMC データ管理のルートディレクトリ。 例，mmchome=I:¥dh アクセス権限。 例，access level=4
SCEME	データスキーマのバイナリ部分の名前。例，sceme=schema.bin

データスキーム

データスキームは，ディレクトリ構造の定義のことで，ファイルを処理し管理するときに，データ管理によって考慮に入れられます。

次のもので構成されます。

- 1 つの要素（ファイル／ディレクトリ）の記述
- ディレクトリ構造の記述

したがって，データスキームでは，データ管理が MMC ハードディスクのディレクトリツリー内に作成できる要素の構造を記述しています。

MMC/NCK ファイルシステム

データスキームで記述されている要素は，MMC ハードディスクの DOS ファイルシステムに置くことができるように，NCK のファイルシステムに置くこともできます。

データスキームで記述されているディレクトリ構造には，NCK ファイルシステムに存在するディレクトリ構造のイメージが含まれています。

データ管理は，両方のシステムに含まれるファイルについての完全な概要と，それらのファイルに対する統一的なアクセスを提供します。

実行時には，データ管理の機能から，データスキームに示されている仕様にアクセスし，ファイルとディレクトリの構造の一貫性を保つことができます。

注 : 定義された構造の一貫性を検査できるのは，MMC ハードディスク上のファイルおよびディレクトリについてだけです。

10.2.1 ディレクトリツリーの要素の特性

データスキームで定義されたディレクトリ構造は，ディレクトリツリーの個々の要素を結合したものです。データ管理側の観点では，要素はファイルかディレクトリになります。

ディレクトリツリーの要素ごとに，データ管理の特定機能を使うことにより，次に示す情報を記録し，読み込むことができます。

データタイプの記述には，パートプログラム固有の特性についての記述が含まれます。

要素については，表 10.2 で示されているように，特性ごとに記述されます。

表 10.2 要素の特性

特性	意味
名前	ファイルまたはディレクトリの名前
拡張子	DOS ファイルシステムでの名前の拡張子
説明	要素を読める形式で表示するための識別子
データフォーマット	ディレクトリまたはファイル

名前

これは，ファイルまたはディレクトリをディレクトリツリーに格納するときに使う名前です。

この要素が NCK システムでのファイル／ディレクトリのイメージの場合で，このシステムが固定の名前を必要とする場合は，この名前がここで使われます。

作成するファイル／ディレクトリの名前が任意指定なのであれば，名前として ” * ” が使われます。

名前の最大長は 25 文字です。

拡張子

データスキームで定義したファイル／ディレクトリはすべて，DOS ファイルシステムでアーカイブできます。

この目的のため，スキームで，要素ごとに 3 文字の名前拡張子を定義する必要があります。

NCK ファイルシステムによって管理され，データ管理が扱う要素の一部であるファイル／ディレクトリには，すでに 3 文字の名前拡張子が付けられています。

したがって，NCK に格納される要素だけでなく，MMC ファイルシステムに格納される要素にも，NCK システムによって定義された拡張子が追加されます。NCK で扱われない要素の場合には，MMC ハードディスクだけで使われる拡張子が定義されます。

説明

要素ごとに，言語に応じた説明が記録されます。これにより，NCK で固定の名前が使われていれば，ファイル名またはディレクトリ名を，理解しやすい，言語に応じた方式で表示できるようになります。

例 10-1 説明の例

データスキームでの定義 : **可能な表示 :**

説明	名前	拡張子	
Werkstük	WPD	DIR	Werkstüke
Werkstük	.*	WPD	WELLE1

または :

parts	WPD	DIR	parts
part	*	WPD	SHAFT1

データフォーマット

データフォーマットでは，ディレクトリツリーの対象のタイプを，ファイルまたはディレクトリとするよう指定します。

保管場所

ディレクトリツリー内に新しいオブジェクトを作るときには，データ管理は必ず該当するアクセスマスクを使います。詳細については，この章の「アクセス許可」および「アクセスマスク」の節で説明されています。

完全なシステムでは，ファイルを保管できる場所として，次の 3 つがあります。

- ハードディスクの MMC ファイルツリー
- NCK ファイルシステムのデータ構造だけに入れる
- MMC ハードディスクと NCK ファイルシステムの両方

この保管場所は，データ管理の LIST 機能で得られる情報の一部です。次のような符号化が適用されます。

FM：ファイル MMC

DM：ディレクトリ MMC

FN：ファイル NCK

DN：ディレクトリ NCK

注意 : ファイルまたはディレクトリの保管場所を入力引数として指定する，データ管理の機能（例えば，COPY）は，指定した場所がデータスキームで定義した場所と一致する場合にのみ，正しく動作します。

アクセス許可

それぞれのファイルおよびディレクトリに適用されるアクセス許可には，8 つのレベルがあります。

表 10.3 アクセス許可の 8 つのレベル

アクセスレベル：	必要なもの：	ユーザグループ
S0	システムパスワード	当社
S1	工作機械メーカーのパスワード	工作機械メーカー
S2	保守用のパスワード	セットアップ／サービス担当員（工作機械メーカー）
S3	ユーザパスワード	特権ユーザ（企業内サービス）
S4	キースイッチ位置 3	プログラマ
S5	キースイッチ位置 2	訓練を積んだオペレータ
S6	キースイッチ位置 1	オペレータ
S7	キースイッチ位置 0	中級程度の技術のオペレータ（NC 開始／NC 停止，操作パネル）

アクセス許可のレベル 0 は最高の設定で，レベル 7 は最低の設定です。

システムの実際のアクセス許可は，現在のキースイッチ位置か，パスワードの入力によって設定されます。

アクセス権限

以下の権限をファイルに設定できます。

READ

WRITE

EXECUTE

SHOW

DELETE

アクセスマスク

データスキームでは，ディレクトリツリーの各要素にアクセスマスクが割り当てられます。ここでは，アクセス権限が，現在のアクセス許可レベルに応じて設定されます。

このようなアクセスマスクは，5 桁の数値で記述されます。

1 桁目	2 桁目	3 桁目	4 桁目	5 桁目
READ	WRITE	EXECUTE	SHOW	DELETE

アクセスマスクの例 : 7 4 7 7 5

アクセスレベル 0 ～アクセスレベル 4 で書き込めます。

アクセスレベル 0 ～アクセスレベル 5 で削除できます。

全アクセスレベルで読み込み，実行，表示が可能です。

要素のアクセスマスクは，データ管理の Get_Attributes 機能で判別し，SetAccess で変更することができます。

アクセス許可の変更

特定のアクセスレベルのアクセス権限は，上位の保護レベルを持っている場合に限り，変更することができます。これを行えるように，データ管理の SetAccess 機能が用意されています。

データスキームで事前定義されていて，グローバルな妥当性を持つアクセスマスクは，この機能で変更することはできません。

10.3 MMC データスキームの要素

次に示すファイルまたはディレクトリは，データ管理の機能を使って管理できます。

表 10.4 データスキームの要素

要素	拡張子	名前	フォーマット	内容／意味／使用方法
アクセスの記述	ACC	*	ファイル	実際に設定されるアクセス許可に依存する，特定の NCK 変数へのアクセスについて記述する
アーカイブ	ARC	*	ファイル	保存の目的で MMC ハードディスクに格納されたデータ
ブートデータ	BOT	*	ファイル	NCK とドライブを運転状態に設定するために必要なバイナリフォーマットのマシンデータ
クリップボード	CLP	CLIP	ディレクトリ	データ管理の一時保管ファイル用のディレクトリ
コメント	COM	*	ファイル	任意の ASCII テキストを含むファイル
補正データ	IKA	*	ファイル	補間された補正用データ
定義	DEF	*	ファイル	システム機能で定義として必要なデータ
ディレクトリ	DIR	*	ディレクトリ	ディレクトリツリー内の任意のディレクトリ
グローバルユーザデータ	GUD	*	ファイル	マシン固有のユーザデータ（GUD：Global User Data)
初期設定プログラム	INI	*	ファイル	各種の NCK または MMC データについてのマシン固有の設定（特定のタイプは指定しない）
マシンデータ	TEA	*	ファイル	NCK のマシンデータ（TEA：Testing Data Active）
NC データ	MDN	*	ディレクトリ	NC マシンデータのディレクトリ
OEM 固有のデータ	USR	*	ファイル	OEM ユーザが定義する任意のデータ
パートプログラム	MPF	*	ファイル	タイプが「メインプログラム」または「サブプログラム」である NC パートプログラム
パートプログラム固有／ローカルユーザデータ	LUD	*	ファイル	パートプログラムで定義されているローカル変数（LUD：Local User Data）
設定可能な原点のオフセットとフレーム（UFR：User Frame）	UFR	*	ファイル	設定可能な原点のオフセットとフレーム（UFR：User Frame）
設定データ	SEA	*	ファイル	設定データ（SEA：Setting Data Active）
サブプログラム	SPF	*	ファイル	NC サブプログラム：他のプログラムによって呼び出されるサイクルプログラムおよび全プログラム
システムデータ	SYF	*	ファイル	システム機能の実行に必要なデータ
工具補正	TOA	*	ファイル	工具補正（TOA：Tool Offset Active）
ワークディレクトリ	WPD	*	ディレクトリ	加工のワークを完全に記述するのに必要なすべてのファイル

これから，それぞれのデータタイプについて説明します。

アクセス記述

実際に設定されるアクセス許可に依存する，特定の NCK 変数へのアクセスについて記述します。

このタイプのファイルには，実際に設定されるアクセス許可に依存するアクセス記述（ACC = ACCess），つまり特定の NCK 変数にアクセスする方法が記されています。

このタイプのファイルは常に特定の NCK 変数を参照するため，事前定義された名前のファイルタイプがいくつか存在し，それぞれ関係するデータの部分を記述しています。

データ記述の名前と拡張子の組み合わせを参照することにより，データ管理側は NCK データに対応するアクセス記述を識別することができます。

例 :

C:¥MMC_OEM.V42¥MMC2¥IB¥BT_TEA.ACC

アーカイブ

保存の目的で MMC ハードディスクに格納されたデータです。

ブートデータ

NCK と 611D ドライブを運転状態に設定するために必要なバイナリフォーマットのマシンデータです。

ブートデータは，主軸および送りドライブをインストールするためのデータです。幾種類か存在する場合があります。各ドライブにはそれぞれのセットアップデータがあるため，ドライブごとに固定名のファイルタイプが定義されていて，それぞれ対応するデータが含まれています。

クリップボード

データ管理の一時保管ファイル用のディレクトリです。

スキームで定義された各ファイルタイプが完全なデータ構造の中で定義されたら，このディレクトリに格納できます。つまり，完全なディレクトリツリーをこの構造の下に 2 度目に格納できます。

このディレクトリは，データ管理によって一時記憶ディレクトリとして使われます。

コメント

任意の ASCII テキストを含むファイルです。

コメントファイルの内容は固有のものなので，複数のコメントファイルが，固定名を持つ特殊ファイルタイプとして定義されます。これらのファイルは，名前と拡張子の組み合わせによって，データ管理に識別されます。

補正データ

補間された補正用データです。

定義

システム機能で定義として必要なデータです。

ここでは，NCK に 1 度しか存在しないファイルタイプで，固定名を持つ別個のファイルタイプが定義されます。

ディレクトリ

データツリーの任意のディレクトリです。

特殊ディレクトリの一例は，ワークディレクトリです。

説明 : ワーク

拡張子 : DIR

名前 : WKS

フォーマット : ディレクトリ

属性 : -

このワークディレクトリでは，‘workpiece’ / WPD タイプのファイルだけを作成できます。

グローバルユーザデータ

マシン固有のユーザデータ（GUD：Global User Data）

このタイプ（GUD：Global User Data）のファイルでは，ユーザはパートプログラムやサイクルプログラムで使うときに必要な，マシン固有のデータを定義できます。グローバルに有効な R パラメータは，このカテゴリに属します。画面表示を構成することにより，これらのデータを表示し，MMC からそれぞれの値を入力することもできます。
この構造は，マシンデータのデータタイプに対応します。

初期設定プログラム

さまざまなタイプの MMC または NCK データのマシン固有の設定です。

初期設定ファイルには，さまざまなタイプの MMC または NCK データの設定が，特定のタイプに制限されることなく記されています。このモジュールは，ワークディレクトリに格納できます（ワークディレクトリを参照）。

マシンデータ

NCK マシンのデータです。

NC データ

NC マシンデータのディレクトリです。

このディレクトリには，特別な NCK 構成を復元するときに必要な情報を含むファイルを格納できます。

これらのファイルは，MMC ハードディスクに格納された NCK のアクティブファイルシステムのイメージを表すことがあります。

OEM 固有データ

OEM ユーザが定義する任意のデータです。

ここでは，固定名を持つ各種のデータタイプを定義することもできます。

パートプログラム

タイプがメインプログラムまたはサブプログラムである NC パートプログラムです。

この種のファイルは，タイプがメインプログラムかサブプログラムである NC パートプログラムです。
NCK では，メインプログラムとサブプログラムを区別しません。

注 : ホストコンピュータからコンピュータリンク経由で転送されたプログラムには，例外が発生します。この場合，ファイルが格納される対応ディレクトリを選択するときに，名前拡張子が使われます。

パートプログラム固有／ローカルユーザデータ

パートプログラムで定義されているローカル変数（LUD：Local User Data）です。

このタイプのファイルには，パートプログラムで定義されているローカル変数（LUD：Local User Data）が含まれます。

注 : これらの変数は，パートプログラムの該当モジュールの実行時にだけ存在します。しかし，これらをファイルとして格納し，データ管理を介してアクセス権を付与することができます。

設定可能な原点のオフセット／フレーム

設定可能な原点のオフセットとフレームです。（UFR：User Frame）

設定データ

設定データです。（SEA：Setting Data Active）

サブプログラム

NC サブプログラム : サイクルプログラムと，他のプログラムから呼び出されるすべてのプログラムです。
NCK では，メインプログラムとサブプログラムを区別しません。

システムデータ

システム機能の実行に必要なデータです。

工具補正

工具補正です。（TOA：Tool Offset Active）

ワークディレクトリ

加工のワークを完全に記述するときに必要なすべてのデータタイプです。

ワークディレクトリには，複数のファイルタイプを含めることができます。パートプログラム，設定データ，工具補正，初期設定モジュールなどを含む，完全な加工ワークで実際に必要なすべてのタイプを含められます。

データタイプ workpiece / WPD の特殊性

加工のために，NCK でのパートプログラムのようなワークを 1 つ選択することができます。

データスキームでは，この特性はアクセスマスクにおける EXECUTE 権限で表されます。
加工のためにワークを選んだら，そのワークと同じ名前の INI ファイルが NCK にロードされます。同時に，ワークと同じ名前を持つパートプログラムのメインモジュールも選択されます。
同じ名前のパートプログラム／ MPF が見つからなければ，エラーメッセージが送られ，以前に選ばれたパートプログラムが引き続き選ばれます。
同じ名前の INI モジュールが使えない場合，オペレータが明示的なアクションを行うことにより，他の初期設定ファイルを手動で実行することができます。

10.4 データ管理サーバの機能

概説

データ管理には，ファイルやディレクトリを扱うための，次のようなインタフェースが備えられています。

- 作成
- 削除
- コピー
- リスト表示
- ダウンロード（MMC から NCK へのコピー。ソースは削除される）
- アップロード（NCK から MMC へのコピー。ソースは削除される）
- アクセス許可の設定
- 補助機能の実行

データ管理サーバでは，実行時に読み込まれる，ハードディスク上のファイル構造の記述（データスキーム）を使います。
ソースファイルが削除されるため，アップロードとダウンロードは，MOVE コマンドに相当します。
データ管理の機能は，DDE サーバのインタフェース DHSERVER.EXE として実装されています。

呼び出しのタイプ

機能の複雑さに応じて，次の 3 タイプの呼び出しが実装されています。

- ジョブの要求 : データを戻す単純な要求
- 単純な実行ジョブ : データを戻さない単純な要求
- 複雑な実行ジョブ : データを戻す複雑な要求。これらのジョブは，長時間にわたって実行されることがある。

表 10.5 には，機能の概要が示されています。

10.4.1 データ管理サーバとの DDE 接続の確立

データ管理サーバとの DDE 接続を確立したい場合，開発者は次に示す構成要素に精通している必要があります。

- Link-Server：DDE サーバの名前
- Link-Topic：トピック
- Link-Item：アクセスするデータ
- Link-Mode：リンクモード

表 10.5 データ管理の機能（概説）

機能	タイプ	項目	意味
Activate	実行 : 複雑	---	ファイルまたはディレクトリを，同じディレクトリの NCK ファイルシステムにコピーする
Activate2	実行 : 複雑	---	Activate と同様だが，ソースファイルは削除されない

Best_Datatype	要求	名前，拡張子	指定した名前と拡張子に最も一致するデータタイプを探す
Cancel	実行 : 単純	---	すべてのジョブを打ち切り，結果変数を渡す
Convert_Possible_Datatypes	要求	データタイプ，拡張子	データファイルへコピーできるデータタイプをリストする
Copy	実行 : 複雑	---	ファイルソースをファイル宛先へコピーする
Create	実行 : 複雑	---	ファイルまたはディレクトリを作成する
Del	実行 : 複雑	---	ファイルまたはディレクトリを削除する
Exist	実行 : 単純	---	ファイルが NCK に存在するのか MMC に存在するのかを検査する
Extern	実行 : 単純	---	実行するファイルを外部から選ぶ
Get_Attributes	要求	dhpath	ファイルまたはディレクトリの属性を戻す
Get_Properties	要求	dhpath, モード	データスキームに基づいてファイルまたはディレクトリの特性を戻す
Get_Propertynames	要求	dhpath, モード	特性リストの特定ビットに，言語に応じたショートカットを提供する
Get_Realpath	要求	dhpath, モード	DOS ファイルまたは DOS ディレクトリ, あるいは NCK ファイルまたは NCK ディレクトリのパスを戻す
List	実行 : 複雑	---	データ管理のディレクトリの内容を読み込む
Passivate	実行 : 複雑	---	ファイルまたはディレクトリを，NCK から同じディレクトリの MMC ファイルシステムに転送する
Possible_Datatypes	要求	dhpath, モード	データ管理のディレクトリに格納できるデータタイプをリストする
Rename	実行 : 単純	---	NewName でデータファイルをリネームする
Select	実行 : 単純	---	NCK で実行するファイルを選ぶ
Set Access	実行 : 単純	---	ファイルまたはディレクトリの許可を設定する
Startsave	実行 : 単純	---	サーバに VarName というメモリ領域を確立する
Stopsave	実行 : 単純	---	メモリ領域 VarName を解放する

これらの機能については，次の章で詳しく説明されています。

10.4.2 要求ジョブ

実現

データの単純な要求を DDEML 要求として実行できるようになっています。それらの要求は同期をとって実行されます。実行時間の短い単純なジョブの場合には，このタイプの呼び出しが使われます。機能の名前は要求のトピックとして渡され，引数は要求の項目として渡されます。get_realpath 機能は，このタイプの一例です。

C/C++ での実現

C または C++ では，ジョブの結果は，要求の結果バッファへ直接に渡されます。エラーの場合は，要求が拒否されます。

Visual Basic での実現

Visual Basic では，要求が呼び出され，その結果はテキストかラベル制御に記録されます。アプリケーションには，この制御の CHANGE イベントを介して通知されます。制御の内容は REQUEST 呼び出しの終了時に更新されます。そこでその内容を調べることができます。

エラーは，エラーハンドラで処理する必要があります。したがって，エラー変数 “Err” に注意する必要があります。

"DHServer|" と対応する機能名は，Linktopic に指定され，引数は必要に応じて Linkitem に指定されます。

■ Best_Datatype

説明

指定した引数の名前と拡張子に最も一致するデータタイプを探します。引数の名前と拡張子に最も近いデータタイプが戻されます。拡張子は正確に一致していなければなりません。名前が一致していなければ，ワイルドカード文字 “*” で検索されます。戻される結果は，拡張子なしのデータタイプになります。

用途

データ管理の特定ディレクトリに適したデータタイプを判別することができます。

構文

LinkTopic: DHServer|best_data type

LinkItem : name extension

引数

表 10.6 Best_Datatype の引数

引数	記述
name	**ファイルまたはディレクトリの名前**
extension	**ファイルまたはディレクトリの拡張子**

適切なデータタイプの判別

ワーク “welle1.wpd” に最適なデータタイプを判別します。

例 10-2 適切なデータタイプの判別

```
Sub Form_Load ()
Label1.LinkTopic = "DHServer|best_datatype"
Label1.LinkItem ="welle1 wpd"
Label1.LinkMode = 2
Label1.LinkRequest
End Sub
```

結果 : * すべてのデータタイプが許可される

適切なデータタイプの判別

ワークディレクトリ “wks.dir” に適したデータタイプを探します。

例 10-3 適切なデータタイプの判別

```
Sub Form_Load ()
Label1.LinkTopic = "DHServer|best_datatype"
Label1.LinkItem ="wks dir"
Label1.LinkMode = 2
Label1.LinkRequest
End Sub
```

結果 : WKS　データタイプ WKS だけが許可される

注 : 一致するファイルタイプがなければジョブは拒否されます。たとえば，標準の DOS ファイルまたはディレクトリの場合などです。

■ Convert_Possible_Datatypes

説明

この機能は，特定のデータファイルをコピーするときに使えるすべてのデータタイプをリストします。たとえば，パートプログラムのデータファイル Part1o.mpf は，サブプログラムのデータファイル Part1u として，サブプログラムのディレクトリにコピーできます。

用途

特定のファイル名があるかどうかについて，データ管理で可能性のある宛先ディレクトリを調べます。

構文

LinkTopic: DHServer|convert_possible_datatypes

LinkItem : name extension

引数

表 10.7 Convert_Possible_Datatypes の引数

引数	記述
name	**ファイル名，または置換文字 “*”**
extension	**ファイル拡張子**

データタイプの判別

パートプログラムのコピー先の可能性のあるデータタイプを判別します。

例 10-4 可能性のあるデータタイプの判別

```
Sub Form_Load ()
Label1.LinkTopic = "DHServer|convert_possible_datatypes"
Label1.LinkItem ="* mpf"
Label1.LinkMode = 2
Label1.LinkRequest
End Sub
```

結果 :

#part program(MPF) * MPF#initialization program(INI) *

_ INI#subprogram(SPF) * SPF#

注 : 適切なデータタイプが見つからない場合（標準の DOS ファイルや DOS ディレクトリなどの場合），ジョブは拒否されます。

■ Get_Attributes

説明

ファイルまたはディレクトリの属性を戻します。属性のリストは，リストジョブにおけるファイル属性に対応しています。

用途

ファイルまたはディレクトリの属性を読み込みます。

構文

LinkTopic: DHServer|get_attributes

LinkItem : name extension

引数

表 10.8 Get_Attributes の引数

引数	記述
name	ディレクトリ名
extension	ファイル拡張子

データ管理の属性の読み込み

パートプログラムへコピーできる可能性のあるデータタイプを判別します。

例 10-5 データ管理の属性の読み込み

```
Sub Form_Load ()
Label1.LinkTopic = "DHServer|get_attributes
Label1.LinkItem ="¥mpf.dir"
Label1.LinkMode = 2
Label1.LinkRequest
End Sub
```

結果 :

MPF part programs DIR MPF 0 848506846 DM 77770

注 : この機能は，MMC のファイルおよびディレクトリだけに適用されます。

■ Get_Properties

説明

この機能は，データファイルまたはディレクトリのデータタイプがデータスキームで定義されていれば，その特性を戻します。結果はビットリストで，LONG の数としてコード化されます。

別の方法として，名前と拡張子で構成されるそれぞれのデータタイプを定義することも可能です。

用途

たとえば，パートプログラムが編集可能かどうかを判別します。

構文

LinkTopic: DHServer|get_properties

LinkItem : name mode

引数

表 10.9 Get_Properties の引数

引数	記述
name	**データ管理パスの名前またはファイル名**
mode	**MMC の場合は -m，NCK の場合は -n**

ワークの編集可能性

ワーク “bolzen.wpd” が編集可能かどうかを調べます。

例 10-6 ワークの編集可能性

```
Sub Form_Load ()
Label1.LinkTopic = "DHServer|get_properties"
Label1.LinkItem ="¥wks.dir¥bolzen.wpd"
Label1.LinkMode = 2
Label1.LinkRequest
End Sub
```

結果 : 1

注 : 現在のところ，ビット 0 だけが，編集可能として定義されています。

■ Get_Propertynames

説明

この機能は，get_properties で照会できる特性リストの特定ビットに，言語に応じたショートカットを提供します。

用途

各ビットの言語に応じた ID を入手できます。

構文

LinkTopic: DHServer|get_propertynames

引数

なし

言語に応じた ID

言語に応じた ID を判別します。

例 10-7 言語に応じた ID

```
Sub Form_Load ()
Label1.LinkTopic = "DHServer|get_propertynames"
Label1.LinkMode = 2
Label1.LinkRequest
End Sub
```

結果 : たとえば，ドイツ語の場合

#editable#-#

注 : このリストは，データ管理サーバの起動時に，選択した言語に応じて初期設定されます。

■ Get_Realpath

説明

ファイルとそのタイプの DOS または NCK パスを戻します。このパスは，DOS または NCK ファイルシステムのファイルシステムインタフェースの機能を呼び出すときに使えます。

用途

たとえば，パートプログラムの物理的な位置を判別します。

構文

LinkTopic: DHServer|get_realpath

LinkItem : dhpath mode

引数

表 10.10 Get_Realpath の引数

引数	記述
dhpath	**データ管理パスの名前またはファイル名**
mode	**MMC の場合は -m，NCK の場合は -n**

NCK での絶対パス

NCK におけるワーク “bolzen.wpd” の絶対パスを判別します。

例 10-8 NCK での絶対パス

```
Sub Form_Load ()
Label1.LinkTopic = "DHServer|get_realpath"
Label1.LinkItem ="¥wks.dir¥bolzen.wpd -n"
Label1.LinkMode = 2
Label1.LinkRequest
End Sub
```

結果 : NC でのファイルのパス

/NC/_N_WKS_DIR/_N_BOLZEN_WPD *

MMC での絶対パス

MMC でのパートプログラムの絶対パスを判別します。

例 10-9 MMC での絶対パス

```
Sub Form_Load ()
Label1.LinkTopic = "DHServer|get_realpath"
Label1.LinkItem ="¥mpf.dir -m"
Label1.LinkMode = 2
Label1.LinkRequest
End Sub
```

結果 : MMC でのファイルのパス

c:¥mmc2¥dh¥MPF.DIR MPF

注 : NCK にアクセスしている時は，ファイルまたはディレクトリが存在するかどうかは検査されません。MMC にアクセスしている時は，ファイル／ディレクトリが存在するかどうかが検査されます。存在しない場合，要求は拒否されてエラーになります。

■ Possible_Datatypes

説明

データ管理のディレクトリに格納できるファイルタイプをリストします。

用途

特定のディレクトリのデータタイプを判別します。

構文

LinkTopic: DHServer|possible_datatypes

LinkItem : dhpath [mode}

引数

表 10.11 Possible_Datatypes の引数

引数	記述
dhpath	**データ管理パスの名前またはファイル名**

引数	記述
mode	**MMC の場合は -m，NCK の場合は -n**

可能なデータタイプ

ワーク “bolzen.wpd” に可能なファイル名をリストします。

例 10-10 可能なデータタイプ

```
Sub Form_Load ()
Label1.LinkTopic = "DHServer|possible_datatypes"
Label1.LinkItem ="¥wks.dir¥bolzen.wpd -n"
Label1.LinkMode = 2
Label1.LinkRequest
End Sub
```

結果 : 可能なデータタイプが “#” で区切ってリストされます。

#Part-program(MPF) * MPF
#Autoturn-program * 41
#Time-calculation(DAT) * DAT
#DP-initialization DPWP * INI
#Initialization-program(INI) INI
#Compensation-data(IKA) * IKA
#Tool-offsets(TOA) * TOA
#Machine-data(TEA) * TEA
#Setting data(SEA) * SEA
#Channel-user-data(GUD) * GUD
#Comment(COM) * COM
#Subroutine(SPF) * SPF
#ZeroOffset/Frame(UFR) * UFR#

10.4.3 単純な実行ジョブ

実現

データを戻さない単純なジョブを DDEML Execute で実行できるようになっています。呼び出しの結果は，DDEML 機能の戻りコードに示されます。ここで戻される値は，DDE_FACC，DDE_FBUSSY，または DDE_FNOTPROCESSED のいずれかです。

C/C++ での実現

C または C++ では，命令と引数は，DDEML 呼び出しの転送バッファのストリングとして渡されます。

Visual Basic での実現

Visual Basic では，EXECUTE 呼び出しは Text または LabelControl にリンクされます。ここでは，エラーは適切なエラーハンドラによって処理されなければなりません。したがって，変数 “Err” に注意する必要があります。

Linktopic として “DHServer|Result” を必ず指定しますが，データ管理サーバは Linkitem を無視します。VB 構文に関する条件を満たすために，ともかく Linkitem を指定する

必要がありますが，値は何でも構いません。

■ Cancel

説明

すべてのジョブ（複雑な実行ジョブ）をキャンセルし，結果変数を示します。

用途

複雑なジョブをキャンセルします。

構文

LinkExecute: **cancel DHServer** 変数

引数

表 10.12 Cancel の引数

引数	記述
DHServer 変数	**複雑な実行ジョブを開始したときに，結果変数として指定された変数**

ジョブのキャンセル

結果変数 “StatVar” でジョブをキャンセルします。

例 10-11 ジョブのキャンセル

```
Sub Form_Load ()
Label1.LinkTopic = "DHServer|Result"
Label1.LinkMode = 2
On Error Resume Next
Label1.LinkExecute "cancel StatVar"
if Err = 0 Then
 no error
Else
' error
End If
End Sub
```

結果 : Err=0: すべてのジョブがキャンセルされた

Err <>0: コマンドが正常に実行されなかった

■ Exist

説明

ファイルが NCK に存在するのか MMC に存在するのかを調べます。

用途

たとえば，ワークピースが存在するかどうかを判別します。

構文

LinkExecute: **exist dhpath mode**

引数

表 10.13 Exist の引数

引数	記述
dhpath	**データ管理パスでの名前**
mode	**MMC の場合は -m，NCK の場合は -n**

ファイルの存在

ワーク “welle1.wpd” が MMC に存在するかどうかを調べます。

例 10-12 ファイルの存在

```
Sub Form_Load ()
Text1.LinkTopic = "DHServer|Result"
Text1.LinkMode = 2
On Error Resume NextText
Text1.LinkExecute "exist ¥wks.dir¥welle1.wpd -m"
If Err = 0 Then
' work piece exists
Else
' work piece does not exist
End If
End Sub
```

結果 : Err=0: ファイルが存在する

Err<>0: ファイルが存在しない

注 : これらのエラーメッセージからは，ジョブが実行されていないのか，あるいはファイルが見つかっていないのかを区別することはできません。

■ Extern

説明

外部 NC プログラムを扱う機能を提供します。MMC では，EXTERN コマンドで最初の 1 ファイルだけが選択されます。このファイルは，データ管理に入れておくことができません。

続いて，NCDDE サーバの機能を使って，このファイルを MMC から NCK へコピーする必要があります。この作業は，ドメインサービスの COPY_TO_NC（8.6 を参照）で行います。

用途

実行するパートプログラムを外部から選びます。

構文

LinkExecute: **extern dhpath channel**

引数

表 10.14 Extern の引数

引数	記述
dhpath	名前
channel	チャネル番号（常に **2** 桁）を指定

外部からの実行

実行するパートプログラム “test.mpf” を外部の最初のチャネルから選択します。

例 10-13 外部からの実行

```
Sub Form_Load ()
Label1.LinkTopic = "DHServer|Result"
Label1.LinkMode = 2
On Error Resume Next
Label1.LinkExecute "extern ¥mpf.dir¥test.mpf 01"
If Err = 0 Then
' selection worked
Else
' selection did not work
End If
Label1.LinkTopic = "ncdde|ncu840D"
Label1.LinkMode = 2
Label1.LinkExecute "COPY_TO_NC(C:¥tmp¥mpf1.mpf,
_ /NC/_N_MPF_DIR/_N_TEST_MPF,trans)"
End Sub
```

結果 : Err=0: 選択が有効

Err<>0: 選択が無効

注 : 選択は，NCK において，指定したデータ管理パスにあるファイルへ常に適用されます。初めて EXTERN コマンドを実行するときには，ファイルは存在しません。ファイルが存在すると，エラーになります。

■ Rename

説明

MMC と NCK のファイルをリネームする機能を提供します。データタイプに一貫性があれば，これは検査されません（Convert_Possible_Datatypes 機能を参照）。

用途

たとえば，ワークとパートプログラムをリネームします。

構文

LinkExecute: **rename dhpath -mode NeuName [datatype] extension**

引数

表 10.15 Rename の引数

引数	説明
dhpath	データ管理のパス
mode	-n=NCK，-m=MMC。ディレクトリは，NCK および MMC でリネームされる
NeuName	データファイルの新しい名前
datatype	新しいデータタイプ，* または名前スキーム（例，WKS）
extension	新しい拡張子。データタイプと一緒になって，データファイルの新しいデータタイプを定義する

パートプログラムのリネーム

MMC で，パートプログラム “mpf1.mpf” を “mpf2.mpf” へリネームします。

例 10-14 パートプログラムのリネーム

```
Sub Form_Load ()
Label1.LinkTopic = "DHServer|Result"
Label1.LinkMode = 2
On Error Resume Next
Label1.LinkExecute "rename ¥mpf.dir¥mpf1.mpf -m mpf2 *.mpf"
If Err = 0 Then
' renamed
Else
 not renamed
End If
End Sub
```

結果 : Err=0: リネームされた

Err<>0: リネームされないか存在しない

注 : エラーメッセージでは，ファイルが存在しないのか，それともリネームできなかったのかを区別できません。

■ Select

説明

パートプログラムとワークを選びます。それらをすでに NC にロードしておく必要があります。

用途

アプリケーションでパートプログラムとワークを選ぶことができます。

構文

LinkExecute: **select dhpath channel**

引数

表 10.16 Select の引数

引数	記述
dhpath	**データ管理パスでの名前**
channel	**2 桁のチャネル番号**

パートプログラムの選択

最初のチャネルでパートプログラム “MPF1.mpf” を選びます。

例 10-15 パートプログラムの選択

```
Sub Form_Load ()
Label1.LinkTopic = "DHServer|Result"
Label1.LinkMode = 2
On Error Resume Next
Label1.LinkExecute "select ¥MPF.DIR¥MPF1.mpf 01"
If Err = 0 Then
' selected
Else
' not selected
End If
End Sub
```

結果 : Err=0: パートプログラムが選択された

Err<>0: パートプログラムが選択されていない

注 : NC 変数 “/Channel/ProgramInfo/ progName[uKanal]” により，パートプログラムが選択されているかどうかが検査されます。

■ Setaccess

説明

ファイルまたはディレクトリのアクセス許可を設定します。現在アクティブなユーザクラスは，ファイルまたはディレクトリのアクセス記述に設定された値以下でなければなりません。アクセス許可を設定するもう一つの方法は，‘_’ を入力することです。‘_’ で指定した許可は変更されることがないので，変更の妥当性は検査されません。通常，妥当性が検査されるときには，EXECUTE 許可は考慮されません。

用途

ファイルおよびディレクトリのアクセス権限を設定します。

構文

LinkExecute: **setaccess dhpath access [mode]**

引数

表 10.17 Setaccess の引数

引数	記述
dhpath	データ管理での名前
access	要求されたアクセス許可
mode	MMC の場合は -m，NCK の場合は -n モードを設定しないと，アクセス権限は MMC ファイルにも NCK ファイルにも設定される。 ディレクトリには，常に両方の許可が設定される。最終的に，指定したモードは無視される。

パートプログラムのアクセス許可の設定

NCK のパートプログラム “MPF1.mpf” に対するアクセス許可を，47774 から 54774 に設定します。

ファイルの以前のアクセス許可 : 47774

現在のユーザクラス : 4　アクセス許可を変更

現在のユーザクラス : 5　アクセス許可は変更されない

例 10-16 パートプログラムのアクセス許可の設定

```
Sub Form_Load ()
Label1.LinkTopic = "DHServer|Result"
Label1.LinkMode = 2
On Error Resume Next
Label1.LinkExecute "setaccess ¥WKS.DIR¥WELLE1.WPD¥MPF1.mpf
_ 54774 –n "
If Err = 0 Then
' access permission has been changed
Else
' access permission has not been changed
End If
End Sub
```

結果 : Err=0: アクセス許可が変更された

Err<>0: アクセス許可が変更されていない

注 : ディレクトリの場合，ディレクトリについての表示は統一されるため，許可は MMC にも NCK にも設定されます。

■ Startsave

説明

サーバに VarName というメモリ領域を確立します。名前を結果変数として定義している場合，stopsave でメモリ領域を解放しない限り，結果変数の実際の内容はメモリ領域に格納されたままになります。変数の内容については，要求の指示（Topic: 結果，Item: 結果変数）で読み取ることができます。この変数は，一連の複数の指示で使うことができます。

用途

この呼び出しは，ホットリンク接続ではなく要求ジョブを繰り返すことにより，ジョブの結果を調べるアプリケーションで使うことができます。

構文

LinkExecute: **startsave VarName**

引数

表 10.18 Startsave の引数

引数	記述
VarName	**変数の名前**

結果変数の作成

例 10-17 結果変数 “test” の作成

```
Sub Form_Load ()
Label1.LinkTopic = "DHServer|Result"
Label1.LinkMode = 2
On Error Resume Next
Label1.LinkExecute "startsave test"
If Err = 0 Then
' no error
Else
' error
End If
End Sub
```

結果 : Err=0: 変数が作成された

Err<>0: 変数は作成されていない

■ Stopsave

説明

startsave の補助機能です。メモリ領域 VarName が解放されます。

用途

データ管理サーバから結果変数を除去します。

構文

LinkExecute: **stopsave VarName**

引数

表 10.19 Stopsave の引数

引数	記述
VarName	**変数の名前**

10.4.4 複雑な実行ジョブ

使用分野

環境によって非常に時間のかかることのある機能は，非同期に実行されます。すなわち，アプリケーションで他のアクションを実行しながら，DHServer は要求されたコマンドを実行するという具合です。

Visual Basic での実現

複雑なジョブを発行するためには，まずアプリケーション側で，自由に選択した結果変数へのホットリンクを設定します。このため，Topic "DHServer|Result" は予約済みとなっています。Item には，結果変数が設定されます。現在の Linkmode が 1 か 2 に設定されていれば，データ管理サーバは内部的に変数を作成し，要求があれば変更をアプリケーションへ通知します。

その後，実際のコマンドを送ることができます。ここでは，Linktopic として "DHServer|Result" も指定しておきます。Linkitem は評価されませんが，VB の規則に応じて設定するようにします。Linkmode には 2（自動）を設定します。これで，Linkexecute を使ってコマンドを呼び出せるようになりました。

呼び出し

転送バッファ内にある複雑な実行ジョブの場合，コマンド行で次の構文を使います。

functionname resultvariable argument1 ....._ argumentN

機能の名前，結果変数，および引数は，ストリングで転送バッファのジョブへ渡されます。データ管理サーバは，そのジョブをジョブリストに載せます。リストに掲載すると，肯定通知による応答がなされます。拒否されるときは否定通知によって応答されます（「単純な実行ジョブ」を参照）。アプリケーションは，ジョブが完了したら，確立したホットリンクをキャンセルすることもでき，別のジョブのために保持しておいても構いません。データ管理サーバは，アプリケーションに関係なく，ジョブリストに沿って機能します。

結果

アプリケーションは，結果変数を使って，ジョブの状態を調べることができます。

C または C++ では，指示は CALLBACK 機能である XTYPE_ADVDATA を介して実行されます。

Visual Basic では，アプリケーションは結果変数へのホットリンクを確立できます。

結果のバッファは，CF_TEXT フォーマットで記入されます。結果のバッファの要素は，# で区切られます。

最初の要素にはコマンドの状態が示され，残りの要素はジョブ固有のものです。

コマンドの状態

状態は，次の値のいずれかになります。

0 ～ 99：ジョブの実行は n パーセント経過（n の範囲は 0 ～ 99）
100：ジョブは正常に実行された

>100：実行時にエラーが生じた

ジョブの実行時には，ホットリンク経由でジョブの進行状況が継続的に報告されます。ジョブの進行状況は，パーセント単位でアプリケーションへ送られます。状態が 100 であるか 100 を超えてエラーになると，ジョブは完了し，結果が渡されます。

注 : 開発の助けとするために，エラーコードの後にテキストのエラーメッセージが挿入されています。このテキストは，データ管理サーバの正式なインタフェースの一部ではありません。
例 :
#106 error when creating or opening the file

次のステップでは，ホットリンクの確立を解除することができます。データ管理によってジョブを実行している間に，アプリケーションは，その後のオペレータのアクションを受け入れる用意ができます。サーバーサーバは，アプリケーションからの追加ジョブを受け入れることができます。アプリケーション自体で，これらのジョブの調整を行う責任があります。

結果バッファの詳しい内容については，該当する機能が説明されている箇所に述べられています。結果は CF_TEXT フォーマットで戻されるので，Visual Basic へ直接に接続できるようになります。

■ Activate

説明

ファイル／ディレクトリを同じディレクトリの NCK ファイルシステムへコピーします。正常に転送したら，ソースファイルは削除されます。ディレクトリをコピーした場合は，ソースは削除されません。

用途

パートプログラム／ワークを NCK にロードします。

構文

LinkExecute: **activate ergvar source [-f]**

引数

表 10.20 Activate の引数

引数	記述
Ergvar	結果変数 : ジョブの現在の状態を含む
source	ソースファイルのデータ管理パス
-f	-f（force= 強制上書き）（宛先ファイルがある場合）

アクセス許可

次のアクセス許可が必要です。
読み込み： ソース
書き込み： 宛先ディレクトリ，宛先
削除： ソース

パートプログラムの活動化

パートプログラム “TP1.MPF” を MMC から NCK へ移動します。

例 10-18 パートプログラムの活動化

```
Sub Form_Load ()
Label1.LinkTopic = "DHServer|Result"
Label1.LinkItem = "transStat"
Label1.LinkMode = 1
Label2.LinkTopic = "DHServer|Result"
Label2.LinkMode = 2
Label2.LinkExecute "activate transStat ¥MPF.DIR¥TP1.MPF"
End Sub
```

結果変数 “transStat” には，次のフォーマットの結果バッファが含まれます。
#state# absolute path of the file on the MMC
#100#c:¥mmc2¥dh¥MPF.DIR#

注 : 宛先ファイルに固定名がある場合，ファイルタイプに属する固定名は，宛先ファイルの名前とは別に，自動的にファイル名またはディレクトリ名として設定されます。

■ Activate2

説明

ファイル／ディレクトリを同じディレクトリの NCK ファイルシステムへコピーします。正常に転送した後も，ソースファイルは削除されません。

用途

パートプログラムを，MMC から削除せずに NCK へロードします。

構文

LinkExecute: **activate2 ergvar source [-f]**

引数

表 10.21 Activate2 の引数

引数	記述
Ergvar	**結果変数 : ジョブの現在の状態を含む**
source	**ソースファイルのデータ管理ファイル**
-f	**(force= 強制上書き) (同じ名前の宛先ファイルがある場合)**

アクセス許可

次のアクセス許可が必要です。
読み込み：ソース

書き込み：宛先ディレクトリ，宛先

■ Copy

説明

ファイルソースをファイル宛先へコピーします。タイプは，新しい宛先のタイプが指定されている場合にだけ変換されます。指定されていなければ，ソースと宛先は同じデータタイプでなければなりません。

copy 機能では，DOS ファイルおよびディレクトリもハードディスクへコピーします。

mode 引数により，ソースおよび宛先の保管場所が決まります。m = MMC ハードディスクか，n = NCK を設定することができます。

宛先については，引数 f（存在する場合，force= 強制上書き）を使って，任意で指定することができます。

用途

データ管理の要素をコピーします。

構文

LinkExecute: **copy ergvar source -mode target -mode[f] [datatype]**

引数

表 10.22 Copy の引数

パラメータ	説明
ergvar	結果変数 : ジョブの現在の状態を含む
source	ソースファイルのデータ管理パス
mode	MMC の場合は -m，NCK の場合は -n
target	宛先ファイルのデータ管理パス
mode	MMC の場合は -m，NCK の場合は -n
-f	強制上書き（同じ名前の宛先ファイルがある場合）

アクセス許可

次のアクセス許可が必要です。

読み込み：ソース

書き込み：宛先ディレクトリ，宛先

ワークピースのコピー

ワーク “WELLE.WPD” を NCK から MMC の “WELLE3.WPD” へコピーします。

例 10-19 ワークのコピー

```
Sub Form_Load ()
Label1.LinkTopic = "DHServer|Result"
Label1.LinkItem = "transStat"
Label1.LinkMode = 1
```

```
Label2.LinkTopic = "DHServer|Result"
Label2.LinkMode = 2
abel2.LinkExecute "copy transStat ¥WKS.DIR¥WELLE.WPD –n
_ ¥WKS.DIR¥WELLE3.WPD -m"
End Sub
```

結果変数 “transStat” には，次のフォーマットの結果バッファが含まれます。
#state# MMC 内のファイルの絶対パス #
#100#c:¥mmc2¥dh¥WKS.DIR¥WELLE3.WPD#

注 : 宛先ファイルに固定名がある場合，ファイルタイプに属する固定名は，宛先ファイルの名前とは別に，自動的にファイル名またはディレクトリ名として設定されます。

■ Create

説明

name という名前，data type というデータタイプ，そして extension が使われている dirpath ディレクトリに，ファイルまたはディレクトリを作成します。

用途

NCK または MMC でパートプログラム／ワークを作成します。データ管理へテキストを追加することができます。

構文

LinkExecute: **create ergvar dirpath name ext data type -mode[file]**

パラメータ

表 10.23 Create のパラメータ

パラメータ	説明
ergvar	結果変数 : ジョブの現在の状態を含む
dirpath	ファイルまたはディレクトリが作成されるディレクトリのデータ管理パス
name	ファイルまたはディレクトリの名前
ext	ファイルまたはディレクトリの拡張子
datatype	*，またはパターンに応じて固定 NCK 名。DOS ファイルの場合は “DOSFILE”，DOS ディレクトリの場合は “DOSDIR”
mode	MMC の場合は -m，NCK の場合は -n，上書きの場合は -f
file	データ管理の新しいオブジェクトの内容を含む DOS ファイルを任意で指定する。書き込み保護を迂回してしまわないように，データ管理ツリーには含められない。file は，たとえば "tmp" ディレクトリのファイルにすることができる。

アクセス許可

次のアクセス許可が必要です。

書き込み : 宛先ディレクトリ，宛先

パートプログラムの作成

NCK にパートプログラム “WELLE.MPF” を作成します。この名前のプログラムがすでに存在する場合，上書きされます。

例 10-20 パートプログラムの作成

```
Sub Form_Load ()
Label1.LinkTopic = "DHServer|Result"
Label1.LinkItem = "transStat"
Label1.LinkMode = 1
Label2.LinkTopic = "DHServer|Result"
Label2.LinkMode = 2
Label2.LinkExecute "create transStat ¥MPF.DIR¥ WELLE MPF * -nf"
End Sub
```

結果変数 “transStat” には，次のフォーマットの結果バッファが含まれます。
#state# ファイルの DOS または NCK パス #
#100#//NC_N_MPF_DIR/_N_WELLE_MPF#

注 : NCK でディレクトリのファイルタイプに固定名があれば，そのファイルタイプに属する固定名は，宛先ファイルに渡される名前とは別に，ディレクトリ名として自動的に設定されます。同じことは，MMC で固定名を持つディレクトリにも当てはまります。

■ Del

説明

アクセス許可によって許可されれば，ファイルまたはディレクトリを削除します。ディレクトリを削除できるのは，そこに何もファイルが含まれていない場合だけです。

用途

NCK または MMC からパートプログラム／ワークを削除します。

構文

LinkExecute: **del ergvar dhpath [mode]**

パラメータ

表 10.24 Del のパラメータ

パラメータ	説明
Ergvar	結果変数 : ジョブの現在の状態を含む
dhpath	ソースファイルのデータ管理ファイル
mode	MMC の場合は -m，NCK の場合は -n

アクセス許可

次のアクセス許可が必要です。

書き込み：ディレクトリ

削除：　ファイル

ワークの削除

NCK からワーク “WELLE2.WPD” を削除します。このワークに何もプログラムが含まれていない場合にだけ，削除することができます。

例 10-21 ワークの削除

```
Sub Form_Load ()
Label1.LinkTopic = "DHServer|Result"
Label1.LinkItem = "transStat"
Label1.LinkMode = 1
Label2.LinkTopic = "DHServer|Result"
Label2.LinkMode = 2
Label2.LinkExecute "del transStat ¥WKS.DIR¥WELLE2.WPD -n"
End Sub
```

結果変数 “transStat” には，次のフォーマットの結果バッファが含まれます。

#state#clear text#

以下に例を示します。
#100#　　削除に成功
#102#file does not exist# 削除に失敗

注 : アクセス許可で許可されれば，ファイル／ディレクトリは削除されます。空ではないディレクトリは，アクセス許可に関係なく，削除されません。

■ List

説明

データ管理のディレクトリの内容を読み込みます。ファイル／ディレクトリは，アルファベット順で提供されます。

用途

ディレクトリの内容をリストします。

構文

LinkExecute: **list ergvar dirpath [mode]**

パラメータ

表 10.25 List のパラメータ

パラメータ	説明
ergvar	結果変数 : ジョブの現在の状態を含む
dirpath	データ管理のパス
mode	MMC の場合は -m，NCK の場合は -n

パラメータ	説明
-f	上書き（同じ名前の宛先ファイルがある場合）

アクセス許可

次のアクセス許可が必要です。

読み込み：ソースディレクトリ

表示： 個々のファイル

ディレクトリの内容のリスト

サブディレクトリ “¥SPF.DIR” の内容をリストします。

例 10-22 ディレクトリの内容のリスト

```
Sub Form_Load ()
Label1.LinkTopic = "DHServer|Result"
Label1.LinkItem = "transStat"
Label1.LinkMode = 1
Label2.LinkTopic = "DHServer|Result"
Label2.LinkMode = 2
Label2.LinkExecute "list transStat ¥SPF.DIR"
End Sub
```

結果変数 “transStat” には，次のフォーマットの結果バッファが含まれます。

#state#lname data type text extension realname length date attribute#

List の状態情報

結果は，CF_TEXT フォーマットで作成され，1 行に 1 つのファイルまたはディレクトリの特徴を記述した 1 項目になります。それぞれの行には，以下の要素が含まれています。

表 10.26 List の状態情報

情報	意味
lname	データ管理でのファイル名（_N_ なしの NC ファイル）
datatype	データスキームに従ったファイルタイプの名前。このスキームに含まれていないファイルは，プレースホルダ “----” で表される
text	ファイルタイプの説明（10.2 章の表を参照）
extension	ファイル拡張子
realname	DOS ファイルシステムまたはNC ファイルシステムにあるファイルの“本当の”名前。この名前は，たとえばファイルを開くときに必要になる。
length	ファイルの長さ。ディレクトリの場合は，0 が入る
date	最終書き込みアクセスの日付。時刻については，01.01.1970（1970 年 1 月 1 日）以降の秒数で示される

attribute	ファイルまたはディレクトリの特性のリスト： D：ディレクトリ F：ファイル N：NCK にあるか，MMC と NCK とにある M：MMC にある
access	アクセス許可：読み込み，書き込み，実行，表示，削除

ディレクトリの項目ごとに，lname から attribute までの値が提供されます。項目は **#** によって区分されます。

#100#U1 * sub program(SPF) SPF U1 31 857568254 FM 64774 #

注： ディレクトリの内容が戻されます。NC に存在し，MMC ハードディスクにも存在するディレクトリの場合，その内容は MMC と NC の両方のディレクトリで構成されます。

注： システム時刻への変換の場合，Visual Basic の CVDate 機能を使います。この機能については，VB のオンラインヘルプで説明されています。

■ Passivate

説明

ファイル／ディレクトリを同じディレクトリの NCK ファイルシステムへコピーします。転送に成功したら，ソースファイルは削除されます。ディレクトリがコピーされる場合，ソースは削除されません。
引数 mode については，-f（force= 強制上書き）と一緒に指定できます。このパラメータを指定しない場合，既存の宛先は上書きされない可能性があります。

用途

NCK からパートプログラム／ワークを削除します。すなわち，パートプログラム／ワークは，NCK から MMC へ移動します。

構文

LinkExecute: **passivate ergvar source [mode] [-f]**

パラメータ

表 10.27 Passivate のパラメータ

パラメータ	説明
ergvar	結果変数：ジョブの現在の状態を含む
source	ソースファイルのデータ管理パス
mode	-f 上書き（同じ名前のファイルがある場合）

アクセス許可

次のアクセス許可が必要です。

読み込み：ソース

書き込み：宛先ディレクトリ，宛先

削除：　ソース

パートプログラムの削除

パートプログラム “TP1.MPF” を NCK からロードします。

例 10-23 パートプログラムの削除

```
Sub Form_Load ()
Label1.LinkTopic = "DHServer|Result"
Label1.LinkItem = "transStat"
Label1.LinkMode = 1
Label2.LinkTopic = "DHServer|Result"
Label2.LinkMode = 2
Label2.LinkExecute "passivate transStat ¥MPF.DIR¥TP1.MPF"
End Sub
```

結果変数 “transStat” には，次のフォーマットの結果バッファが含まれます。

#state# ファイルの DOS パス #

#100#c:¥mmc2¥dh¥MPF.DIR¥TP1.MPF#

注： NCK でディレクトリのファイルタイプに固定名があれば，そのファイルタイプに属する固定名は，宛先ファイルに渡される名前とは別に，ディレクトリ名として自動的に設定されます。

11　参照データ

この章では，オンラインヘルプおよびPIサービスを使用する上で必要なすべての情報を記載しています。

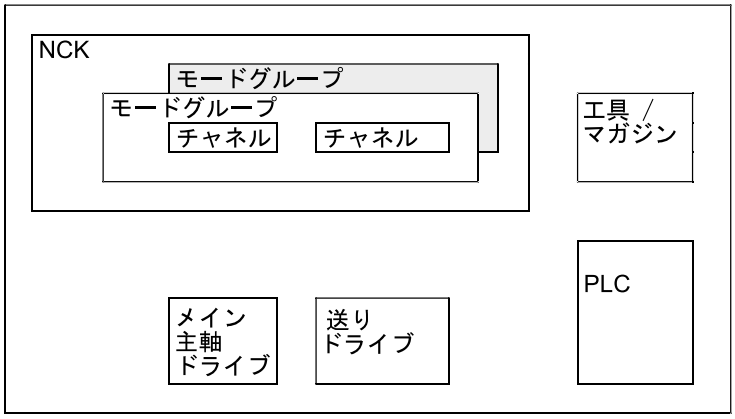
詳細については，listbook /LIS/ またはヘルプファンクション「NCDDEデータヘルプ」で説明されています。データアクセスについては，8章「NC-DDEサーバ」で説明しています。

11.1 概説

11.1.1 NC のエリア

以下の図は NC のエリアを示しており，続く表ではこれらのアドレスをリストします。

表 11.1 エリアのリスト



エリア	リンク項目の ID	エリアアドレス	PI サービスの ID	エリアの ID
NCK	Nck	0	001	N
モードグループ	Bag	1	1xx	B
チャネル	Channel	2	2xx	C
軸	Axis	3	3xx	A
工具補正，ツールマガジンデータ	Tools	4	4xx	T
PLC	Plc			?
送りドライブ	DriveVSA	5	5xx	V
メイン主軸ドライブ	DriveHSA	6	6xx	H
通信モジュール			K00	
非アクティブファイルシステム			P01	

11.1.2 NC のファイルシステム

NCK コンポーネントのデータは，外部からの視点で見ると，アクティブファイルシステムと非アクティブファイルシステムという 2 つの部分に分けられます。

アクティブファイルシステム

アクティブファイルシステムとは，NCK のメインメモリのことです。これはいくつかのエリアとそれらのエリアの単位で構成されています。アクティブファイルは，常に単位とエリアの組み合わせで直接割り当てられます。これらを削除することはできません。また，これらの処理（ダウンロードおよびアップロード）は非アクティブファイルの処理とは異なります。

表 11.2 では，アクティブファイルシステムのファイルをリストします。

表 11.2 アクティブファイルシステムのファイル（抜粋）

データタイプ	ファイル名
すべての NC データ	_N_INITIAL_INI
すべてのマシンデータ	_N_COMPLETE_TEA
すべてのグローバル NC マシンデータ	_N_NC_TEA
すべてのチャネル固有マシンデータ	_N_CH_TEA
チャネル n のマシンデータ	_N_CHn_TEA
すべての軸固有のマシンデータ	_N_AX_TEA
軸 n のマシンデータ	_N_AXn_TEA
すべての設定データ	_N_COMPLETE_SEA
すべてのグローバル NC 設定データ	_N_NC_SEA
すべてのチャネル固有設定データ	_N_CH_SEA
チャネル n の設定データ	_N_CHn_SEA
すべての軸固有の設定データ	_N_AX_SEA
軸 n の設定データ	_N_AXn_SEA
すべてのオプションデータ	_N_COMPLETE_OPT
すべてのグローバル NC オプションデータ	_N_NC_OPT
すべてのチャネル固有オプションデータ	_N_CH_OPT
すべての軸固有オプションデータ	_N_AX_OPT
すべてのグローバルユーザデータ	_N_COMPLETE_GUD
すべてのグローバル NC ユーザデータ	_N_NC_GUD
すべてのチャネル固有ユーザデータ	_N_CH_GUD
チャネル n のユーザデータ	_N_CHn_GUD

すべての工具／マガジンデータ	_N_COMPLETE_TOA
すべての工具データ	_N_TO_TOA
エリア t の工具データ	_N_TOt_TOA
すべてのマガジンデータ	_N_TO_TMA
エリア t のマガジンデータ	_N_TOt_TMA
すべての保護エリア	_N_COMPLETE_PRO
データタイプ	ファイル名
すべてのグローバル NC 保護エリア	_N_NC_PRO
すべてのチャネル固有の保護エリア	_N_CH_PRO
チャネル n のチャネル保護エリア	_N_CHn_PRO
すべての R パラメータ	_N_COMPLETE_RPA
すべてのチャネル固有 R パラメータ	_N_CH_RPA
チャネル n のチャネル R パラメータ	_N_CHn_RPA
すべての原点オフセット	_N_COMPLETE_UFR
すべてのグローバル NC 原点オフセット	_N_NC_UFR
すべてのチャネル固有の原点オフセット	_N_CH_UFR
チャネル n のチャネル原点オフセット	_N_CHn_UFR
すべての改変された補正データ	_N_NC_CEC
すべての象限エラー補正データ	_N_AX_QEC
軸 n の象限エラー補正データ	_N_AXn_QEC
すべての親ねじエラー補正データ	_N_AX_EEC
軸 n の親ねじエラー補正データ	_N_AXn_EEC

受動ファイルシステム

非アクティブファイルシステムは，NCK の大容量記憶を表します。これは階層構造になっています。非アクティブファイルシステムは，後でアクティブとなるファイルのために中間記憶装置を用意します。たとえば，ここにはパートプログラムが保管されます。非アクティブファイルシステムは常に事前定義されている部分と動的な変数構造で構成されています。

表 11.3 では，非アクティブファイルシステムの標準ディレクトリをリストします。

表 11.3 非アクティブファイルシステムの標準ディレクトリ（抜粋）

データタイプ	ディレクトリ名
MMC 表示マシンデータ	_N_BD_DIR
ワークピース	_N_WKS_DIR

グローバルパートプログラム	_N_MPF_DIR
グローバルサブプログラム	_N_SPF_DIR
ユーザサイクル	_N_CUS_DIR
当社提供標準サイクル	_N_CST_DIR
コメントデータファイル	_N_COM_DIR
マクロおよびグローバルユーザデータの定義ファイル	_N_DEF_DIR
バイナリフォーマットのメイン主軸ドライブマシンデータ	_N_HS_DIR
バイナリフォーマットのフィードドライブマシンデータ	_N_VS_DIR
ユーザデータ（OEM）	_N_OEM_DIR
システムデータ	_N_SYF_DIR

11.1.3 ドメインサービス

ドメインサービスは，MMC と NCK との間でデータを転送するためのものです。

アプリケーションについては，8 章「MMC ⇔ NCK/PLC 間のインタフェース」で示されています。

表 11.4 では，ドメインサービスをリストします。

表 11.4 ドメインサービスの概要

コマンド	説明
COPY_FROM_NC	NCK から MMC へのコピー。追加情報あり
COPY_FROM_NC_BINARY	NCK（PLC）から MMC へのコピー。追加情報なし
COPY_TO_NC	MMC から NCK へのコピー。追加情報あり
COPY_TO_NC_BINARY	MMC から NCK（PLC）へのコピー。追加情報なし
MAP_ACC_NC	NC カーネルから ACC ファイルをロードし, DDE インタフェースを準備

11.1.4 PI サービス

PI サービス（プログラム呼び出し）は，NCK の首尾一貫した特定の機能を呼び出すために使用されます。

アプリケーションについては，8 章「MMC ⇔ NCK/PLC 間のインタフェース」で示されています。

以下の表では，PI サービスをリストします。

表 11.5 PI サービスの概要

機能グループ	意味	PI 名
NC-	セットアップ	_N_IBN_SS
機能	再構成	_N_CONFIG
	ドメインオプション引き数	_N_DOMOPT
	MMC セマフォ	_N_MMCSEM
	コンパイル - サイクルにアドレス指定される PI サービス	_N_CMPCYC
	ワークピースカウンタ	_N_TMPCIT
	アップロードするファイルの選択	_N_F_XFER
	MDA メモリの削除	_N_F_DMDA
	NC アラームの取り消し	_N_CANCEL
	ディジタル化をオンに設定	_N_DIGION
	ディジタル化をオフに設定	_N_DIGIOF
	測定システムの設定 :（inch/mm）	_N_SCALE_
NC-	チャネルで実行するプログラムの選択	_N_SELECT
プログラム	外部から外部実行するプログラムの選択	_N_EXTMOD
機能	外部から外部実行するプログラムの選択	_N_EXTERN
	サーチの活動化	_N_FINDBL
	ユーザフレームの設定	_N_SETUFR
	記憶超過をオンに設定	_N_OST_ON
	記憶超過をオフに設定	_N_OST_OF
	割り込みの割り当て	_N_ASUP__
ファイル	ファイルのオープン	_N_F_OPEN
機能	ファイルのクローズ	_N_F_CLOS
	ファイルの名前変更	_N_F_RENA
	NCK 内のファイルのコピー	_N_F_COPY
	ファイルの削除	_N_F_DELE
	ファイルサーチポインタの位置決め	_N_F_SEEK
	実際のユーザデータの活動化	_N_SETUDT
	変数エリアの削除	_N_DELVAR
アクセスレベル	新規パスワード	_N_NEWPWD
機能	ログイン	_N_LOGIN_
	ログアウト	_N_LOGOUT
	ファイルへのファイル保護の追加	_N_F_PROT
工具	工具の作成	_N_CREATO
機能	工具管理のための工具の作成	_N_TMCRTO
	新しい切削刃の作成	_N_CREACE
	既存の工具の削除	_N_DELETO
	ロード用の空場所の検索	_N_TMFDPL
	ロード用にマガジン位置を提供。工具をアンロード	_N_TMMVTL
	マガジンエリアまたは工具の位置決め	_N_TMPOSM
	既存の工具用の切削刃の作成	_N_CRCEDN

	既存の工具用の切削刃の削除	_N_DELECE
	工具の作成 : 名前，工具番号，および duplo 番号を指定	_N_TMCRTO
	空場所の検索 : タイプ，マガジン，および工具用スペース	_N_TMFPBP
	T 番号の検索 : 工具名および duplo 番号を指定	_N_TMGETT
	固有の D 番号の検査の開始	_N_CHEKDM
	姉妹工具グループ内の工具の活動化	_N_SETTST
	マガジン内のアクティブな磨耗グループの設定	_N_TMAWCO
	定義された番号の切削刃を使用した工具の作成	_N_TMCRTC
	アクティブ状態のリセット	_N_TMRASS
PLC	モジュールの活動化	_INSE
機能	非アクティブモジュールの削除	_DELE
	PLC のスタート／ストップ	_PROGRAM
	PLC メモリのリセット（PLC_MEMORYRESET を参照）	

11.1.5 データフォーマット

この説明では以下のデータフォーマットを使用します。:

表 11.6 データフォーマットの概要

データタイプ	長さ（ビット）	長さ（バイト）	範囲
char	8	1	0 ～ 255
unsigned	16	2	0 ～ 65535
long	32	4	-2.147.483.648 ～ +2.147.483.647
double	64	8	_}10 -307 ～ ±10 308
string	n x 8	n	ASCII 文字のセット。末尾に "¥0" が付く
Byte (PLC)	8	1	
Word (PLC)	16	2	
Doubleword	32	4	
Float	32	4	先行する 4 バイトは，浮動小数点数と解釈されます。
Double	64	8	先行する 8 バイトは，浮動小数点数と解釈されます。

11.2 変数定義

11.2.1 概要

このオンラインヘルプには，DDE インタフェースを介してアクセスできる NC 変数のリストが含まれています。この使用方法については，8 章「MMC ⇔ NCK/PLC 間のインタフェース」を参照してください。

11.2.2 アクティブファイルシステムおよび非アクティブファイルシステムに共通のモジュール

これらのモジュールは，NCU のアクティブファイルシステムおよび非アクティブファイルシステムに保管できます。

これらのファイルの分類を簡単にするために，規則に従ってそれぞれの末尾に拡張子を付けなければなりません。これらのデータの主な特徴は，ドメインサービスおよび変数サービスを使ってアクセスできることです。DOS との互換性を保つために，ファイル名拡張子が 3 文字を超えないようにしてください。

各モジュールタイプの特徴は，名前拡張子によって判別されます。アクティブファイルシステムおよび非アクティブファイルシステムのデータ構造については，「PI サービス」の章を参照してください。

表 11.7 概説：ドメインサービスの共通モジュール

EXT	モジュールタイプ	ファイルシステムエリア	意味
ACC	NCU 変数のアクセス記述	N，C，A，V，H，非アクティブ	NCU 変数にアクセス（ACCess）する必要のある NCK 変数のプロパティを説明します。
ASP	非同期サブプログラム	非アクティブ	
BIN	バイナリファイル	非アクティブ	BIN（バイナリファイル）は NC パートプログラム（メインプログラムとサブプログラムは区別されない）であり，コンパイルした形式かまたは一般のバイナリファイルとして使用できます。
COM	コメント	非アクティブ	COM（コメントファイル）。任意の ASCII テキスト
CEC	梁のたわみ / 誤差角度	N，非アクティブ	
CYC	サイクルプログラム	非アクティブ	
DIR	ディレクトリ	非アクティブ	タイプ DIR（ディレクトリ）のモジュールには，このグループでアクセスできるモジュールについての情報が含まれています。すべてのタイプのモジュールをこのディレクトリに入れることができます。規則による制約しか受けません（Data Scheme）。
EDI	エディタウィンドウ	N	ファイルを編集するには，ファイルのセクションが NCU メモリからロードされたら，エディタウィンドウで編集します。
EEC	エンコーダーのエラー補正	A，非アクティブ	
GUD	グローバル／チャネル固有ユーザデータ	N，C，非アクティブ	ユーザ（工作機械メーカー，OEM）は，モジュールタイプ 'user data'（Global User Data）を使用して，ユーザ自身のマシン固有のデータを作成することができます。これらのデータは，パートプログラムおよびサイクルからアドレス指定できます。MMC で構成する場合には，GUD も表示用にアドレス指定することができます。

INI	初期化モジュール	非アクティブ	初期化モジュール（INItial ファイル）は，物理的にはアクティブファイルシステムの一部ではありません。これには，特殊な機能があります。つまり，特定のタイプに限定されない変数の設定値が含まれています。初期化モジュールはアクティブファイルシステムに転送できますが，物理モジュールとしてそのシステムに存在するわけではありません。このモジュールに含まれる値は，ダウンロードプロセス中にアクティブファイルシステムの対応するモジュールで設定されます。
LUD	ローカル／プログラム固有ユーザデータ	C，非アクティブ	ユーザはプログラム内でローカルデータを定義することができます。
MPF	メインプログラム	非アクティブ	MPF（メインプログラムファイル）は，タイプ 'main program' の NC パートプログラムです。
OPT	オプションデータ	N，非アクティブ	オプションデータ（OPTion ファイル）
PRO	保護エリア	N，C，非アクティブ	
EXT	モジュールタイプ	ファイルシステムエリア	意味
QEC	象限エラー補正	A，非アクティブ	
RPA	計算パラメータ	C，非アクティブ	
SEA	設定データ	N，C，A，非アクティブ	設定データ（SEtting data Active）
SPF	サブプログラム	非アクティブ	SPF（サブプログラムファイル）は，タイプ 'sub program' の NC パートプログラムです。
SYF	システムデータ	非アクティブ	システムファイル（SYFile）には，完全なシステムについての情報が含まれます。
TEA	マシンデータ	N，C，A，V，H，非アクティブ	マシンデータ
TMA	マガジン説明	T，非アクティブ	以下のものが含まれます。 • 1 つのチャネル（エリア単位）で定義したマガジンのディレクトリ TMV（Tool Magazine Directory V） • 1 つのチャネル（エリア単位）のマガジン制御ブロック TMC（Tool Magazine Control） • 1 つのチャネル（エリア単位）で使用されるマガジンの記述 TM（Tool Magazine Active） • 1 つのチャネル（エリア単位）のマガジン位置の状態および占有量 TP（Tool Magazine-DataPlace Active） • 1 つのチャネル（エリア単位）のマガジン位置のタイプの階層（エリア単位）TT（ToolMagazine-Data Place Type） • マガジン位置への複数の割り当ての説明と，参照したマガジンどうしの外部位置を表す距離（Tool Magazine PlaceMulti）

TOA	工具補正	T，非アクティブ	工具補正（Tool Offset Active）： • 工具補正 TO（Tool Offset Active） • 一工具データ TD（Tool Data Active） • 研磨専用工具補正 TG（ToolGrinding Active） • 工具監視データ TS（Tool SupervisionActive） • ユーザ固有（OEM）工具データ TU（Tool UserActive） • 切削刃データ TUE（Tool User EdgeDataActive） • 工具ディレクトリ TDV（Tool Data Directory V）（エリア単位）
UFR	ユーザフレーム	C，非アクティブ	FRAMES（ユーザ入力フレーム）が含まれます（拡張原点オフセット拡張を設定可能）
WPD	ワークピースディレクトリ	非アクティブ	タイプ WPD のモジュール（ワークディレクトリ）には，このグループを介してアクセスできるモジュールについてのすべての情報が含まれています。各タイプのモジュールは，ワークピースディレクトリ（DIR および WPD を除く）に格納できます。

11.3 アクティブファイルシステムのモジュールタイプ

これらのモジュール（ファイル）は，非アクティブファイルシステムに保管することはできません。アクティブモジュールのみ保管できます。これらのモジュールの重要な特性として，変数サービスを介してのみアクセスできることが挙げられます。

表 11.8 アクティブファイルシステムのモジュールタイプ

データタイプ	アクティブファイルシステム内のエリア	タイプ	番号（16 進数）
システム状態データ	N, C	Y	10
NC 命令グループリスト	N	YNCFL	11
ユーザフレーム	C	FU	12
現行フレーム	C	FA	13
工具補正	T	TO	14
算術パラメータ	C	RP	15
設定データ	N,C,A	SE	16
グローバルユーザデータ GUD（=GD1=SGUD）	N,C	GUD	17
ローカルパートプログラムデータ	C	LUD	18
改変された補正	C	IK	19
マシンデータ	N,C,A,V*,H*	M	1A
コンパイル - サイクル	N	CC	1F
外部フレーム	C	FE	20
一般工具データ	T	TD	21
工具監視データ	T	TS	22
研磨用工具データ	T	TG	23
OEM 工具データ	T	TU	24
工具ユーザ切削刃データ	T	TUE	25

工具データディレクトリ	T	TV	26
マガジン説明	T	TM	27
マガジン位置データ	T	TP	28
マガジン位置への複数の割り当て	T	TPM	29
マガジン位置のタイプ	T	TT	2A
マガジンディレクトリ	T	TMV	2B
マガジン制御ブロック	T	TMC	2C
グローバルユーザデータ 2（=MGUD）	N,C	GD2	2D
グローバルユーザデータ 3（=UGUD）	N,C	GD3	2E
グローバルユーザデータ 4	N,C	GD4	2F
グローバルユーザデータ 5	N,C	GD5	30
グローバルユーザデータ 6	N,C	GD6	31
グローバルユーザデータ 7	N,C	GD7	32
グローバルユーザデータ 8	N,C	GD8	33
グローバルユーザデータ 9	N,C	GD9	34
記憶保護域	N,C	PA	35
ニブルデータ		NIB	37
状態データメッセージ（ドライブ，サーボ）		SDME	6E
プログラムポインタへの割り込み	C	SPARPI	6F
拡張状態データ形状軸	C	SEGA	70
拡張状態データマシン軸	C	SEMA	71
状態データ主軸	C	SSP	72
状態データ形状軸	C	SGA	73
状態データマシン軸	C	SMA	74
最新の状態データアラーム	N	SALAL	75
最も古い状態データアラーム	N	SALAP	76
状態データアラーム	N	SALA	77
状態データ同期アクション	C	SSYNAC	78
状態データブロックサーチポインタ	C	SPARPF	79
状態データプログラムポインタ	C	SPARPP	7A
状態データ NC 機能	C	SNCF	7B
状態データ割り込み状況	C	SINT	7C
状態データパートプログラム情報	C	SPARP	7D
状態データチャネル効果	C	SINF	7E
状態データ	N,B,C,V*,H*	S	7F

* 構文 ID 611D のデータ

11.3.1 例

例 1

次の変数アドレスを指定してください。2 番目のチャネルにある H 関数の値

一般的に，それぞれの変数および変数アドレスの値を指定すると，以下の DDE 接続になります。

LinkTopic NC-DDE|local

LinkItem /area /module type /column index

この例では，以下のことを意味します。

オンラインヘルプのセクション C には，ここで定義されるモジュールのリストが含まれます。モジュール SSYNAC および特に変数 Hval では，括弧内に索引の説明が含まれています。上部にある参照例を使用すると，リンク項目が PCU50 ではどのようなものかを表示することができます。

/Channel/SelectedFunctions/Hval[u<areaIndex>, <lineIndex>]

<areaIndex> = channel number

<lineIndex> = consecutive number

リンク項目は次のようになります。

/Channel/SelectedFunctions/Hval[u2,3]

例 2　工具補正データ

次は，やや複雑なアプリケーションの例を紹介します。TO エリアの工具補正データは必ず指定してください。

工具補正データの編成については，プログラミングの資料および関数の説明 W1 にあります。詳細については，オンラインヘルプを参照してください。

工具補正のリンク項目の一般的なフォーマットは次のとおりです。

Link Item general /Tool/Compensation/Edgedata [ux, cy, n (-m)]

ここで，それぞれ次のものを表します。

x は TO エリア

y は工具番号

n は工具パラメータ

m はオプションパラメータ

複数の切削刃のある工具では，以下のようにして，必要な切削刃をパラメータ n でアドレス指定します。

n = (CuttingEdgeNo - 1) * MaxNumParam + ParamNumber

ここで，MaxNumParam = 25（工具パラメータの最大数）であり，パラメータの範囲は 1 ～ 25 になります（例 2.3 を参照）。

■ 例 2.1

TO エリア 1 にある 2 番目の工具で最初の切削刃の長さ 1 をサーチ : します。

Link Item /Tool/Compensation/Edgedata[u1,c2,3]

ここで，それぞれ次のものを表します。

u1 は TO エリア 1

c2 は 2 番目の工具

3 は形状長さ補正（パラメータ 3)。

■ 例 2.2

TO エリア 1 にある 2 番目の工具で最初の切削刃の長さ 1 と長さ 2 をサーチします。:

Link Item /Tool/Compensation/Edgedata[u1,c2,3,4]

ここで，それぞれ次のものを表します。

u1 は TO エリア 1

c2 は 2 番目の工具

3 は形状長さ補正（パラメータ 3）

4 は形状長さ補正（パラメータ 4）

■ 例 2.3

TO エリア 1 にある 3 番目の工具で 2 番目の切削刃の長さ 2 をサーチします。:

Link Item /Tool/Compensation/Edgedata[u1,c3,29]

ここで，それぞれ次のものを表します。

u1 は TO エリア 1

c3 は 3 番目の工具

29 は 2 番目の切削刃の形状長さ補正（パラメータ 4）

（25 を追加します）

11.4 マシンデータ

パートプログラムが "$ 変数 " としてアクセスできるシステムデータの多くも，MMC によってアクセスすることができます。

表 11.9 PCU50 での $ 変数の変数名

NC 変数へのポインタ	変数名
$A_DBB[x] x = ByteNo	aDbb
$A_DBD[x] x = Offset	aDbd
$A_DBR[x] x = Offset	aDbr
$A_DBW[x] x = Offset	aDbw
$A_IN[x] x = DigitalinputNo	digitInpVal
$A_INA[x] x = AnaloginputNo	analogInpVal
$A_INCO[x] x = InputNo	aInco
$A_OUT[x] x = DigitaloutputNo	digitOutpVal
$A_OUTA[x] x = AnalogoutputNo	analogOutpVal
$A_TOOLMLN[x] x = ToolNo	toolInPlace
$A_TOOLMN[x] x = ToolNo	toolInMag
$AA_COUP_ACT[x] x = Spindle	aaCoupAct
$AA_COUP_OFFS[x] x = Axis	cmdCouppPosOffset

$AA_COUP_OFFS[x] x = Spindle	aaCoupOffs
$AA_CURR[x] x = Axis	aaCurr
$AA_DELT[x] x = Axis	aaDelt
$AA_DTBB[x] x = Axis	aaDtbb
$AA_DTBW[x] x = Axis	aaDtbw
$AA_DTEB[x] x = Axis	aaDteb
$AA_DTEPB[x] x = Axis	aaDtepb
$AA_DTEPW[x] x = Axis	aaDtepw
$AA_DTEW[x] x = Axis	aaDtew
$AA_ETRANS[x] x = FrameNo	linShift
$AA_FXS[x] x = Axis	fxsStat
$AA_IM[x] x = Axis	cmdToolBasePos
$AA_IW[x] x = Axis	cmdToolBasePos
$AA_LEAD_P[x] x = Axis	aaLeadP
$AA_LEAD_SP[x] x = Axis	aaLeadSp
$AA_LEAD_SV[x] x = Axis	aaLeadSv
$AA_LEAD_TYP[x] x = Axis	aaLeadTyp
$AA_LEAD_V[x] x = Axis	aaLeadV
$AA_LOAD[x] x = Axis	aaLoad
$AA_MM[x] x = Axis	aaMm
$AA_MM1[x] x = Axis	aaMm1
$AA_MM2[x] x = Axis	aaMm2
$AA_MM3[x] x = Axis	aaMm3
$AA_MM4[x] x = Axis	aaMm4
$AA_MW[x] x = Axis	aaMw
$AA_OFF[x] x = Axis	aaOff
$AA_OFF_LIMIT[x] x = Axis	aaOffLimit
$AA_OSCILL_REVERSE_POS1[x] x = Axis	aaOscillReversePos1
$AA_OSCILL_REVERSE_POS2[x] x = Axis	aaOscillReversePos2
$AA_OVR[x] x = Axis	aaOvr
$AA_POWER[x] x = Axis	aaPower
$AA_S[x] x = SpindleNo	cmdSpeed
$AA_SOFTENDN[x] x = Axis	aaSoftendn
$AA_SOFTENDP[x] x = Axis	aaSoftendp
$AA_STAT[] aaStat$AA_SYNC[x] x = Axis	aaSync
$AA_TORQUE[x] x = Axis	aaTorque
$AA_TYP[x] x = Axis	aaTyp
$AA_VACTB[x] x = Axis	actFeedRate
$AA_VACTM[x] x = Axis	cmdFeedRate
$AA_VACTW[x] x = Axis	actFeedRate
$AA_VC[x] x = Axis	aaVc
$AC_DELT acDelt$AC_DRF[x] x = Axis	drfVal
$AC_DTBB	acDtbb
$AC_DTBW	acDtbw
$AC_DTEB	acDteb
$AC_DTEPB	acDtepb

$AC_DTEPW	acDtepw
$AC_DTEW	acDtew
$AC_FCT0[x] x = PolynomNo	acFct0
$AC_FCT1[x] x = PolynomNo	acFct1
$AC_FCT2[x] x = PolynomNo	acFct2
$AC_FCT3[x] x = PolynomNo	acFct3
Pointer to the NC-variable	変数名
$AC_FCTLL[x] x = PolynomNo	acFctll
$AC_FCTUL[x] x = PolynomNo	acFctul
$AC_FIFOx[y] x = FIFONo y = ParameterNo	acFifoN
$AC_MARKER[x] x = MarkerNo	acMarker
$AC_MEA	acMea
$AC_OVR	avOvr
$AC_PARAM[x] x = ParameterNo	acParam
$AC_PATHN	acPathn
$AC_PLTBB	acPltbb
$AC_PLTEB	acPlteb
$AC_PRESET[x] x = Axis	PRESETVal
$AC_RETPOINT[x] x = Axis	acRetpoint
$AC_SDIR[x] x = SpindleNo	turnState
$AC_SMODE[x] x = SpindleNo	opMode
$AC_TIME	acTime
$AC_TIMEC	acTimec
$AC_TIMER[x] x = TimerNo	acTimer
$AC_TRAFO	acTrafo
$AC_VACTB	cmdFeedRateIpo
$AC_VACTW	acVactw
$AC_VC	avVc
$AN_MARKER[x] x = MarkerNo	anMarker
$P_PFRAME または $P_ACTFRAME または $P_IFRAME	linShift
$P_PFRAME または $P_ACTFRAME または $P_IFRAME	mirrorImgActive
$P_PFRAME または $P_ACTFRAME または $P_IFRAME	rotation
$P_PFRAME または $P_ACTFRAME または $P_IFRAME	scaleFact
$P_TOOL	actDNumber
$P_TOOLL[1]	actToolLength1
$P_TOOLL[2]	actToolLength2
$P_TOOLL[3]	actToolLength3
$P_TOOLND[x] x = ToolNo	numCuttEdges
$P_TOOLNO	actTNumber
$P_TOOLR	actToolRadius
$P_UIFR[x] x = FrameNo	linShift
$P_UIFR[x] x = FrameNo	mirrorImgActive
$P_UIFR[x] x = FrameNo	rotation
$P_UIFR[x] x = FrameNo	scaleFact

$P_UIFRNUM	actFrameIndex
$R[x] x = ParameterNo	R
$SC_PA_ACTIV_IMMED[x] x = Number protection zone	MDU_PA_ACTIV_IMMED
$SC_PA_CENT_ABS[x,y] x = Number protection zone	MDD_PA_CENT_ABS_y
$SC_PA_CENT_ORD[x,y] x = Number protection zone	MDD_PA_CENT_ORD_y
$SC_PA_CONT_ABS[x,y] x = Number protection zone	MDD_PA_CONT_ABS_y
$SC_PA_CONT_NUM[x] x = Number protection zone	MDU_PA_CONT_NUM
$SC_PA_CONT_ORD[x,y] x = Number protection zone	MDD_PA_CONT_ORD_y
$SC_PA_CONT_TYP[x,y] x = Number protection zone	MDU_PA_CONT_TYP_y
$SC_PA_LIM_3DIM[x] x = Number protection zone	MDU_PA_LIM_3DIM
$SC_PA_MINUS_LIM[x] x = Number protection zone	MDD_PA_MINUS_LIM
$SC_PA_ORI[x] x = Number protection zone	MDU_PA_ORI
$SC_PA_PLUS_LIM[x] x = Number protection zone	MDD_PA_PLUS_LIM
$SC_PA_T_W[x] x = Number protection zone	MDU_PA_TW
$SN_PA_ACTIV_IMMED[x] x = Number protection zone	MDU_PA_ACTIV_IMMED
$SN_PA_CENT_ABS[x,y] x = Number protection zone	MDD_PA_CENT_ABS_y
$SN_PA_CENT_ORD[x,y] x = Number protection zone	MDD_PA_CENT_ORD_y
$SN_PA_CONT_ABS[x,y] x = Number protection zone	MDD_PA_CONT_ABS_y
$SN_PA_CONT_NUM[x] x = Number protection zone	MDU_PA_CONT_NUM
$SN_PA_CONT_ORD[x,y] x = Number protection zone	MDD_PA_CONT_ORD_y
$SN_PA_CONT_TYP[x,y] x = Number protection zone	MDU_PA_CONT_TYP_y
$SN_PA_LIM_3DIM[x] x = Number protection zone	MDU_PA_LIM_3DIM
$SN_PA_MINUS_LIM[x] x = Number protection zone	MDD_PA_MINUS_LIM
$SN_PA_ORI[x] x = Number protection zone	MDU_PA_ORI
$SN_PA_PLUS_LIM[x] x = Number protection zone	MDD_PA_PLUS_LIM
$SN_PA_T_W[x] x = Number protection zone	MDU_PA_TW
$TC _DPC1[x,y] .. $TC_DPC10[x,y] x=ToolNo y=EdgeNo	edgeData
$TC _DPC1[y,z] ... $TC_DPC10[y,z]x=ToolNo y=EdgeNo	dummy
$TC_DPx[y,z] x = ParamNo y = ToolNo z = EdgeNo	cuttEdgeParam
$TC_DPx[y,z] x = ParamNo y = ToolNo z = EdgeNo	edgeData
$TC_MAP1	magKind
$TC_MAP2	magIdent
$TC_MAP3	magState
$TC_MAP4	magLink1
$TC_MAP5	magLink2
$TC_MAP6	magDim
$TC_MAPCx[y] x = ParameterNo y = MagazineNo	data
$TC_MOP1(x,y) ... $TC_MOP4(x,y) x=ToolNo y=EdgeNo	dummy
$TC_MOP1[y,z] ... $TC_MOP4[y,z]	data
$TC_MOPCx[y,z] x=ParameterNo y=ToolNo z=EdgeNo	data
$TC_MPPCx[y,z] x=ParamNo y=MagazineNo z=MagPlaceNo	data
$TC_TP1	toolIdent
$TC_TP10	toolSearch
$TC_TP11	toolInfo
$TC_TP2	duploNo
$TC_TP3	toolsize_left

$TC_TP4	toolsize_right
$TC_TP5	toolsize_upper
$TC_TP6	toolsize_down
$TC_TP7	toolplace_spec
$TC_TP8	toolState
$TC_TP9	toolMon
$TC_TPC1[x] ... $TC_TPC10[x] x = ToolNo	dummy
$TC_TPCx[y] x = ParameterNo y = ToolNo	data
$TC_TPG1	spinNoDress
$TC_TPG2	conntectPar
$TC_TPG3	minToolDia
$TC_TPG4	minToolWide
$TC_TPG5	actToolWide
$TC_TPG6	maxRotSpeed
$TC_TPG7	maxTipSpeed
$TC_TPG8	inclAngle
$TC_TPG9	paramNrCCV
$VA_COUP_OFFS[x] x = Axis	actCouppPosOffset
$VA_IM[1]	measPos1
$VA_IM[2]	measPos2
$VA_IS[x] x = Axis	safeMeasPos
$VA_VACTM[x] x = Axis	vaVactm

11.5 PLC – データ

11.5.1 概要

入力，出力，フラグデータおよびデータブロックは，NC 変数と同様にアクセスできます。変数 ID は 3 つの部分で構成されています。その他に，DDE インタフェースでのウォーム／ホットリンクの巡回時間を設定するにはオプションパラメータ（-FAST）を指定します。

表 11.10 PLC データの変数名

1 部	/PLC	名前の定数部分
2 部	/Input/Output/ Memory/DataBlock/ Directory	入力（内部イメージ）にアクセス出力（内部イメージ）にアクセスフラグにアクセスデータブロックにアクセス PLC のディレクトリにアクセス
3 部	/Bit	アクセス幅 : 1 ビット
	/Byte	アクセス幅 : 1 バイト（8 ビット）
	/Word	アクセス幅 : 1 ワード（2 バイト，16 ビット）
	/DoubleWord	アクセス幅 : 1 ダブルワード（4 バイト，32 ビット）
	/Float	浮動数として解釈される，連続する 4 バイトにアクセス
	/Double	浮動数として解釈される，連続する 8 バイトにアクセス
	/Hierarchy1	すべてのモジュールタイプのリストにアクセス
	/Hierarchy2	1 つのモジュールタイプのリストにアクセス
	/Hierarchy3	1 つの特定のモジュールタイプのリストにアクセス

	/S7Timer	タイマ内容にアクセス（ミリ秒単位）
	/S7Counter	カウンタにアクセス（フォーマットは整数）
	/TimeAndDate	クロックにアクセス（フォーマットは以下を参照）
	/Statelist/Complete	PLC の完全なシステム状態リストにアクセス
	/Messages	PLC メッセージ／アラームにアクセス
4 部	パラメータなし	PLC 変数への巡回サービスの場合，スキャン／巡回間隔は 500 ミリ秒
	パラメータ : -FAST	同じクラスタのどの変数（PLC 変数およびその他の変数）でも，スキャン／巡回間隔は 100 ミリ秒

索引

特定のビット，バイトなどは，通常，索引を介してアドレス指定されます。索引は次のように理解できます。

- ND-DDE サーバの側から見ると，行索引は Byte/Word/DWord オフセットを指定します。
  重要 **!** S7 では異なることに注意してください。S7 はバイトオフセットしか使用しません !
- ビットがアクセス実行中の場合，ビット番号に行索引を追加しなければなりません。それには，0 ～ 7 の範囲の数を '.' で区切って追加します。.
- データブロックがアクセス実行中の場合，列索引 c によってブロックが指定されます。

重要事項 !
NC-DDE サーバーによる PLC データのアドレス指定は，S7 プログラムで使用されるアドレス指定とは異なります !

例 : ダブルワード 5（DWORD 5）のアドレス指定

表 11.11 PLC プログラムによるアドレス指定と NCDDE サーバによるアドレス指定の違い

変数	S7 の PLC プログラムによるアクセス :	NCDDE サーバによるアクセス :
DWORD 5	WORD 5 +6	WORD 10 + 11
	BYTE 5 +6 +7 +8	BYTE 20 + 21 + 22 + 23

例 :

有効な PLC 変数 ID は，以下のとおりです。

/Plc/Input/Word[9] 入力バイト 18 および 19 にアクセス
/Plc/Memory/Bit[9.3] フラグデータバイト 9 のビット 3 にアクセス data byte 9
/Plc/Output/Doubleword[7] 28 ～ 31 の出力バイトにアクセス
/Plc/DataBlock/Bit[c100,9.3] データブロック 100 のバイト 9 のビット 3 にアクセス

すべての PLC データの説明は，『840DI の Installation & Startup Guide』の「PLC Interface」の章にあります。したがって，本書ではこれらのデータについて詳しく解説していません。ご使用の MMC アプリケーションで使用しようとしているデータを，ファイル PLC202.NSK にしたがって現行の NSK ファイルに入力しなければなりません。そうすると，次のシステム始動の際にそれらのデータが認識されるようになります。

11.5.2 PLC データの要約

表 11.12 MMC での PLC データの変数名

データタイプ	パラメータ範囲	ID，パラメータ
PLC 入力		
入力ビット	0 ～ 7	/Plc/Input/Bit[入力バイトビット]
入力バイト	0 ～ 63	/Plc/Input/Byte[入力バイト]
入力ワード	0 ～ 31	/Plc/Input/Word[入力ワード]
入力ダブルワード	0 ～ 15	/Plc/Input/DoubleWord[入力ダブルワード]
PLC 出力		
出力ビット	0 ～ 7	/Plc/Output/Bit[出力バイトビット]
出力バイト	0 ～ 63	/Plc/Output/Byte[出力バイト]
出力ワード	0 ～ 31	/Plc/Output/Word[出力ワード]
出力ダブルワード	0 ～ 15	/Plc/Output/DoubleWord[出力ダブルワード]
PLC フラグデータ		
フラグデータビット	0 ～ 7	/Plc/Memory/Bit[フラグデータバイトビット]
フラグデータバイト	0 ～ 255	/Plc/Memory/Byte[フラグデータバイト]
フラグデータワード	0 ～ 127	/Plc/Memory/Word[フラグデータワード]
フラグデータダブルワード	0 ～ 63	/Plc/Memory/DoubleWord[フラグデータダブルワード]
PLC データブロック		
データビット	0 ～ 7	/Plc/DataBlock/Bit[cデータブロック, データバイト.ビット]
データバイト	0 ～ 1023	/Plc/DataBlock/Byte[c データブロック , データバイト]
データワード	0 ～ 511	/Plc/DataBlock/Word[c データブロック , データワード]
データダブルワード	0 ～ 255	/Plc/DataBlock/DoubleWord[cデータブロック ,データダブルワード]
浮動小数点		/Plc/DataBlock /Float[c データブロック , データバイト]
PLC タイマ	**0 ～ 127**	/Plc/S7timer[タイマ番号]
PLC カウンタ	**0 ～ 63**	/Plc/S7Counter[カウンタ番号]
PLC クロック	1	/Plc/TimeAndDate[月 . 日 . 年 時 : 分 : 秒 : ミリ秒 曜日 状態]
PLC 階層		
すべてのモジュールのリスト	1	/Plc/Directory/Hierarchy1
1 つのモジュールタイプのリスト	1	/Plc/Directory/Hierarchy2[" データブロックタイプ "]
1 つのモジュールのリスト	1	/Plc/Directory/Hierarchy3[" データブロック名 "]

PLC システム状態	**1**	/Plc/Statelist/Complete
PLC アラーム	**1**	/Plc/Messages

PLC: カウンタ，タイマ，およびクロック

PLC のカウンタ，タイマ，およびクロックに対応する変数 ID は，以下のとおりです。

/Plc/S7Timer
/Plc/S7Counter
/Plc/TimeAndDate

タイマおよびカウンタは行索引で選択されます。DDE インタフェースでは，カウンタおよびタイマは実数として提供されます。タイマ値は，通常はミリ秒になります。オプションとして -FAST を追加指定すれば，処理速度が向上します（標準の 500 ミリ秒サイクルから 100 ミリ秒サイクルに）。

変数 "/Plc/TimeAndDate" を使用すると，PLC の日付および時刻にアクセスできます（ウォーム／ホットリンクは使用不可）。PLC クロックにアクセスするには，必ずクリップボードフォーマット CF_TEXT を使用してください。データセットには以下の項目ごとにエリアがあります。

- 日付 ("<month> . <day> . <year>")
- 時刻 ("<hours> : <minutes> : <seconds> . <mill seconds>")
- 曜日 Dow ("SUN", "MON", "TUE", "WED", "THU", "FRI" または "SAT")
- クロック状況（16 進コード）

それぞれはスペース文字で区切ります。

PLC クロックの状況は，16 進数形式を使って <Status> に戻されます。ビット 0 ～ 15 のうち，ビット 3 およびビット 4 では，解像度を指定します。割り当ては，以下の表に示されています。

表 11.13 クロックの解像度

ビット 3	**ビット 4**	**解像度**
0	0	1ms
0	1	10ms
1	0	100 ms
1	1	1000 ms

たとえば，1999 年 7 月 26 日，月曜日，15:30，解像度が 1 秒の場合
:<month>.<day>.<year> <hour>:<min>:<sec>.<millisec> <DoW><status>

07.26.99 15:30:00:000 MON 0018

PLC: メッセージおよびアラーム

変数 "/Plc/Messages" は，PLC のメッセージおよびアラームシステムへのインタフェースを提供します。この変数へのホットリンクは，PLC メッセージおよびアラームを戻します。MMC はリアルタイムシステムではないことに注意してください。したがっ

て，すべてのメッセージが返されることは保証できません。

レジストリモジュールを介してメッセージ／アラームにアクセスすると，この制限の影響を最大限回避することができます。詳細については，9 章「アラーム - サーバー」で説明されています。

オプションの行索引を使用すると，特定のメッセージをフィルタ処理できます。索引がアドバイスリンクで指定されている場合には，索引のビット 2（最初のメッセージニブル）が設定されている場合にのみメッセージが戻されます。索引がない場合には，使用可能なすべてのメッセージ／アラームが戻されます。アクセスは，バイナリ形式で実行しなければなりません。

PLC: システム状態

変数 "/Plc/Statelist/Complete" を使用すると，PLC の完全なシステム状態リストを読み取ることができます（ウォーム／ホットリンクは使用不可）。アクセスは，バイナリ形式でのみ実行できます。

PLC ブロックリスト

以下のディレクトリ変数

/Hierarchy1 すべてのモジュールタイプのモジュールリストにアクセス
/Hierarchy2 1 つのモジュールタイプのモジュールリストにアクセス
/Hierarchy3 特定のモジュールのモジュールリストにアクセス

は，PLC ディレクトリ情報への読み取りアクセスを許可します（ウォーム／ホットリンクは使用不可）。

変数 "/Plc/Directory/Hierarchy1" にはパラメータはありません。

いくつかのデータセット（<CR><LF> で区切られる）では，この変数は 1 つのモジュールタイプで使用できるモジュールの数を戻します。それぞれのデータセットには，フィールドモジュールタイプと数が <TAB> で区切られて入っています。あるモジュールで，PLC の受動ファイルシステムに使用可能な変形があり，PLC のアクティブファイルシステムにも使用可能な変形がある場合（チェーンモジュール），どちらもカウントされます。

例 :

PLC に，3 つの OB（タイプ 08），8 つの FB（タイプ 0E），1 つの FC（タイプ 0C），そして 4 つの DB（タイプ 0A）があるとします。

変数 : /Plc/Directory/Hierarchy1

結果 : 08 <TAB>3 <CR><LF>
0E <TAB> 8 <CR><LF>
0C <TAB> 1 <CR><LF>
0A <TAB> 4

変数 "/Plc/Directory/Hierarchy2" では，パラメータとしてモジュールタイプが指定されるはずです。

いくつかのデータセット（<CR><LF> で区切られる）では，この変数はモジュールについての短い記述を提供します。それぞれのデータブロックには，次のフィールドが <TAB> で区切られて入っています。

- モジュール番号
- モジュールの状況（16 進形式でコード化される）
- 保管場所（値 : "RAM"，"EPROM"，"???"）
- 状態（値 : "ACTIVE"，"PASSIVE"，"???"）

例 :

PLC で，RAM にタイプ OE の 8 つの機能モジュールがあり，アクティブであるとします。

変数 : /Plc/Directory/Hierarchy2["0E"]

結果 :
1 <TAB> 1201 <TAB> RAM <TAB> ACTIVE <CR><LF>
2 <TAB> 1201 <TAB> RAM <TAB> ACTIVE <CR><LF>
.....
8 <TAB> 1201 <TAB> RAM <TAB> ACTIVE <CR><LF>

変数 "/Plc/Directory/Hierarchy3" では，パラメータとして attribute1 のないモジュール名が指定されるはずです。

（Attribut1 は "$"（アップロード用のモジュールヘッダ）または "_"（完全なモジュール）の付いたファイル ID です）。

この変数は，情報が受動（P），アクティブ（A），または両方のモジュール（B）のどれで要求されたかに応じて，指定されたモジュールに 1 つまたは 2 つのデータセット（<CR><LF> で区切られる）を提供します。それぞれのデータセットには，次のフィールドが <TAB> で区切られて入っています。

- モジュール名
- モジュールの状況（16 進形式でコード化）
- 保管場所（値 : "RAM"，"EPROM"，"???"）
- 状態（値 : "ACTIVE"，"PASSIVE"，"???"）
- モジュールヘッダ（16 進形式でコード化され，スペース文字で区切られた 6 個のダブルワード : 24 バイト）
- オプションのヘッダ拡張（例，46 バイト。16 進フォーマットでコード化され，各ワードをスペース文字で区切った 11 個のダブルワード）。

例 : PLC では，編成モジュール OB1 がアクティブになっています。

変数 : /Plc/Directory/Hierarchy3["0800001A"]

結果 :
0800001A <TAB> 1230 <TAB> RAM <TAB> ACTIVE <TAB>
07070001 01080001 000000f2 00000000 01edce06 110c01ed <CR>
ce06110c .... <CR><LF> .....

11.6 PI サービス

11.1.4 およびオンラインヘルプには，利用可能な PI サービスのリストがあります。

パラメータ指定

NC の PI 名は _N_ で始まり，その後に 6 文字が続きます。PLC サービス名の規則は，これとは少し異なります。

PI サービスを呼び出すコマンド行は，次のように構成します。

PI_START(Server-Name, Parameter 1, Parameter 2 ... Parameter n,PI-Name)

表 11.14 PI サービスへのパラメータの提供

PI 記号 ID	PI 数値 ID
最初の引き数（名前）	引き数の説明
2 番目の引き数（名前）	引き数の説明
:	:
n 番目の引き数（名前）	引き数の説明
PI 記号 ID	

最初のパラメータは，領域または単位を指定します。

次の ID を使用します（xx は 2 桁を表します）。

001 NCK 固有サービス
1xx BAG 固有サービス
2xx チャネル固有サービス
3xx 軸固有サービス
4xx ツール固有サービス
5xx VSA 固有サービス
6xx HSA 固有サービス
K00 COM モジュールに関するサービス
P01 受動ファイルシステム用サービス

パラメータの記述では，以下のパスおよびファイル名の表記を使用します。

xxx = DIR または WPD（すなわち，ディレクトリ）
yyyyy = ファイル名またはディレクトリ名を表す任意のストリング（最大 25 文字）
zzz = ファイル拡張子を表す任意のストリング（最大 3 文字）

例 : PI サービス汎用 SELECT を使ったパートプログラムの選択 :

PI_START(NC, channel, path name/program name, _N_SELECT).

以下のパラメータ
channel 1 = 201 および
path name/program name = MPF_DIR/TEST_MPF
は結果として以下のコマンド行になります。

PI_START(NC, 201, /_N_MPF_DIR/_N_TEST_MPF, _N_SELECT).

11.7 NCDDE - エラーメッセージ

NC-DDE サーバは，接続に固有のエラー変数（lastError）を提供します。この変数は，コマンドの否定応答の後に「Request」を使って評価できます。「Request」に続いて，lastError は自動的に 00 00 00 00 にリセットされます。

11.7.1 変数 LastError

変数「lastError」は long int 型の値で，以下の 4 つの構成要素（バイト）で構成されます。

1 番目のバイト	2 番目のバイト	3 番目のバイト	4 番目のバイト
一般エラークラス番号	エラー領域番号	エラークラス番号	エラーコード番号

DDETest を使用すると，右側の列に 8 桁の 16 進値が表示されます。戻される変数に 5 桁しか含まれていない場合には，以下の表を適用する前に先行ゼロを追加してください。

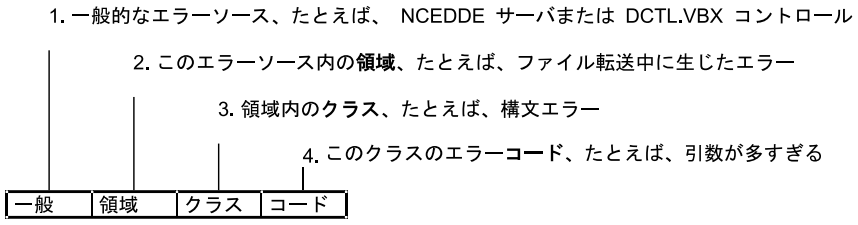
4 つのセクションの特徴は次のとおりです。

一般的なエラーソース，たとえば，NCDDE サーバまたは DCTL.OCX コントロール

このエラーソース内の領域，たとえば，ファイル転送中に生じたエラー

領域内のクラス，たとえば，構文エラー

このクラスのエラーコード，たとえば，引き数が多すぎる



デフォルト

00	00	00	00

11.7.2 一般エラークラス定義

一般	領域	クラス	コード

表 11.15 一般エラークラス

番号	ID	説明

00	UNKNOWN_GENERAL_CLASS	一般クラスは（まだ）設定されていません
01	NCDDE_ERROR	ncdde サーバがエラーを生成しました
02	MPI_ERROR	profibus インタフェースでの層 4 移送エラー，エラーの解釈の詳細については，このファイルの profibus インタフェースエラーのセクションを参照してください。
03	L7_ERROR	NC から通知される層 7 エラー この場合，エラークラスとエラーコードは，対応する S7 プロトコルのエントリ（ERRCLS/ERRCOD）から取られています
04	L7_ADDITIONAL_ERROR	サーバのプロトコルハンドラで検出される層 7 エラー
05	L7_ACCESS_ERROR	NC から通知される層 7 エラー エラーコードは，「アクセス結果」S7 プロトコルのエントリから取られています
06	L4_ERROR	他の層 4 エラー この場合，エラークラスとエラーコードは，層 4 エラー表から取られています
07	DCTL_ERROR	DCTL コントロールによって検出されたエラー
08	DOSERRNO_ERROR	エラーは DOS 呼び出しによって送信されました エラーコード／エラークラスは「errno」値を保持します
09	DDEML_ERROR	ddeml インタフェースでの予期しないエラー エラーコード／エラークラスは ddeml エラー値を保持します

！重要 3 番目および 4 番目のバイトの意味は，最初のバイトの値によって決まります。

11.7.3 エラー領域の定義

XX	領域	クラス	コード

表 11.16 エラー領域

番号	ID	説明
00	UNKNOWN_REGION	領域が設定されていません
01	FT_ERROR	ファイル転送エラー
04	REQUEST_ERROR	単一変数の読み取りエラー
05	POKE_ERROR	単一変数の書き込みエラー
06	DIAGNOSE_ERROR	診断結果の告知または非告知時のエラー
08	PI_ERROR	pi コマンドエラー
09	EXECUTE_ERROR	dde-exec-cmd 実行エラー
0A	ADVISE_ERROR	勧告エラー
0B	CLIENT_ERROR	XACT_COMPLETE トランザクション中のエラー
0E	COMIC_ERROR	comic コマンド実行エラー
F3	NO_ACCESS	領域情報へのアクセスはありません

F4	NO_ERROR	内部のみ
FF	NO_REGION	エラーが生じるまで領域は認識されません

11.7.4 general = 1, 4 および 6 の詳細

01, 04, 06	xx	クラス	xx

表 11.17 general= 1, 4, 6 のエラークラス

クラス	ID	説明
00	UNKNOWN_CLASS	エラークラスは設定されていません
01	TIMEOUT_ERROR	通信の終了を待っていてタイムアウトになったため，エラーコードは，使用したタイムアウト［秒］を表示します
02	TRANSPORT_ERROR	層 4 によって送信されたエラー
03	DDE_ERROR	DDE インタフェースでのエラー
04	GENERAL_ERROR	他のエラークラス
05	FILEIO_ERROR	ferror 呼び出しによって送信されたエラー
06	SYNTAX_ ERROR	構文エラーが検出されました
07	ARGUMENT_ERROR	ユーザ引き数の 1 つが受け入れられません
08	PDU_ACCESS_ERROR	「Zugriffsergebnis」エントリへの設定は PDU 全体を意味します
09	VAR_ACCESS_ERROR	「Zugriffsergebnis」エントリの設定は単一変数を意味します
0A	PROTOCOL_ERROR	NC 通信によって生じたエラー
0B	CONVERSION_ERROR	データの変換時に生じたエラー
0C	DOM_STATUS	ISO 層 7 エラー: エラーコードとしてドメイン転送状況が続きます
0D	CFILE_ERROR	CFileException によって送信されるエラー。これは入出力エラーです
0E	COMIC_ERROR	コミック内で生じるエラー

01, 04, 06	xx	xx	コード

表 11.18 general= 1, 4, 6 のエラーコード

コード	ID	説明
00	UNKNOWN_CODE	エラーコードは設定されていません
01	LOCATION_ERROR	層 4 アドレスが受け入れられない場合に使用されます
02	UNKNOWN_TOPIC_ERROR	不明なトピック仕様との通信を試行します
03	UNKNOWN_ITEM_ERROR	不明な項目仕様との通信を試行します
04	UNKNOWN_CLIPFORMAT_ERROR	不明なクリップボード形式との通信を試行します
05	ADRMAP_ERROR	アドレスのマッピングがありません
06	ARGCNT_ERROR	引き数の数が受け入れられません
07	ARG_ERROR	引き数が受け入れられません

08	UNKNOWN_CMD_ERROR	コマンドが認識されません
09	TOKENIZE_ERROR	コマンドが認識されません
0A	VARIABLE_IN_USE_ERROR	変数は現在使用中です
0B	VARIABLE_TYPE_ERROR	変数タイプが受け入れられません
0C	VARIABLE_CREATION_ERRO R	変数作成がエラーです（メモリ不足）
0D	NODATA_ERROR	アクセス可能または送信可能なデータがありません
0E	FRAMING_ERROR	誤った PDU 構造が検出されました
0F	PDU_TOO_LARGE_ERROR	クラスタが大きすぎます
10	TOO_MANY_ARGS_ERROR	引き数の数値が大きすぎます
11	CAN_NOT_CONVERT	データ変換を行うことができません
12	DATA_CORRUPTED	データの不整合が検出されました
13	UNDEF_FORMAT	LINK コマンドの形式が不明です
14	DATA_TOO_LARGE	データが大きすぎて処理できません
15	SLICE_NOT_ALLOWED	索引範囲は許可されていません
16	SAME_ITEM_ADVISED	この項目はすでにこの接続で勧告されています
17	REMOTE_DOWN	このリモート装置とは通信できません
18	CONNECTING	すぐに通信できます
19	BROKEN_LINE	通信不可です
1A	NO_LINE	通信不可です
1B	UNSUPPORTED_INDEX	変数索引は受け入れられません
1C	FORMATTING_ERROR	テキストの変換形式が正しくありません
1D	UNEXPECTED_PDU	この種の PDU は予期されておらず，PDU 構文エラーです
1E	FTSTOP_ERROR	ファイル転送を中断する際は，下位のエラーコードのバイトを両方とも設定する必要があります
1F	EXIT_ABORT	ncdde の終了が取り消されました
20	UNKNOWN_TRANSPORT_TYP E	要求されたトランスポートクラスは使用できません
21	OUT_OF_RESOURCES	一部のリソースを使い果たしました
22	CAN_NOT_OPEN_DEV	トランスポートデバイスを開くことができません
23	CAN_NOT_INIT_DDEML	DDE の初期化呼び出しは失敗しました
24	NO_SERVER	NC への DDE 接続は失敗しました
25	NC_NOT_RESPONDING	NC ライフサインが停止しました
26	BOOT_REJECTED	NCがブート要求／ブートデータを拒否しました
27	SYNC_ERROR	NC が同期していません
28	OUT_OF_MEMORY	メモリが不足しています
29	BUFOVERFLOW_ERROR	内部固定長バッファが小さすぎます
2A	PROTOCOL_UNKNOWN	使用できる L7 プロトコルハンドラがありません
2B	OUT_OF_RESOURCES	ウィンドウのリソースを使い果たしました
2C	ADRMAP_SELF	ADDRESS: /SELF が定義されていないか，構文エラーです
2D	HEADER_CORRUPTED	ヘッダーをデコードできないため，ファイル転送は失敗しました

2E	ARITH_ERROR	演算式の評価エラーです。未知の変数があるか, 変数タイプが正しくありません
2F	INDEX_ERROR	予期しない索引タイプです
30	ITEM_SYNTAX	項目のデコードが構文エラーです
31	VALUE_SYNTAX	値をデコードできません
32	LINK_SYNTAX	リンクステートメントのテキストパラメータをデコードできません
33	STRING_SYNTAX	文字列の構文が正しくありません
34	ATOM_TUNING	ncdde のアトムテーブルがオーバーフローしています
35	INDEX_NOTAVAIL	索引のデータが欠落しています
36	EXIT_REJECT	ncdde の終了中に拒否されました
37	SHARED_MEMORY_SYNTAX	共用メモリーのセレクタをデコードできません
38	SHARED_MEMORY_ACCESS	送信された共用メモリーのセレクタを使用できません
39	INVALID_TIME	タイムコードが破壊されました
3A	SHARED_MEMORY_EXHAUSTED	SHM が小さすぎます
3B	SHARED_MEMORY_CORRUPTED	SHM が受け入れられません
3C	MAP_ERROR	acc ファイル項目をマップできませんでした
3D	PIPE_IN_USE_ERROR	パイプ変数のデータ受信側はすでに存在しています
3E	EOF_ERROR	EOF 書き込みの後にパイプ変数を書き込みました
3F	NO_TRANSFER	このアクセスでファイルの転送は行われません
40	DDEINIT_ERROR	DDE-NC インタフェースの初期化エラー
41	UNKNOWN_COMIC	指定したコミックは存在しません
42	COMIC_ALREADY_EXISTS	指定されたコミックはすでに存在しています
43	NO_CONV_ESTABLISHED	DDE 接続は失敗しました
44	TRANSIENT_STATE	処理が拒否されました（しばらくしてから再試行してください）
45	NO_LASTERROR_PROVIDED	外部 ncdde からの lastError 情報はありません
46	OUT_OF_STRINGHANDLES	よく似た文字列操作が多すぎます
47	WRITE_PROTECT	書き込みが許可されていません
48	OUT_OF_AMAPS	トピックのカウントが高すぎて ncdde.exe が戻されます ???Was kann der Anwender da tun ???
49	ADVISE_NOT_SUPPORTED	ホット／ウォームリンクは使用できません
4A	REQUEST_NOT_SUPPORTED	要求トランザクションはサポートされていません
4B	POKE_NOT_SUPPORTED	書き込みトランザクションはサポートされていません
4C	EXECUTE_NOT_SUPPORTED	実行トランザクションはサポートされていません
4D	DEVICE_NOT_PRESENT	ncdde は要求されたデバイスを検出しませんでした

4E	DATA_COUNT	索引の範囲がデータフィールドと等しくありません
FC	INTERNAL_ERROR	ncdde の内部障害です
FD	UNEXPECTED_EXCEPTION	予期しない例外が発生しました
FE	CAN_NOT_OPEN_ERROR	オープンできませんでした
FF	NO_ERRCODE	使用できるエラーコードがありません

11.7.5 general = 2 の詳細

02	xx	クラス	xx

表 11.19 general = 2 のエラークラス

クラス	ID	説明
01	WRITE	致命的でない ihi_write の戻り時にエラーが検出されました エラーコードは profibus 命令コードに対応します（以下を参照）
02	TIMEOUT	インタフェースが突然コマンドに応答しなくなりました エラーコードは profibus 命令コードに対応します（以下を参照）
03	RESPONSE_ERROR	この時点では予期していない応答を profibus インタフェースから受け取りました エラーコードは profibus インタフェースの応答コードに対応します
04	READ_FATAL	ihi_read の呼び出しに対して致命的なエラーを受け取りました ハードウェア／ソフトウエアの構成を調べる必要があります
05	WRITE_FATAL	ihi_write の呼び出しに対して致命的なエラーを受け取りました ハードウェア／ソフトウエアの構成を調べる必要があります

02	xx	クラス	コード

表 11.20 general = 2 のエラーコード

クラス	コード	ID	説明
01, 02	00	OPEN_REQ	オープンエラー
01, 02	01	SEND_CONN_REQ	初期化層 4 に接続しました
01, 02	06	SEND_EOM_DATA	データ書き込みエラー
01, 02	07	RECEIVE_DATA	データ読み取りエラー

01, 02	0C	CLOSE_REQ	クローズエラー
03	02	INVALID_REQ	内部障害
03	04	NO_RESOURCES	メモリが 1 MB を下回っています
03	06	UNKNOWN_REFERENCE	内部障害
03	0C	ILLEGAL_REQ	内部障害
03	0E	REM_ABORT	リモート端末が拒否されました
03	10	LOC_TIMEOUT	リモート端末が応答しません
03	12	UNKNOWN_CONN_CLASS	内部障害
03	14	DUP_REQ	内部障害
03	16	CONN_REJECT	リモート端末への接続が拒否されました
03	18	NEGOT_FAILED	端末は非互換です
03	1A	ILLEGAL_ADDRESS	アドレスパラメータが正しくありません
03	1C	NETWORK_ERROR	ネットワークエラー

11.7.6 general = 3 の詳細

general = 3 の場合，LastError のバイト 3 および 4 は NCU によって生成されます。

そのため，それらは NCU のソフトウェアバージョンによって決まります。

11.7.7 general = 5 の詳細

05	xx	xx	コード

表 11.21 general = 5 のエラーコード

番号	説明
01	ハードウェアエラー
03	オブジェクトへのアクセスは許可されていません
05	アドレスが無効です
06	データ・タイプがサポートされていません
07	データ・タイプが矛盾しています
0A	オブジェクトが存在しません

11.7.8 general = 7 の詳細 :ecol.

07	xx	01	コード

DCTL コントロール（general = 7）の場合，エラーコードは 1 です。

表 11.22 general = 7 のエラーコード :ecol.

番号	説明
01	タスクリストのオーバーロード : 現在 20 を超えるタスクが DCTL コントロールを使用しています
02	プロパティ LinkTopic で正しくない構文が検出されました
03	Windows のアトムテーブルを超過しています
04	サーバへの接続を確立できません
05	プロパティ LinkTopic の文字ストリングの最大数を超えています (最大 512 文字)
06	同じ項目が多すぎます。DCTL コントロールでは，同じ項目は 1 プロセスにつき 100 に制限されています
07	時間制限を超過しています
08	プロパティ LinkCmd の値が無効です
09	サーバがトランザクションを拒否しましたが，エラーコード LastError を戻しませんでした
0A	DDEML ライブラリの初期化に失敗しました
0C	プロパティ LinkTopic の値が無効です
0D	DDE クライアントトランザクションの呼び出しで，予期しない障害が発生しました
0F	リンクリストのオーバーフロー : このタスクには，20 を超えるさまざまな LinkTopics を使ったホットリンクがあります
10	サーバによって接続が中止されました
11	最後のエラーがセットされていません
12	コントロールの同期トランザクションの活動中に，呼び出し側がその同じ DCTL コントロールのトランザクション（読み取り，書き込み，実行または勧告）をパラメータ指定しようとしました。これは許可されません。DCTL コントロールは，前のトランザクションが終了するまで，次のトランザクションを実行することができません。

11.7.9 general = 8 の詳細

08	xx	**DOS からの整数**

DOS エラーメッセージ（general = 8）の場合，2 番目のバイトには意味はありません。

DOS によって戻されるコードの長さは 1 語で，3 番目および 4 番目のバイトに渡されます。

次の表は，最も頻繁に表示されるメッセージのリストです。

表 11.23 MS-DOS からのエラーメッセージ

番号	ID	説明

02	ENOENT	このファイルまたはディレクトリは見つかりません。指定したファイルまたはディレクトリは存在しないか，検出できません。このメッセージは，指定したファイルが存在していないか，パス名のコンポーネントが既存のディレクトリを指定していない場合に生じます。
07	E2BIG	引き数リストが長すぎます。DOS では，引き数リストは 128 バイトを超過するか，または環境情報に必要なスペースが 32K を超過します。
08	ENOEXEC	実行形式のエラーです。実行不可の，または無効な実行可能ファイル形式のファイルを実行しようとしました。
09	EBADF	ファイル番号が正しくありません。指定したファイルハンドルが有効なファイルハンドル値でないか，オープンファイルを参照しません。または，読み取り専用で開いたファイルまたはデバイスに書き込もうとしました（または，書き込み専用で開いたファイルまたはデバイスを読み取ろうとしました)。
0A	ECHILD	子プロセスはありません
0B	EAGAIN	これ以上プロセスはありません。プロセスのスロットがもうないか，十分なメモリがないか，または最大ネストレベルに到達したため，新しいプロセスの作成を試行しました。
0C	ENOMEM	コアが不十分です。試行したオペレータが使用できるメモリが不十分です。たとえば，このメッセージは，子プロセスを実行するのに必要なメモリを使用できないときや，_getcwd 呼び出しにおける割り振り要求を満たすことができないときに生じる場合があります。
0D	EACCES	アクセス権が拒否されました。このファイルのアクセス権の設定では，指定したアクセスは許可されません。このエラーは，ファイルの属性と互換性のないファイル（または，場合によってはディレクトリ）へのアクセスを試行したことを示します。たとえば，このエラーは，開いていないファイルから読み取ろうとしたり，既存の読み取り専用ファイルを書き込むために開こうとしたり，またはファイルではなくディレクトリを開こうとしたりすると生じます。MS-DOS バージョン 3.0 以降では，EACCES はロック違反または共用違反を表す場合もあります。このエラーは，ファイルまたはディレクトリの名前を変更しようとしたり，既存のディレクトリを削除しようとしたりしても生じる場合があります。
11	EEXIST	ファイルが存在しています。すでに存在しているファイルを作成しようとしました。
番号	ID	説明
		たとえば，オープン呼び出しでは _O_CREAT および _O_EXCLflags フラグが指定されますが，名前付きファイルはすでに存在してます。
12	EXDEV	デバイス相互のリンク。ファイルを（名前変更機能を使って）別のデバイスに移動しようとしました。
16	EINVAL	引き数が無効です。関数に対する引き数の 1 つに無効な値を指定しました。たとえば，（fseek を呼び出すことにより）ファイルポインタの位置を決める際，プログラム原点に指定する値は，ファイルの先頭よりも前です。
18	EMFILE	開いているファイルが多すぎます。これ以上利用できるファイルハンドルがないため，さらにファイルを開くことはできません。

1C	ENOSPC	デバイスにスペースが残されていません。書き込み用に使用できるスペースがデバイスにはありません（たとえば，ディスクが満杯です)。
21	EDOM	数学引き数
22	ERANGE	結果が大きすぎます。数学関数に対する引き数が大きすぎるため，結果の有効数字は部分的または全体的に消失が生じます。このエラーは，引き数が予期されているより大きい場合，他の関数でも生じる場合があります（たとえば，_getcwd 関数のパス名引き数が予期されているよりも大きい場合)。
24	EDEADLOCK	リソースのデッドロックが生じた可能性があります。数学関数に対する引き数が，その関数のドメインにありません。

11.7.10 general = 9 の詳細

ここで，バイト 3 および 4 には，DDEML からのエラーメッセージ含まれます。これは，Windows のエラーメッセージとは別のものです。これは，NCU および MMC との通信のエラーを表すのではなく，NCDDE サーバとアプリケーションとの通信が中断されていることを示します。このコードの完全な定義は，ddeml.h ファイルに記載されています。このファイルは，ほとんどすべての Windows コンパイラの構成部分です。

11.8 データ管理のエラーメッセージ

表 11.24 データ管理のエラーメッセージ

エラーコード	エラーメッセージのテキスト	意味
101	DOS ファイルの作成またはオープン時にエラーが生じました。	読み取り専用ディレクトリにファイルを作成しないでください
102	ファイルが存在しません	既存のファイルを開かないでください
103	ファイルが読み取り専用です	書き込み禁止ファイルを上書きしないでください，空でないディレクトリを削除しないでください
104	一致するノットが見つかりません	パスがデータスキームと一致しません
105	一致するデータタイプは見つかりません	ここではこのデータタイプは使用しないでください（たとえば，SPF ディレクトリで MPF は使用しないでください)。
106	ファイル／ディレクトリはすでに存在しています	既存のファイルに上書きしないでください
107	データ管理のパスが有効ではありません	無効なパスは使用しないでください
108	引き数エラーです	完全でないまたは正しくない引き数セットは呼び出さないでください
109	ブロックコピー中にエラーが発生しました	ハードディスクで読み取り／書き込みエラーが検出されました
110	無効なソースです	データ管理のパスが無効です
111	宛先のデータ・タイプが無効です	たとえば，SPF から MPF へのコピーなどはしないでください。
112	転送エラーです	NCU への接続のエラーメッセージを要約してください
113	読み取り許可がありません	適切な読み取り許可なしでデータにアクセスしないでください
114	書き込み許可がありません	適切な書き込み許可なしでデータにアクセスしないでください
115	実行許可がありません	適切な実行許可なしでデータにアクセスしないでください
116	表示許可がありません	適切な表示許可なしでデータにアクセスしないでください
117	削除許可がありません	適切な削除許可なしでデータにアクセスしないでください
118	一般アクセス許可がありません	NCU のアクセス違反のメッセージを要約してください
119	リソースエラーです	NCU のリソースエラーです
120	メモリ（malloc）エラーです	MMCremedy でメモリの割り振り中にエラーが発生しました。アプリケーションを閉じてください。
121	情報ファイルエラーです	追加情報（25 文字のファイル名，アクセス権，その他）を含むファイルでエラーが発生しました。 対処法 : 情報ファイルを削除して，DOS ファイルから復元してください。

122	リストバッファのオーバーフローエラー	64 KB の固定バッファ長を超過しました
123	ハードディスクが満杯です	ハードディスクの残量は，ジョブの実行を終了するには不十分です
124	コマンドが取り消されました	コマンドが CANCEL によって取り消されました
125	ディスクの位置が開いています	フロッピーディスクへのアクセス時に，ディスクドライブがオープンします
126	これ以上ファイルのタイプはありません	
200	DDE エラー	MMC 領域で DDE 通信中にエラーが発生しました

11.9 データ管理のディレクトリ構造

|_ROOT DIR	MMC-Daten
|_BD DIR	Anzeigemaschinendaten
| |_* TEA	Maschinendaten(TEA)
|_COM DIR	Kommentare
| |_* COM	Kommentar(COM)
|_CST DIR	Standard-Zyklen
| |_SC COM	Aufrufbeschr/System
| |_DPWP INI	DP-Initialisierung
| |_* AWB	Bildbeschreibung(AWB)
| |_* LST	Bildliste(LST)
| |_* COM	Kommentar(COM)
| |_* SPF	Unterprogramm(SPF)
|_CUS DIR	Anwender-Zyklen
| |_COV COM	Zyklenübersicht
| |_UC COM	Aufrufbeschr/Anwender
| |_DPWP INI	DP-Initialisierung
| |_* AWB	Bildbeschreibung(AWB)
| |_* LST	Bildliste(LST)
| |_* COM	Kommentar(COM)
| |_* SPF	Unterprogramm(SPF)
|_DEF DIR	Definitionen
| |_GUD4 DEF	NC-Anwenderdaten-4
| |_GUD5 DEF	NC-Anwenderdaten-5
| |_GUD6 DEF	NC-Anwenderdaten-6
| |_GUD7 DEF	NC-Anwenderdaten-7
| |_GUD8 DEF	NC-Anwenderdaten-8
| |_GUD9 DEF	NC-Anwenderdaten-9
| |_MGUD DEF	GlobaleDaten/Mherst
| |_MMAC DEF	Makros/Mherst
| |_SGUD DEF	GlobaleDaten/System
| |_SMAC DEF	Makros/System
| |_UGUD DEF	GlobaleDaten/Anwender

|_UMAC DEF	Makros/Anwender
|_DH DIR	Datenhaltung
|_DCF COM	Konfiguration
|_LOG COM	Logbuch/DH
|_TYP COM	Datentyp
|_HS DIR	HSA-Daten
|_HS-TEA ACC	Zugr/HSA-TEA
|_HS1 BOT	Bootdaten/HSA1
|_HS10 BOT	Bootdaten/HSA10
|_HS11 BOT	Bootdaten/HSA11
|_HS12 BOT	Bootdaten/HSA12
|_HS13 BOT	Bootdaten/HSA13
|_HS14 BOT	Bootdaten/HSA14
|_HS15 BOT	Bootdaten/HSA15
|_HS16 BOT	Bootdaten/HSA16
|_HS17 BOT	Bootdaten/HSA17
|_HS18 BOT	Bootdaten/HSA18
|_HS19 BOT	Bootdaten/HSA19
|_HS2 BOT	Bootdaten/HSA2
|_HS20 BOT	Bootdaten/HSA20
|_HS21 BOT	Bootdaten/HSA21
|_HS22 BOT	Bootdaten/HSA22
|_HS23 BOT	Bootdaten/HSA23
|_HS24 BOT	Bootdaten/HSA24
|_HS25 BOT	Bootdaten/HSA25
|_HS26 BOT	Bootdaten/HSA26
|_HS27 BOT	Bootdaten/HSA27
|_HS28 BOT	Bootdaten/HSA28
|_HS29 BOT	Bootdaten/HSA29
|_HS3 BOT	Bootdaten/HSA3
|_HS30 BOT	Bootdaten/HSA30
|_HS31 BOT	Bootdaten/HSA31
|_HS4 BOT	Bootdaten/HSA4
|_HS5 BOT	Bootdaten/HSA5

|_HS6 BOT	Bootdaten/HSA6
|_HS7 BOT	Bootdaten/HSA7
|_HS8 BOT	Bootdaten/HSA8
|_HS9 BOT	Bootdaten/HSA9
|_MPF DIR	Teileprogramme
|_* CEC	Durchhang/Winkligkeit(CEC)
|_* QEC	Quadrantenfehlerkomp(QEC)
|_* EEC	Meßsystemfehlerkomp(EEC)
|_DPWP INI	DP-Initialisierung
|_* MPF	Teileprogramm(MPF)
|_OEM DIR	OEM-Daten
|_* USD	Anwenderdaten(USD)
|_SPF DIR	Unterprogramme
|_DPWP INI	DP-Initialisierung
|_* SPF	Unterprogramm(SPF)
|_SYF DIR	System
|_* SYF	Systemdaten(SYF)
|_LOGBOOK SYF	Logbuch
|_VERSION SYF	Version
|_VS DIR	VSA-Daten
|_VS-TEA ACC	Zugr/VSA-TEA
|_HL1 BOT	Bootdaten/HLA1
|_HL10 BOT	Bootdaten/HLA10
|_HL11 BOT	Bootdaten/HLA11
|_HL12 BOT	Bootdaten/HLA12
|_HL13 BOT	Bootdaten/HLA13
|_HL14 BOT	Bootdaten/HLA14
|_HL15 BOT	Bootdaten/HLA15
|_HL16 BOT	Bootdaten/HLA16
|_HL17 BOT	Bootdaten/HLA17
|_HL18 BOT	Bootdaten/HLA18
|_HL19 BOT	Bootdaten/HLA19
|_HL2 BOT	Bootdaten/HLA2
|_HL20 BOT	Bootdaten/HLA20

||_HL21 BOT	Bootdaten/HLA21
||_HL22 BOT	Bootdaten/HLA22
||_HL23 BOT	Bootdaten/HLA23
||_HL24 BOT	Bootdaten/HLA24
||_HL25 BOT	Bootdaten/HLA25
||_HL26 BOT	Bootdaten/HLA26
||_HL27 BOT	Bootdaten/HLA27
||_HL28 BOT	Bootdaten/HLA28
||_HL29 BOT	Bootdaten/HLA29
||_HL3 BOT	Bootdaten/HLA3
||_HL30 BOT	Bootdaten/HLA30
||_HL31 BOT	Bootdaten/HLA31
||_HL4 BOT	Bootdaten/HLA4
||_HL5 BOT	Bootdaten/HLA5
||_HL6 BOT	Bootdaten/HLA6
||_HL7 BOT	Bootdaten/HLA7
||_HL8 BOT	Bootdaten/HLA8
||_HL9 BOT	Bootdaten/HLA9
||_SL1 BOT	Bootdaten/SLM1
||_SL10 BOT	Bootdaten/SLM10
||_SL11 BOT	Bootdaten/SLM11
||_SL12 BOT	Bootdaten/SLM12
||_SL13 BOT	Bootdaten/SLM13
||_SL14 BOT	Bootdaten/SLM14
||_SL15 BOT	Bootdaten/SLM15
||_SL16 BOT	Bootdaten/SLM16
||_SL17 BOT	Bootdaten/SLM17
||_SL18 BOT	Bootdaten/SLM18
||_SL19 BOT	Bootdaten/SLM19
||_SL2 BOT	Bootdaten/SLM2
||_SL20 BOT	Bootdaten/SLM20
||_SL21 BOT	Bootdaten/SLM21
||_SL22 BOT	Bootdaten/SLM22
||_SL23 BOT	Bootdaten/SLM23

|_SL24 BOT	Bootdaten/SLM24
|_SL25 BOT	Bootdaten/SLM25
|_SL26 BOT	Bootdaten/SLM26
|_SL27 BOT	Bootdaten/SLM27
|_SL28 BOT	Bootdaten/SLM28
|_SL29 BOT	Bootdaten/SLM29
|_SL3 BOT	Bootdaten/SLM3
|_SL30 BOT	Bootdaten/SLM30
|_SL31 BOT	Bootdaten/SLM31
|_SL4 BOT	Bootdaten/SLM4
|_SL5 BOT	Bootdaten/SLM5
|_SL6 BOT	Bootdaten/SLM6
|_SL7 BOT	Bootdaten/SLM7
|_SL8 BOT	Bootdaten/SLM8
|_SL9 BOT	Bootdaten/SLM9
|_VS1 BOT	Bootdaten/VSA1
|_VS10 BOT	Bootdaten/VSA10
|_VS11 BOT	Bootdaten/VSA11
|_VS12 BOT	Bootdaten/VSA12
|_VS13 BOT	Bootdaten/VSA13
|_VS14 BOT	Bootdaten/VSA14
|_VS15 BOT	Bootdaten/VSA15
|_VS16 BOT	Bootdaten/VSA16
|_VS17 BOT	Bootdaten/VSA17
|_VS18 BOT	Bootdaten/VSA18
|_VS19 BOT	Bootdaten/VSA19
|_VS2 BOT	Bootdaten/VSA2
|_VS20 BOT	Bootdaten/VSA20
|_VS21 BOT	Bootdaten/VSA21
|_VS22 BOT	Bootdaten/VSA22
|_VS23 BOT	Bootdaten/VSA23
|_VS24 BOT	Bootdaten/VSA24
|_VS25 BOT	Bootdaten/VSA25
|_VS26 BOT	Bootdaten/VSA26

|_VS27 BOT	Bootdaten/VSA27
|_VS28 BOT	Bootdaten/VSA28
|_VS29 BOT	Bootdaten/VSA29
|_VS3 BOT	Bootdaten/VSA3
|_VS30 BOT	Bootdaten/VSA30
|_VS31 BOT	Bootdaten/VSA31
|_VS4 BOT	Bootdaten/VSA4
|_VS5 BOT	Bootdaten/VSA5
|_VS6 BOT	Bootdaten/VSA6
|_VS7 BOT	Bootdaten/VSA7
|_VS8 BOT	Bootdaten/VSA8
|_VS9 BOT	Bootdaten/VSA9
|_WKS DIR	Werkstücke
|_* WPD	Werkstück(WPD)
|_* JOB J	obliste
|_* RPA	Rechenparameter(RPA)
|_* TMA	Magazindaten(TMA)
|_* PRO	Schutzbereiche
|_* CEC	Durchhang/Winkligkeit(CEC)
|_* UFR	Nullpunktversch/Frame(UFR)
|_* SPF	Unterprogramm(SPF)
|_* COM	Kommentar(COM)
|_* GUD	Kanalanwenderdaten(GUD)
|_* SEA	Settingdaten(SEA)
|_* TEA	Maschinendaten(TEA)
|_* TOA	Werkzeugkorrekturen(TOA)
|_* IKA	Kompensationsdaten(IKA)
|_* INI	Initialisierungsprogramm(INI)
|_DPWP INI	DP-Initialisierung
|_* DAT	Zeitberechnung(DAT)
|_* 041	Autoturn-Programm
|_* TOP	Werkzeugplan(TOP)
|_* TCM	Werkzeugplan-Unform.(TCM)
|_* MPF	Teileprogramm(MPF)

|_* MDN	NC-Daten-Sicherung
| |_GUD DIR	Anwenderdaten
| | |_COMPLETE_GUD INI	Anwenderdaten-Komplett
| | |_CH_GUD DIR	Kanal-Anwenderdaten
||||_CH_GUDINI	Kanal-Anwenderdaten-Komplett
||||_CH%c_GUDDIR	Anwenderdaten-Kanal
| | | |_CH%c_GD1 INI	Anwenderdaten-1-Kanal
| | | |_CH%c_GD2 INI	Anwenderdaten-2-Kanal
| | | |_CH%c_GD3 INI	Anwenderdaten-3-Kanal
| | | |_CH%c_GD4 INI	Anwenderdaten-4-Kanal
| | | |_CH%c_GD5 INI	Anwenderdaten-5-Kanal
| | | |_CH%c_GD6 INI	Anwenderdaten-6-Kanal
| | | |_CH%c_GD7 INI	Anwenderdaten-7-Kanal
| | | |_CH%c_GD8 INI	Anwenderdaten-8-Kanal
| | | |_CH%c_GD9 INI	Anwenderdaten-9-Kanal
| | | |_CH%c_GUD INI	Anwenderdaten-Komplett-Kanal
| | |_NC_GUD DIR	Globale-Anwenderdaten
| | |_NC_GD1 INI	Globale-Anwenderdaten-1
| | |_NC_GD2 INI	Globale-Anwenderdaten-2
| | |_NC_GD3 INI	Globale-Anwenderdaten-3
| | |_NC_GD4 INI	Globale-Anwenderdaten-4
| | |_NC_GD5 INI	Globale-Anwenderdaten-5
| | |_NC_GD6 INI	Globale-Anwenderdaten-6
| | |_NC_GD7 INI	Globale-Anwenderdaten-7
| | |_NC_GD8 INI	Globale-Anwenderdaten-8
| | |_NC_GD9 INI	Globale-Anwenderdaten-9
| | |_NC_GUD INI	Globale-Anwenderdaten-Kompl
| |_* INI	Initialisierungsprogramm(INI)
| |_NC_CEC INI	Durchhang/Winkligkeit-Kompl
| |_EEC DIR	Meßsystemfehlerkompensation
| | |_AX%a_EEC INI	Meßsystemfehlerkomp-Achse
| | |_AX_EEC INI	Meßsystemfehlerkomp-Komplett
| |_OPT DIR	Optionsdaten
| | |_AX_OPT INI	AX-Optionsdaten

| | |_CH_OPT INI CH-Optionsdaten

| | |_COMPLETE_OPT INI Optionsdaten-Komplett

| | |_NC_OPT INI NC-Optionsdaten

| |_PRO DIR Schutzbereiche

| | |_COMPLETE_PRO INI Schutzbereiche-Komplett

| | |_NC_PRO INI NC-Schutzbereiche

| | |_CH_PRO DIR Kanal-Schutzbereiche

| | |_CH%c_PRO INI Schutzbereiche-Kanal

| |_QEC DIR Quadrantenfehlerkompensation

| | |_AX%a_QEC INI Quadrantenfehlerkomp-Achse

| | |_AX_QEC INI Quadrantenfehlerkomp-Komplett

| |_RPA DIR R-Parameter

| | |_CH%c_RPA INI R-Parameter-Kanal

| | |_CH_RPA INI R-Parameter-Komplett

| |_SEA DIR Settingdaten

| | |_COMPLETE_SEA INI Settingdaten-Komplett

| | |_NC_SEA INI Allgemeine-Settingdaten

| | |_AX_SEA DIR Achs-Settingdaten

| | | |_AX%a_SEAINI Settingdaten-Achse

| | | |_AX_SEAINI Achs-Settingdaten-Komplett

| | |_CH_SEA DIR Kanal-Settingdaten

| | |_CH%c_SEA INI Settingdaten-Kanal

| | |_CH_SEA INI Kanal-Settingdaten-Komplett

| |_TEA DIR Maschinendaten

| | |_COMPLETE_TEA INI Maschinendaten-Komplett

| | |_NC_TEA INI NC-Maschinendaten

| | |_AX_TEA DIR Achs-Maschinendaten

| | | |_AX%a_TEAINI Maschinendaten-Achse

| | | |_AX_TEA INI Achs-Maschinendaten-Komplett

| | |_CH_TEA DIR Kanal-Maschinendaten

| | |_CH%c_TEA INI Maschinendaten-Kanal

| | |_CH_TEA INI Kanal-Maschinendaten-Komplett

| |_TO DIR Werkzeug-/Magazindaten

| | |_TO_INI INI Werkzeug-/Magazindaten-Kompl

| | |_TO_TMA DIR Magazindaten
||||_TO%t_TMA INI Magazindaten-Einheit
||||_TO_TMA INI Magazindaten-Komplett
| | |_TO_TOA DIR Werkzeugkorrekturen
| | |_TO%t_TOA INI Werkzeugkorrekturen-TO
| | |_TO_TOA INI Werkzeugkorrekturen-Komplett
| |_UFR DIR Nullpunktverschiebungen
| |_CH_UFR DIR Kanal-Nullpunktverschiebungen
| | |_CH%c_UFR INI Nullpunktverschiebungen-Kanal
| | |_CH_UFR INI Kanal-Nullpunktversch.-Komple
| |_COMPLETE_UFR INI Nullpunktverschiebungen-Kompl
| |_NC_UFR INI Allgemeine-Nullpunktverschieb
|_ARC DIR Archive
| |_* ARC Archiv(ARC)
|_* CLP Zwischenablage
|_DG DIR Diagnose
| |_* UMA Maschine
| | |_BITMAP DIR Bitmap/Grafiken
||||_*BMP Bitmap(BMP)
| | |_DAU DIR D/A-Umsetzer
||||_*DAC D/A-Konfiguration
| | |_FG DIR Funktions-Generator
||||_*FGC FG-Konfiguration
| | |_IPOTRC DIR IPO-Trace
||||_*COM Kommentar(COM)
| | |_KFT DIR Kreisformtest
||||_KFTDIADIR Kreisformtest-Diagramm
|||||_*SUD Setup-Diagramm(SUD)
||||_KFTPARDIR Kreisformtest-Parameter
| | | |_* SUP Setup-Parameter(SUP)
| | |_MCC DIR Stromregelkreis
||||_MCCGR1DIR MCC-Grafik1
|||||_*MC1 MCC-Grafik1-Daten
||||_MCCGR2DIR MCC-Grafik2

||||_*MC2	MCC-Grafik2-Daten
|||_MCCPARDIR	MCC-Parameter
| | |_* MCC	MCC-Parameter-Daten
| |_MDAX DIR	MaschDat/Achse
|||_*TEA	Maschinendaten(TEA)
| |_MDBT DIR	MaschDat/Bedientafel
|||_*TEA	Maschinendaten(TEA)
| |_MDCH DIR	MaschDat/Kanal
|||_*TEA	Maschinendaten(TEA)
| |_MDCOMP DIR	Datenvergleich
|||_*COM	Kommentar(COM)
| |_MDHSA DIR	MaschDat/HAS
|||_*TEA	Maschinendaten(TEA)
| |_MDNC DIR	MaschDat/NC
|||_*TEA	Maschinendaten(TEA)
| |_MDVSA DIR	MaschDat/VSA
|||_*TEA	Maschinendaten(TEA)
| |_MPC DIR	Lageregelkreis
|||_MPCGR1DIR	MPC-Grafik1
||||_*MP1	MPC-Grafik1-Daten
|||_MPCGR2DIR	MPC-Grafik2
||||_*MP2	MPC-Grafik2-Daten
|||_MPCPARDIR	MPC-Parameter
| | |_* MPC	MPC-Parameter-Daten
| |_MSC DIR	Drehzahlregelkreis
|||_MSCGR1DIR	MSC-Grafik1
||||_*MS1	MSC-Grafik1-Daten
|||_MSCGR2DIR	MSC-Grafik2
||||_*MS2	MSC-Grafik2-Daten
|||_MSCPARDIR	MSC-Parameter
| | |_* MSC	MSC-Parameter-Daten
| |_SVTRC DIR	Servo-Trace
| |_SVTGR1 DIR	SVT-Grafik1
| | |_* ST1	SVT-Grafik1(ST1)

| | |_SVTGR2 DIR SVT-Grafik2
| | | |_* ST2 SVT-Grafik2(ST2)
| | |_SVTPAR DIR SVT-Parameter
| | |_* SVT SVT-Parameter(SVT)
| |_BITMAP DIR Bitmap/Grafiken
| | |_* BMP Bitmap(BMP)
| |_DAU DIR D/A-Umsetzer
| | |_* DAC D/A-Konfiguration
| |_FG DIR unktions-Generator
| | |_* FGC FG-Konfiguration
| |_INIT DIR MMC-Initialisierung
| | |_* INI Initialisierungsprogramm(INI)
| | |_* ZIP MMC-Konfiguration
| |_IPOTRC DIR IPO-Trace
| | |_* COM Kommentar(COM)
| |_KFT DIR Kreisformtest
| | |_KFTDIA DIR Kreisformtest-Diagramm
| | | |_*SUD Setup-Diagramm(SUD)
| | |_KFTPAR DIR Kreisformtest-Parameter
| | |_* SUP Setup-Parameter(SUP)
| |_KLB DIR Konfig.Listenbilder
| | |_* KLB Konfig.Listenbild
| |_LOGFILES DIR Protokolldateien
| | |_* COM Kommentar(COM)
| |_MCC DIR Stromregelkreis
| | |_MCCGR1 DIR MCC-Grafik1
| | | |_*MC1 MCC-Grafik1-Daten
| | |_MCCGR2 DIR MCC-Grafik2
| | | |_*MC2 MCC-Grafik2-Daten
| | |_MCCPAR DIR MCC-Parameter
| | |_* MCC MCC-Parameter-Daten
| |_MDAX DIR MaschDat/Achse
| | |_* TEA Maschinendaten(TEA)
| |_MDBT DIR MaschDat/Bedientafel

| | |_* TEA Maschinendaten(TEA)
| |_MDCH DIR MaschDat/Kanal
| | |_* TEA Maschinendaten(TEA)
| |_MDCOMP DIR Datenvergleich
| | |_* COM Kommentar(COM)
| |_MDDRV DIR MaschDat/Antriebe-Komplett
| | |_* TEA Maschinendaten(TEA)
| |_MDHSA DIR MaschDat/HAS
| | |_* TEA Maschinendaten(TEA)
| |_MDNC DIR MaschDat/NC
| | |_* TEA Maschinendaten(TEA)
| |_MDVSA DIR MaschDat/VSA
| | |_* TEA Maschinendaten(TEA)
| |_MPC DIR Lageregelkreis
| | |_MPCGR1 DIR MPC-Grafik1
| | | |_*MP1 MPC-Grafik1-Daten
| | |_MPCGR2 DIR MPC-Grafik2
| | | |_*MP2 MPC-Grafik2-Daten
| | |_MPCPAR DIR MPC-Parameter
| | |_* MPC MPC-Parameter-Daten
| |_MSC DIR Drehzahlregelkreis
| | |_MSCGR1 DIR MSC-Grafik1
| | | |_*MS1 MSC-Grafik1-Daten
| | |_MSCGR2 DIR MSC-Grafik2
| | | |_*MS2 MSC-Grafik2-Daten
| | |_MSCPAR DIR MSC-Parameter
| | |_* MSC MSC-Parameter-Daten
| |_PLC DIR PLC-Daten
| | |_* COM Kommentar(COM)
| |_PLCUD DIR PLC-Operanden-Masken
| | |_* PLC Operanden-Maske(PLC)
| |_SVTRC DIR Servo-Trace
| |_SVTGR1 DIR SVT-Grafik1
| | |_* ST1 SVT-Grafik1(ST1)

| | |_SVTGR2 DIR SVT-Grafik2
| | | |_* ST2 SVT-Grafik2(ST2)
| | |_SVTPAR DIR SVT-Parameter
| | |_* SVT SVT-Parameter(SVT)
| |_DP DIR Dialog-Programmierung
| | |_AWB DIR DP-Anwenderbilder
| | | |_* COM Kommentar(COM)
| | | |_* LST Bildliste(LST)
| | | |_* AWB Bildbeschreibung(AWB)
| | |_DPT DIR DP-Werkzeuge
| | | |_* BMP Bitmap(BMP)
| | |_GPMAC DIR GP-Makros
| | | |_MACM DIR GP-Fräsmakros
| | | | |_*DSC GP-Makrobeschr(DSC)
| | | | |_*MAC GP-Makrocode(MAC)
| | | |_MACT DIR GP-Drehmakros
| | | |_* DSC GP-Makrobeschr(DSC)
| | | |_* MAC GP-Makrocode(MAC)
| | |_HLP DIR DP-Hilfe
| | | |_* BMP Bitmap(BMP)
| | | |_* COM Kommentar(COM)
| | |_INF DIR DP-Basisinfo
| | | |_* BMP Bitmap(BMP)
| | | |_* COM Kommentar(COM)
| | |_SIM DIR Simulationsdaten
| | | |_GUD4 DEF NC-Anwenderdaten-4
| | | |_GUD5 DEF NC-Anwenderdaten-5
| | | |_GUD6 DEF NC-Anwenderdaten-6
| | | |_GUD7 DEF NC-Anwenderdaten-7
| | | |_GUD8 DEF NC-Anwenderdaten-8
| | | |_GUD9 DEF NC-Anwenderdaten-9
| | | |_MGUD DEF GlobaleDaten/Mherst
| | | |_MMAC DEF Makros/Mherst
| | | |_SGUD DEF GlobaleDaten/System

| | |_SMAC DEF Makros/System
| | |_UGUD DEF GlobaleDaten/Anwender
| | |_UMAC DEF Makros/Anwender
| | |_* INI Initialisierungsprogramm(INI)
| | |_* COM Kommentar(COM)
| | |_CST DIR Standard-Zyklen
|||| _*SPF Unterprogramm(SPF)
| | |_CUS DIR Anwender-Zyklen
| | |_* SPF Unterprogramm(SPF)
| |_TS DIR Technologie-Speicher
| |_* LDB TS-Listenstruktur(LDB)
| |_* MDB TS-Daten(MDB)
|_IBN DIR Inbetriebnahme
| |_DAC BIN DAU-Parametrierung
| |_* TRC Tracedatei(TRC)
|_MB DIR MBDDE-Alarmtexte
| |_* COM Kommentar(COM)
|_NC_CARD DIR NC-Card
| |_ARC DIR Archive
| |_* ARC Archiv(ARC)
|_PDA DIR Bearbeitungsfolge
| |_* WPP Werkstückplan(WPP)
|_TEMPL DIR Templates
| |_MANUF DIR Hersteller
| | |_* MPF Teileprogramm(MPF)
| | |_* SPF Unterprogramm(SPF)
| | |_* JOB Jobliste
| |_NEWFILES DIR Vorlagen
| | |_* MPF Teileprogramm(MPF)
| | |_* SPF Unterprogramm(SPF)
| | |_* JOB Jobliste
| | |_* TCM Werkzeugplan-Unform.(TCM)
| | |_* TOP Werkzeugplan(TOP)
| | |_* 041 Autoturn-Programm

| | |_* DAT Zeitberechnung(DAT)
| | |_* INI Initialisierungsprogramm(INI)
| | |_* IKA Kompensationsdaten(IKA)
| | |_* TOA Werkzeugkorrekturen(TOA)
| | |_* TEA Maschinendaten(TEA)
| | |_* SEA Settingdaten(SEA)
| | |_* GUD Kanalanwenderdaten(GUD)
| | |_* COM Kommentar(COM)
| | |_* UFR Nullpunktversch/Frame(UFR)
| | |_* CEC Durchhang/Winkligkeit(CEC)
| | |_* PRO Schutzbereiche
| | |_* TMA Magazindaten(TMA)
| | |_* RPA Rechenparameter(RPA)
| |_Siemens DIR Siemens
| | |_* MPF Teileprogramm(MPF)
| | |_* SPF Unterprogramm(SPF)
| | |_* JOB Jobliste
| |_USER DIR Anwender
| |_* MPF Teileprogramm(MPF)
| |_* SPF Unterprogramm(SPF)
| |_* JOB Jobliste
|_WZV DIR Werkzeugverwaltung
| |_MCFG DIR Magazin-Konfiguration
| | |_* INI Initialisierungsprogramm(INI)
| |_WCFG DIR WZV-Konfiguration
| | |_* CTC Konvert.Vorschrift(CTC)
| | |_* WMF Metafile(WMF)
| |_WDAT DIR WZV-Daten
| |_* LDB TS-Listenstruktur(LDB)
| |_* MDB TS-Daten(MDB)
|__NC_ACT DIR NC-Aktive-Daten

11.10 ソフトウェア情報

11.10.1 ファイル拡張子

表 11.25 ファイル拡張子の概要

拡張子	ファイルの内容と意味
$$$	一時ファイル
.000	インストールおよびオーバーレーファイル
.041	AUTOTURN プログラム
.ACC	NCU データ用のアクセス記述
.ANN	Windows ヘルプファイルに対する注
.APP	ユーザインタフェース GEM のアプリケーションファイル
.ARC	ARC で圧縮したファイル
.ARJ	ARJ で圧縮したファイル
.ASM	プログラミング言語アセンブラで作成されたソースコードファイル
.ASP	非同期サブプログラム
.BAK	ファイルのバックアップコピー
.BAT	バッチファイル（スタックファイル）
.BIN	バイナリファイル
.BMK	Windows ヘルプファイルのブックマーク
.BOT	ブートデータ VSA/HSA-611D
.BSP	
.C	プログラミング言語 C で作成されたソースコードファイル
.CAT	dBase カタログファイル
.CFG	構成ファイル
.CHK	CHKDSK で復元されたファイル
.COM	実行可能プログラム，コメントファイル
.CPI	文字テーブルを持つファイル
.CYC	サイクルプログラム
.DAT	*.DAT 時間計算 (DAT) = AUTOTURN 用データ

.DAT	AUTOTURN 用データ（時間計算）
.DBF	dBase データファイル
.DIR	ディレクトリ
.DLL	Windows アプリケーションのファイル
.DOC	テストファイル（ワードプロセッシングまたは ASCII）
.EMN	
.EXE	実行可能プログラム
.FPU	カタログ MMC2 にあるファイル
.FW?	フレームワークファイル
.GEM	グラフィカルユーザインタフェース GEM のファイル
.GUD	グローバル／チャネル固有ユーザデータ
.HLP	ヘルプファイル
.ICO	
.IKA	補間された補正
.INI	構成ファイル（通常 Windows アプリケーション）
.INX	索引ファイル（たとえばデータベース用）
.LHA	LHARC で圧縮したファイル
.LIB	ライブラリファイル（プログラム言語用）
拡張子	ファイルの内容と意味
.LUD	ローカル／プログラム固有ユーザデータ
.LZH	LHA で圧縮したファイル
.MPF	パートプログラム
.NDX	索引ファイル（たとえばデータベース用）
.OBJ	オブジェクトファイル（プログラム言語）
.OPT	オプション
.ORF	ProFilAct II で復元されたファイル
.OVL	オーバーレーファイル
.PAS	プログラム言語 PASCAL のソースコードファイル
.PIF	構成ファイル（Windows）
.PRG	アプリケーションプログラミングファイル（dBase など）

.REC	RECOVER で復元されたファイル
.REF	相互参照ファイル
.SEA	NC 設定データ
.SIK	ファイルのバックアップコピー（.BAK など）
.SPF	サブプログラム
.SWP	スワップファイル，たとえば仮想作業メモリー用
.SYF	システムファイル
.SYS	プログラム（ドライバ，CONFIG.SYS からの呼び出し）
.TEA	NC マシンデータ
.TMP	一時ファイル
.TOA	ツールデータ
.TXT	テキストファイル（ワードプロセッシングまたは ASCII）
.UFR	原点シフト／フレーム
.WKS	作業データファイル（.WDB および .WPS と同じ）
.WPD	ワークディレクトリ
.ZIP	PKZIP で圧縮したファイル

11.10.2 アラーム番号

表 11.26 アラーム番号範囲のリスト

アラーム番号	アラームエリア	アラームテキストファイル
	NCK アラーム	
000 000 ～ 009 999	一般アラーム	ALN_xx.COM
010 000 ～ 019 999	チャネルアラーム	ALN_xx.COM
020 000 ～ 029 999	軸スピンドルアラーム	ALN_xx.COM
030 000 ～ 039 999	機能アラーム	
040 000 ～ 059 999	予約済み	
060 000 ～ 062 999	サイクルアラーム	SIEMENS ALZ_xx.COM
063 000 ～ 064 999	予約済み	
065 000 ～ 067 999	サイクルアラームユーザ	

068 000 ～ 069 999	予約済み	
070 000 ～ 079 999	コンパイルサイクル開発者および OEM	ALN_xx.COM,ALC_xx.COM
080 000 ～ 099 999	予約済み	
	MMC アラーム／メッセージ	
100 000 ～ 109 999	MMC 100	
100 000 ～ 100 999	基本システム	
101 000 ～ 101 999	診断	
102 000 ～ 102 999	サービス	
103 000 ～ 103 999	マシン	
104 000 ～ 104 999	パラメータ	
105 000 ～ 105 999	プログラミング	
106 000 ～ 106 999	予約済み	
107 000 ～ 107 999	OEM	
108 000 ～ 109 999	予約済み	
110 000 ～ 119 999	MMC	
101120 000 ～ 129 999	PCU50	ALM_xx.COM
130 000 ～ 139 999	OEM	
140 000 ～ 199 999	予約済み	
（200 000 ～ 299 999	MCU アラーム）	
300 000 ～ 399 999	ドライブアラーム	ALN_xx.COM
	PLC アラーム／メッセージ	
400 000 ～ 499 999	一般アラーム	ALP_xx.COM
500 000 ～ 599 999	チャネルアラーム	
600 000 ～ 699 999	軸スピンドルアラーム	
700 000 ～ 799 999	ユーザエリア	
800 000 ～ 899 999	シーケンス／グラフ	ALP_xx.COM

810 000 ～ 810 009	PLC でのシステムエラーメッセージ	ALP_xx.COM
900 000 ～ 999 999	予約済み	

11.10.3 言語の略称

言語は，言語 DLL の名前およびアラームテキストファイルの名前から，2 文字の略称によって見分けることができます。

表 11.27 言語略称の概説

略称	言語
CH	中国語（中華人民共和国）
FR	フランス語
GB	英語
GR	ドイツ語
IT	イタリア語
KO	韓国語
SP	スペイン語
TW	中国語（台湾）

これらの略称は，LanguageList の下のデータファイル MMC.INI の［LANGUAGE］セクションにあります。

11.10.4 ドライバ

表 11.28 ドライバのリスト

名前	機能
ディレクトリ L:¥MMC2¥DRIVERS	
KBD.DRV	
SCANTAB.EXE	END キーを AUTOEXEC.BAT から呼び出される TAB にリセットする
VKD_MMC2.386	WINDOWS ドライバ VKD.386 を取り替えたもの。垂直 SK バー，ETC キー，および MACHINE キーを使用。シフトキーを保管。説明は，第 7 章，SYSTEM.INI の項目，セクション［386Enh］を参照。

VMMC2D.386	モノクロームオペレータパネルのコントラストを設定するための，DOS アプリケーションのエリア切り替え
ディレクトリ L:¥MMC2¥DRV.ID	
S7CFGPGX.DAT	MPI インタフェースのパラメータ
S7MONPGX.EXE	
S7MPIPGX.EXE	
SIN_SERV.EXE	

vkd_mmc2.386 ドライバの特殊キーコード

このドライバだけを使用する場合，シフト +F1 からシフト +F8 にはキーコードを提供せずに次の表にしたがって垂直ソフトキーと他のキーをコード化することに注意してください。

このコード範囲は，WINDOWS では使用しないでください。

表 11.29 ドライバ vkd_mmc2.386 の特殊キーコード

キー	スキャンコード	コード
垂直ソフトキー 1	5E	0xE0
垂直ソフトキー 2	5F	0xE1
垂直ソフトキー 3	62	0xE2
垂直ソフトキー 4	63	0xE3
垂直ソフトキー 5	64	0xE4
垂直ソフトキー 6	65	0xE5
垂直ソフトキー 7	66	0xE6
垂直ソフトキー 8	67	0xE7
ETC	69	0xE8
MACHINE	6A	0xE9
開き括弧（		0xEA
閉じ括弧）		0xEB

注 : 垂直ソフトキー 1（5E）のスキャンコードと垂直ソフトキー 2（5F）のスキャンコードは，通常は DOS の下でインプリメントされるので，キーを押す（コードを作成する）ための値（それぞれ値 60 と 61）とは異なります。
たとえば，垂直ソフトキー 1 に移動する（コードを中断する）コードは E0 に対応しており，接頭部コードを使うと（たとえば ALT キーを 2 回押す），それを押したとき（作成時）に E1 を戻すため，このコードは 5E と 5F に置き換えられています。

キーボードドライバを使った OP 031 のスキャンコードの修正

いくつかのキー（VSK0-7，M キー，ETC キー，' (', ') '，SingleQuote）では，OP031 がスキャンコードを生成します（PC エリアで予約済み）。これらのコードは標準キーコードではなく，今後の OP では生成されない可能性があるため，できるだけ早く除去する必要があります。P5 に含まれるキーボードドライバには，スキャンコードマッピング機能が装備されており，この機能によって OP031 が生成する特殊スキャンコードを再定義できます。

P5 では，以下のような非互換性が起こることがあります。

P5.1 までは，OP031 が次のコードを作成しました。

表 11.30 SW バージョン 5.1 までのコード

キー	スキャンコード	VirtKeyk コード
VSK0:	0x5E	0xE0
VSK1:	0x5F	0xE1
VSK2:	0x62	0xE2
VSK3:	0x63	0xE3
VSK4:	0x64	0xE4
VSK5:	0x65	0xE5
VSK6:	0x66	0xE6
VSK7:	0x67	0xE7
SingleQuote:	0x68	0xBF（PC など）
ETC:	0x69	0xE8
MACHINE:	0x6A	0xE9
(	0x6B	0xEA
)	0x6C	0xEB

P5.1UPD から，キーボードドライバは次のマッピングを提供します。

表 11.31 SW バージョン 5.1UPD からのコード

キー	スキャンコード	VirtKey コード

VSK0:	シフト -F1 と同様	シフト -F1 と同様
VSK1:	シフト -F2 と同様	シフト -F2 と同様
VSK2:	シフト -F3 と同様	シフト -F3 と同様
VSK3:	シフト -F4 と同様	シフト -F4 と同様
VSK4:	シフト -F5 と同様	シフト -F5 と同様
VSK5:	シフト -F6 と同様	シフト -F6 と同様
VSK6:	シフト F7 と同様	シフト F7 と同様
VSK7:	シフト -F8 と同様	シフト -F8 と同様
SingleQuote:	SingleQuote と同様	SingleQuote と同様
ETC:	シフト -F9 と同様	シフト -F9 と同様
MACHINE:	シフト -F10 と同様	シフト -F10 と同様
(	（と同様	（と同様
)	）と同様	）と同様

マッピング時には，ファイル SYSTEM.INI に含まれる表を使用します。VirtKey コードがあれば，一連の VirtKey コード（P5.1UPD から）によってその場で置き換えられます。P5.1 用の SYSTEM.INI は，上記のマッピングを示す表とともに提供されます。したがって，P5.1UPD のキーボードドライバは，PC 互換のスキャンコードだけを生成します。PCU50 で，「古い」OP031 コードだけを処理する OEM アプリケーションがインストールされると，SYSTEM.INI ではマッピング機能が使用不可になることがあります。これは，PCU50 アプリケーションには影響しません。

いくつかのアプリケーションがインストールされているときに，それらがキーコード要件を満たしていないと（つまり，たとえばある OEM アプリケーションが「古い」コードだけを処理し，別のアプリケーションが「新しい」コードだけを処理する場合など），問題が生じます。しかし，このケースは今のところ発生しません。なぜなら，これまでそのようなアプリケーションは PCU50-SW リリース <P5.1UPD> では実行できなかったので，「新しい」キーコードにしか対応しない OEM アプリケーションは存在しなかったからです。

どちらのキーコード（古いものと新しいもの）にも対応するアプリケーションは，すべての SW リリースで実行できるので，問題はありません。OP031 コードだけを処理するアプリケーションには問題が発生する可能性があるため，以前のスキャンコードでしか作業できません。

MF2 キーボードを使った OP031 のシミュレート

表 11.32 VB および AL を持つエリアアプリケーションの機能キー

機能キー	意味

F1 ～ F8	水平ソフトキーをアクティブにする
F9	RECALL キー
F10	エリア切り替えキー
F11	チャネル切り替えキー
F12	情報キー
Shift+F1 ～ Shift+F8	垂直ソフトキーをアクティブにする
Shift+F9	ETC キー
Shift+F10	マシンエリアキー
Shift+F11	現在未使用
Shift+F12	現在未使用
Esc キー	アラーム認識
Home キー	ウィンドウ切り替えキー

11.10.5 サポートされる言語

表 11.33 言語およびそれが適用される ANSI 表／コードページ

言語	省略形	コードページ（DOS）	ANSI 表（Windows）
ドイツ語	GR	850	1252
英語	UK	850	1252
スペイン語	SP	850	1252
イタリア語	IT	850	1252
フランス語	FR	850	1252
中国語（簡体字）	CH	936	-
中国語（繁体字）	TW	950	-
韓国語	KO	949	-
日本語	JA	932	-
スウェーデン語	SW	850	1252
ハンガリー語	HU	852	1250
ポルトガル語	PO	850	1252
チェコ語	CZ	852	1250

トルコ語	TR	857	1254
ロシア語	RU	866	1251
ポーランド語	PL	852	1250
オランダ語	NL	850	1252
フィンランド語	FI	850	1252

11.10.6 ANSI 表とフォントの割り当て

表 11.34 ANSI 表 1250（中央ヨーロッパ）

フォント	ファイル	文字
Arial	Cearial.ttf	Arial CE (True Type)
Arial bold	Cearіabd.ttf	Arial CE Bold (True Type)
Arial italic	Cearіali.ttf	Arial CE Italic (True Type)
Arial bold italic	Caeriabi.ttf	Arial CE Bold Italic (True Type)

表 11.35 Ansi 表 1251（キリル文字）

フォント	ファイル	文字
Arial	Aricyr.ttf	Arial Cyr (True Type)
Arial bold	Aricyb.ttf	Arial Cyr Bold (True Type)
Arial italic	Aricyri.ttf Arial	Cyr Italic (True Type)
Arial bold italic	Aricyrbi.ttf	Arial Cyr Bold Italic (True Type)

表 11.36 ANSI 表 1252（西ヨーロッパ）

フォント	ファイル	文字
Arial	Windows 標準	Arial (True Type)
Arial bold	Windows 標準	Arial Bold (True Type)
Arial italic	Windows 標準	Arial Italic (True Type)
Arial bold italic	Windows 標準	Arial Bold Italic (True Type)

これらの言語を使用するには，それぞれの言語を Windows にインストールする必要があります。

11.11 ハードウェア情報

11.11.1 メモリアドレス

表 11.37 メモリアドレス範囲の概要

種目	範囲（16 進）
ブートベクトルシステム BIOS（リセット後）	FFFF000:FFF0FFFF000:0000
予約済み	200000:0000
DRAM	40000:0000
DRAM	10000:0000
ブートベクトルシステム BIOS	F000:FFF0F000:0000
PCMCIA ウィンドウ範囲／ EMS	E000:0000
アダプタ RAM/ROM（PCMCIA/LAN/SCSI）	D000:0000
アダプタ RAM/ROM（PCMCIA/LAN/SCSI）	CC00:0800
MPI/AMPlus-L 範囲	CC00:0000
アダプタ RAM/ROM（PCMCIA/LAN/SCSI）	C800:0000
VGA BIOS	C000:0000
VGA DRAM	A000:0000
DRAM	0050:0000
BIOS 変数	0040:0000
ベクトルテーブル	0000:0000

11.11.2 割り込み要求

次の表は，割り込み要求（IR）および ISA アダプタまたは PC カードアダプタに対して指定できる割り当てを示します。最高度の優先順位は 0 です。IRQ 2 を要求するアダプタはそれぞれ IRQ 9 に転送されます。

表 11.38 MMC の割り込み要求

優先順位	割り込みコントローラ 1	割り込みコントローラ 2	IRQ	オプションの ISA アダプタ	オプションの PC カードアダプタ
0	タイマ		0	なし	なし

1	キーボードコントローラ		1	あり	なし
	割り込みコントローラ 2		2	-	-
2		リアルタイムクロック	8	なし	なし
3		自由（グラフィック）	9	あり	あり
4		MPI（K バス）	10	あり	あり
5		自由	11	あり	あり
6		自由（COM 3/4）	12	あり	あり
7		数値計算コプロセッサ	13	なし	なし
8		ハードディスク	14	あり	あり
9		自由	15	あり	あり
10	シリアルインタフェース 2（COM 2）	3	あり	あり	
11	シリアルインタフェース 1（COM 1）	4	あり	あり	
12	自由（パラレルインタフェース 2，LAN）	5	あり	あり	
13	フロッピードライブ	6	あり	なし	
14	パラレルインタフェース 1（LPT 1）	7	あり	あり	

11.11.3 入出力アドレス MMC

入出力アドレス 000 ～ 0FF はメインボード用に予約済みであり，ポート 100 ～ 3FF は拡張カードで使用可能です。

表 11.39 MMC の入出力アドレスの概要

種目	アドレス（16 進）
DMA コントローラ #1	000 ～ 01F
割り込みコントローラ	020 ～ 03F
タイマ	040 ～ 05F
キーボードコントローラ	060 ～ 06F
リアルタイムクロック，CMOS メモリ，NMI マスク	070 ～ 07F

メーカ診断チェックポイント（POST コード）	080
DMA ページレジスタ	080 ～ 09F
割り込みコントローラ #2	0A0 ～ 0BF
DMA コントローラ #2	0C0 ～ 0DF
演算コプロセッサ	0F0 ～ 0FF
ハードディスク（セカンダリ）	170 ～ 177
ハードディスク	1F0 ～ 1F7
ゲーム入出力（ジョイスティック A/D ポート）	200 ～ 207
サウンドカード	220 ～ 257
パラレルプリンタ #2	278 ～ 27F
EGA #2	2C0 ～ 2DF
非同期アダプタポート #2	2F8 ～ 2FF
プロトタイプカード	300 ～ 31F
例 : コンピュータリンクボード DF 15（COM 3）	338 ～ 33F
LAN カード	360 ～ 36F
パラレルプリンタ #1	378 ～ 37F
2 進同期ポート #2（SDLC）	380 ～ 38F
2 進同期ポート #1	3A0 ～ 3AF
Video Graphics Array（VGA）またはその代替品 モノクロディスプレイアダプタ／プリンタ #1 または代替品 拡張グラフィックスアダプタ（EGA）#1 または代替品 カラーグラフィックスアダプタ（CGA）および EGA	3B0 ～ 3DF 3B0 ～ 3BF 3C0 ～ 3CF 3D0 ～ 3DF
PCMCIA コントローラ	3E0 ～ 3E1
ディスクコントローラ	3F0 ～ 3F7
非同期アダプタポート #1	3F8 ～ 3FF

11.11.4 PCU50 のハードウェアの詳細

グラフィックチップ

タイプ 65550 のチップは，SVGA グラフィック用チップセットとして使用されます。

バス仕様

PCI/ISA ボックスの ISA および PCI バスは，以下の仕様に従います。:

PCI LOCAL BUS Specification Revision 2.1 (PCI Special InterestGroup 1.6.95)

(E)ISA Specification Version 3.2 BCPR Service, Inc.© 1989-1992

12　アプリケーション

OEM パッケージ MMC のデリバリボリュームにはサンプルがいくつか含まれており，これを利用すると，オペレータのインタフェースの設計についてすぐに理解し，840DI の MMC 構成要素の機能を拡張することができます。

この章では，提供されているサンプルについて簡単に説明しています。

前提条件

この章に載せられているサンプルを使って作業する場合，Visual Basic についての基本的な知識が必要です。

後に続く短い説明は，プログラム文書の一部です。これは，SRC ディレクトリにある該当するサンプル OEMBSPn（n はサンプルの順序番号）の OEMBSPn.WRI ファイルに示されています。

アプリケーションのパターン

サンプル OEMBSP0 には，アプリケーションのパターンが示されています。

このパターンを，独自の MMC-OEM オブジェクトの枠組みとして使うことができます。

VISUAL BASIC を使って，このプロジェクトに関係するファイルで作業する前に，

それらのファイルをコピーし，リネームするようにします。

12.1 シーケンス制御／アプリケーションの転換

目的

サンプル OEMBSP1 は基本サンプルで，MMC-OEM トピックを紹介し，シーケンス制御の使い方を例示しています。

さらに，840DI 開発環境に VISUAL BASIC が組み込まれる場合に，VISUAL BASIC でアプリケーションを作成するときに考慮する必要のあることも説明されています。

独自のシーケンス制御を実現する方法や，ソフトキーテキストを変更する方法について例示しています。最後に，独自のアプリケーションを Regie に組み込む方法も説明されます。

12.2 NCDDE サーバ

目的

サンプル OEMBSP2 は最初のサンプルに基づいており，NCDDE サーバの基本機能を扱っています。

サンプルの目的は，次のとおりです。

- DDE の基本を紹介する
- DDE サーバを初期設定する
- データ記述ファイルを構造化／処理する
- NC データにアクセスし，独自の VISUAL BASIC アプリケーションを表示する
- DDE LinkExecute ジョブを送る父 pートプログラムを MMC から NC へコピーする
- 実行するパートプログラムを選ぶ

条件

次の点に精通していなければなりません。

- VISUAL BASIC
- VISUAL BASIC を使った 840DI に適するアプリケーションの作成
- 独自のシーケンス制御の実現
- 言語 DLL の開発
- Regie へのアプリケーションの組み込み

12.3 アラームサーバ

目的

サンプル OEMBSP3 では，メッセージモジュール MBDDE で作業する方法が示されています。DDE メカニズムを用いて，アラームサーバはアラーム処理のサービスを提供します。

アラームサーバは，Visual Basic の下で直接に使うか，またはシーケンス制御によって使うことができます。

VB の下で直接アラームサーバを使う

"oembsp3\alar_dde" ディレクトリに，"mbddetst" というプログラム例があります。

このプログラムを使うと，Visual Basic の下でアラームサーバ "mbdde.exe" を直接に指定できます。

該当するラベルをクリックすると，コマンドがアラームサーバに送られるか，ラベルに応じたリンクが確立されます。

シーケンス制御によってアラームサーバを使う

以下のサンプルが実現します。

シーケンス制御のソフトキーおよびダイアログボックスを使うと，該当するコマンドがアラームサーバへ送られ，機能が呼び出されます。

データは，リンクメカニズムによって，該当する表示ボックスにコピーされます。

条件

VISUAL BASIC，VISUAL C++（DLL 変更用）の使い方についての基本的な知識が必要です。

Visual Basic の下でサンプル OEMBSP3 を実行する前に，以下のものを起動する必要があります。

- NC-DDE サーバ
- "mbdde.exe"（アラームサーバ，メッセージモジュール）
- "wtrace.dll"
- "regie.dll"

12.4 サンプル Regie/OEMFRAME

目的

サンプル OEMBSP4 には，Regie の使い方と，標準の WINDOWS アプリケーションを組み込む方法が示されています。

'OEM-Frame' の機能を使い，Regie にどのような WINDOWS アプリケーションでも組み込むことができます。ここでは，領域アプリケーション（ここではシーケンス制御）を使い，2 つの WINDOWS プログラム CLOCK および WRITE（どちらも標準の WINDOWS 配布ボリュームの一部）を Regie に組み込みます。

以下の手順を実行する必要があります。

- WINDOWS アプリケーションをコピーします。
- Regie.ini ファイルを展開します。
- モジュール Regie の言語 DLL を展開します。

条件

VISUAL BASIC，VISUAL C++（DLL の変更用）の使い方，および OEM アプリケーション開発についての基本的な知識。

12.5 データ管理サーバ

目的

サンプル OEMBSP5 には，データ管理の機能の使い方が示されています。

簡単な Visual Basic アプリケーションで，以下のデータ管理の機能が使われます。

- ファイルまたはディレクトリの作成
- ファイルまたはディレクトリの削除
- この方法で作成したファイルまたはディレクトリのリスト

条件

VISUAL BASIC，VISUAL C++（DLL の変更用）の使い方，および OEM アプリケーション開発についての基本的な知識。

この知識については，サンプル OEMBSP1 で学ぶことができます。

12.6 PLC データ用の NCDDE サーバ

目的

サンプル OEMBSP6 には，PLC とのデータ交換のために NCDDE サーバを使う方法が示されています。

- PLC データにアクセスし Visual Basic アプリケーションを表示する
- PLC データ作成のためにジョブ DDE-LinkPoke を開始する

条件

VISUAL BASIC，VISUAL C++（DLL の変更用）の使い方，および OEM アプリケーション開発についての基本的な知識。

この知識については，サンプル OEMBSP1 と OEMBSP2 で学ぶことができます。

12.7 WINDOWS アプリケーションの位置決め

目的

サンプル OEMBSP8 には，SINUMERIK ウィンドウ内での標準アプリケーションの位置を決める方法が示されています。

EXCEL，EDITOR，CLOCK のような標準の WINDOWS アプリケーションは，API 機能を使って，SINUMERIK 表示フォームに置くことができます。

条件

VISUAL BASIC，VISUAL C++（DLL の変更用）の使い方，および OEM アプリケーション開発についての基本的な知識。

この知識については，サンプル OEMBSP1 で学ぶことができます。

12.8 Visual Basic と DLL との間のデータ交換の例

目的

サンプル OEMBSP10 では，次の 2 つのアプリケーションを例示します。

- DLL 機能の実現
- VB アプリケーションと DDL 機能との間の簡単なデータ交換

条件

VISUAL BASIC，VISUAL C++（DLL の変更用）の使い方，および OEM アプリケーション開発についての基本的な知識。

12.9 シーケンス制御の機能

目的

OEM アプリケーションで使えない Visual Basic の構成要素（メッセージボックスなど）もあります。シーケンス制御には，代わりのソリューションが備えられています。サンプル 11 には，OEM アプリケーションの開発者が，シーケンス制御のこの機能や別の機能を使って作業を簡単にする方法が示されています。

条件

VISUAL BASIC，VISUAL C++（DLL 編集用）について，そして OEM アプリケーションを作成して統合する方法についての基本的な知識。

12.10 複数の非同期ジョブの逐次処理

目的

データ管理のコマンドによっては，コマンドを発行した VB コマンドが終了しても，まだ終了しないものもあります。サンプル 12 には，複数のコマンドが発行されるときにも正しい順番を保てるようにする方法が示されています。

条件

VISUAL BASIC，VISUAL C++（DLL 編集用）について，そして OEM アプリケーションを作成して統合する方法についての基本的な知識。

さらに，サンプル 5 に示されているデータ管理サーバの機能についても精通している必要があります。

12.11 OEM ソフトキーによるアプリケーションの始動と DCTL 接続の実現

目的

このサンプルには，OEM ソフトキーでアプリケーションを始動する方法，アプリケーションを終了する方法，Visual Basic コントロール 'DCTL.OCX' を使う方法，言語の切り替えを実現する方法が示されています。

条件

VISUAL BASIC，VISUAL C++（DLL 編集用）について，そして OEM アプリケーションを作成して統合する方法についての基本的な知識。

さらに，サンプル 5 に示されているデータ管理サーバの機能についても精通している必要があります。

12.12 ソフトキーによる 32 ビット C++ アプリケーションの操作

目的

このサンプルには，C++ アプリケーション（32 ビット）を適合させて，840DI の特別なユーザパネルで操作する方法が示されています。開発には，Microsoft Developer Studio Visual C++ 4.1 が使われました。これはダイアログベースのアプリケーションでもあり，単一文書のアプリケーションでもあります。

条件

VISUAL BASIC，VISUAL C++（DLL 編集用）について，そして OEM アプリケーションを作成して統合する方法についての基本的な知識。

さらに，サンプル 5 に示されているデータ管理サーバの機能についても精通している必要があります。

12.13 MMC-OEM アプリケーションへの SprintPlus の組み込み

目的

このサンプルには，パラメータ指定ツールで開発されたアプリケーションを，既存の MMC-OEM アプリケーション（ここでは OemBsp0）に組み込む方法が示されています。

条件

VISUAL BASIC，VISUAL C++（DLL 編集用）について，そして OEM アプリケーションを作成して統合する方法についての基本的な知識。

さらに，サンプル 5 に示されているデータ管理サーバの機能についても精通している必要があります。

付録

この章では，OEM パッケージ MMC で使われる略称と技術用語を説明しています。

付録 .1 略称

ACC

ACCESS モジュール

ASCII

情報交換用米国標準コード

ASUP

非同期 NC サブプログラム

BIOS

基本入出力システム

DDE

動的データ交換 : 2 つの WINDOWS アプリケーション間でのデータ交換の方式

DLL

ダイナミックリンクライブラリ

DRAM

ダイナミックランダムアクセスメモリ

IPO

補間（Interpolator）

ISA

業界標準アダプタ : IBM-AT をベースにした PC 拡張アダプタボードのバス規格

MCS

機械座標系

MMC

ヒューマンマシンインタフェース : NC の操作インタフェースで，オペレータのパネルで実行するソフトウェアに適用される

MPI

マルチポートインタフェース

NC

数値制御

NCDDE

NC カーネルでの動的データ交換

NCK

数値制御カーネル。ブロック準備，モーションコントロールなどで構成される

NCU

数値制御ユニット : NCK ソフトウェアに適応する 840DI のハードウェア構成要素

NSK

数値システムキーワード : NCDDE サーバで NC 変数をアドレッシングするときのキーワード

OEM

相手先商標製造会社 : 840DI のオープンシステム機能を使い，独自の目的で標準の構成要素を改良する機械装置メーカー

ユーザ　　パラメータ化

MTB　　構成

OEM　　プログラム

PLC プログラミングの場合は，'OEM' という表現は該当しません。

OP

オペレータのパネルは，モニタ画面，キーパッド，LED，ボタンで構成される。840DI の場合は，さまざまな OP010 が使われている（/BH/ Operator Components Manual 文書で説明されている）

PC

パーソナルコンピュータ

PCMCIA

PCU50 で使われる Personal Computer Memory Card International Association 規格のメモリカード。NCK システムの他の規格とも互換性を持つ。今日では，単に PC カードと呼ばれる。

PI

プログラムインスタンス

PLC

プログラマブル論理コントローラ

RAM

データの読み書きのためのランダムアクセスメモリ

WCS

ワーク座標系

VB

Visual Basic

WfW

Windows for Workgroups

付録 .2 技術用語

アクティブファイルシステム

アクティブファイルシステムには，それぞれの名前で参照される NCK のすべての変数が含まれる。これらの変数は，NCK ソフトウェアの構成と NCK ソフトウェアの特別な機械への適応のために使われるだけでなく，ツールを記述したり，NC パートプログラムで計算変数を指定するときにも使われる。

アラームサーバ

アラームサーバは，MMC のシステムに，現在アクティブなアラームとメッセージを提供する。

軸キー

軸キーは特殊な型宣言（AxKey）で，各軸または主軸に 1 ビットを割り当てる。軸に対応するビットをセットすると，この軸は該当する AxKey で記述された状態になる。最初の軸は，常に bit0 に割り当てられる。

ドメインサービス

NC カーネルに対して相互関連データ（ファイル）のアップロード／ダウンロードを実行する。

動的メモリ

NC の動的メモリに格納されたデータには，一時的な存続時間があり，バッテリで維持される必要がない。この資料では，「動的メモリ」という用語は，「DRAM」および「バッファ不使用メモリ」と同義で使われている。

外部通信

外部通信により，NC カーネルの外部操作インタフェースが実現する。この通信は，BTSS 定義と変数定義に基づいて指定される。

機能単位

ブロック準備および補間に対する ASCII パートプログラムの変換処理から，軸の位置制御に至るまでの処理シーケンスは，複数の機能単位に構造化される。NCK のこれらの機能単位には，NC データの処理方式が含まれる。

それぞれの機能単位は，先行する機能単位の結果を使う。

キーボードサーバ

キーボードサーバは，オペレータのパネルと，対応するコードを，OEM ユーザのために管理する。

ISA バスアダプタ

MMC 101/102 のハードウェア拡張機能 : 標準の PC および AT 拡張ボードを接続するためのスロット（ISA バス）がある。

MMC-OEM 基本システム

MMC-OEM 基本システムは，OEM パッケージ MMC に付属するソフトウェアであり，OEM アプリケーションのベースを提供する。

NC-DDE サーバ

NC-DDE サーバは，MMC-OEM 基本システムの 1 つの構成要素であり，データ転送に関する次の 3 つの作業を行う。

- 変数サービス : NC データ，PLC データ，ドライブデータにアクセスする。
- ドメインサービス : ファイルを MMC から NCK，また NCK から MMC へコピーする。
- 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。

受動ファイルシステム

受動ファイルシステムは，バッテリで維持されるメモリ（SRAM）内に存在する。これは，ファイルおよび NC パートプログラムを格納するために使われる。受動ファイルシステムは階層構造になっていて，ディレクトリとサブディレクトリとで構成される。

PC カード

PC カードは，PCMCIA 規格に従ったメモリカードであり，NCU のシステムファームウェアが含まれている。オプションとして，コントロールの MMC 構成要素に別の PC カードアダプタを接続することができる。これにより，メモリと通信機能を小さなスペースに収めることができる。

PI サービス

PI サービスは，NC カーネルで定義された特別なコマンドの実行を起動する。この機能は，NCU で開始する。

Regie

Regie は，MMC 基本システムの構成要素で，システムの起動をつかさどる。さらに，OEM アプリケーションも起動する。これは，WINDOWS のプログラムマネージャに相当するものである。

シーケンス制御

シーケンス制御には，当社標準のアプリケーションと，互換性のある OEM アプリケーションのための枠組みが備えられている。これは，ソフトキー機能やテキストを含む，メニューの構造を管理する。

シンボル

シンボルには，特別に ASCII 名が付けられたアクティブなファイルシステムの 1 つの要素が記述されている。

変数サービス

変数サービスは，NC カーネルのデータを一度に 1 項目ずつ読み書きする。データ項目（たとえばツールデータ）はそれぞれの ID で指定される。

付録 .2 技術用語

索引

G

H

I

K

L

M

N

U

V

W

Z

あ

い

う

え

お

か

く

Yaskawa Siemens CNC シリーズ

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

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・ジャパン株式会社

工作機械営業本部
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**　〒141-8644
TEL（03）3493-7411　FAX（03）3493-7422

アフターサービス

カスタマーサービス事業本部
TEL 0120-996095（フリーダイヤル）　FAX (03)3493-7433

シーメンス・ジャパン株式会社
http://www.siemens.co.jp

02- WSC02, 02-WSC09