JP1 Version 8　JP1/Script（UNIX(R)用）

目次

1. JP1/Script の概要 4
1.1 JP1/Script とは 4
1.2 JP1/Script を構成するプログラムの種類 4
1.3.1 ファイルの種類 5
1.3.2 ファイルの容量 7
1.3.3 ラージファイル 8
1.3.4 システム環境ファイル 9
1.4 JP1/Script の全体構成 12
1.5 スクリプトを作成する前に，実行する前に 13
2. JP1/Script を利用するための準備 13
2.1 JP1/Script のインストールとアンインストール方法 13
2.1.1 プログラムのインストール先のフォルダ 13
2.1.2 インストール方法 14
2.1.3 アンインストール方法 17
2.2 JP1/Script のセットアップ 18
2.2.1 セットアップ項目 18
2.2.2 環境変数の設定 19
3. JP1/Script の操作 21
3.1 Script 実行環境ファイルコンバータ 21
3.1.1 Script 実行環境ファイルコンバータの使用手順 22
3.1.2 Script 実行環境ファイルコンバータの実行方法 24
3.2 実行環境構文ファイル（.SPU） 28
3.3 Windows 上で作成したスクリプトファイル(.SPT)を UNIX 上に移行する際の注意事項 29
4. JP1/Script の規則 31
4.1 スクリプトに関する規則 31
4.1.1 変数 31
4.1.2 定数 40
4.1.3 数字の記述規則 40
4.1.4 文字列の記述規則 40
4.1.5 演算規則 41
4.1.6 コーディング規則 41
4.1.7 終了コード 44
4.2 コマンドラインに関する規則 45
4.2.1 コマンドラインの形式 45
4.2.2 コマンドラインのパラメタの説明 45
4.2.3 コマンドラインの記述規則 47
4.2.4 コマンドラインに対するエラー処理 50

5. ステートメント 52
5.1 ステートメント一覧 52
5.2 ステートメントの詳細 53
6. 基本コマンド 76
6.1 基本コマンド一覧 76
6.2 変数操作コマンド 82
6.3 文字列操作コマンド 90
6.4 日付操作コマンド 116
6.5 ファイル・ディレクトリ操作コマンド 131
6.6 メッセージ出力コマンド 165
6.7 演算処理コマンド 169
6.8 チェック処理コマンド 190
6.9 外部プログラム呼び出しコマンド 200
6.10 コメントコマンド 203
6.11 その他のコマンド 204
7. 障害発生時の対処方法 208
7.1 スクリプト実行時の障害の対処方法 208
7.1.1 解析トレースファイルおよび実行トレースファイルから原因を調査する 208
7.1.2 syslog ファイルから原因を調査する 208
7.1.3 システム情報ファイルから原因を調査する 211
7.2 バックアップとリカバリー 211
8. メッセージ 212
8.1 メッセージの出力形式 212
8.2 メッセージの記述形式 213
8.3 メッセージの出力先 213

1. JP1/Script の概要

JP1/Script は，UNIX 上でジョブを制御するスクリプトを作成して実行するためのプログラムです。

1.1 JP1/Script とは

JP1/Script は，UNIX 上で，ジョブを制御するスクリプトを実行するためのプログラムです。
また，今までほかのシステムでジョブ制御を行っていた場合でも，ほかのシステムでのジョブ制御と同じようなイメージで，UNIX 上でもジョブ制御を実現できます。
JP1/Script には，次のような特長があります。

多様な業務に幅広く対応できます

- **プログラムの実行結果を取得できます**
  JP1/Script は，プログラムの実行結果を取得できます。実行結果を取得することで，次の処理を判断できるため，プログラムの切り替えなど，柔軟な連携処理を実現できます。
- 別のスクリプトファイルやユーザプログラムを呼び出せます
  別のスクリプトファイルやユーザプログラムを呼び出すことによって，機能を拡張できるため，多様な業務に幅広く対応できます。

使いやすいユーザインタフェースです

- Basic **言語に似たコマンド体系です**
  JP1/Script では，Basic 言語に似たコマンドを使用してスクリプトファイルを作成します。

1.2 JP1/Script を構成するプログラムの種類

JP1/Script は，表 1-1 に示すプログラムによって構成されています。各プログラムの関連性については，「1.4 JP1/Script の全体構成」を参照してください。

表 1-1 JP1/Script のプログラム

プログラムの種類	プログラム名	機能の概要
スクリプト実行制御	sptxe	スクリプトファイル（拡張子.SPT）を解析して実行します。
実行環境ファイルコンバータ	sptuspv	実行環境構文ファイル（拡張子.SPU）と実行環境ファイル（拡張子.SPV）を相互変換できます。

1.3.1 ファイルの種類

JP1/Script で扱うファイルを表 1-2 に示します。
なお，ファイルの容量の計算方法については，「1.3.2 ファイルの容量」を，トレースファイルの出力形式については，「付録 A.1 トレースファイルの出力形式」を参照してください。

表 1-2 JP1/Script で扱うファイルの種類

ファイルの種別	ファイル名	ファイルの拡張子	ファイルの形式	ファイルの内容
実行ファイル	スクリプトファイル	.SPT	テキスト形式	スクリプト文を保存するファイルです。ファイルの拡張子には小文字を指定できます。大文字と小文字を混在させることもできます。
	実行環境ファイル	.SPV	バイナリ形式	一つのスクリプトファイルを実行するときの環境が設定されたファイルです。 ファイル名は，拡張子以外の部分をスクリプトファイル名と同じ名前にしてください。
トレースファイル	解析トレースファイル	.SPA	テキスト形式	スクリプト文の構文解析の結果が保存されたファイルです。 ファイル名は，拡張子以外の部分がスクリプトファイル名と同じになります。
	実行トレースファイル	.SPX	テキスト形式	スクリプト文のコマンドの実行結果が保存されたファイルです。 ファイル名は，拡張子以外の部分がスクリプトファイル名と同じになります。
	ユーザトレースファイル	.TXT	テキスト形式	Message コマンドによって出力されたトレースが保存されたファイルです。 ファイル名はユーザが任意に指定できます。
その他のファイル	トレース管理ファイル※1	.SPB	バイナリ形式	トレースファイルを管理するファイルです。 ファイル名は，/opt/jp1script/data/SPTLOGDB で固定です。
	グローバル変数ファイル	.SPG	バイナリ形式	スクリプトの実行時に設定されたグローバル変数が保存されたファイルです。 ファイル名は，/opt/jp1script/data/SPTGV で固定です。
	予約語ルールファイル	.SPR	バイナリ形式	スクリプトファイルを解析，または実行するために必要となる予約語に関する規則を格納したファイルです。

				ファイル名は，Vervvrr で固定です。vv は JP1/Script のバージョンを表し，rr は JP1/Script のリビジョンを表します。
	ワークファイル	.TMP	バイナリ形式	JP1/Script 内部で利用するワークファイルです。
	システム環境ファイル	.cf	テキスト形式	スクリプトの実行環境を設定するファイルです。ファイル名は/opt/jp1script/conf/jp1script で固定です。
	実行環境構文ファイル	.SPU	テキスト形式	Script 実行環境ファイルコンバータで利用するファイルです。ファイルの拡張子には小文字を指定できます。また，大文字と小文字を混在させることもできます。実行環境ファイルに設定する内容が文字列で設定されています。 ファイル名はユーザが任意に指定できます。
	システム情報ファイル	.LOG	テキスト形式	実行時のマシン環境や実行環境を格納するファイルです。ファイルは/opt/jp1script/log/sptenv ディレクトリに作成されて，ファイル名は SPTENVnn.LOG になります（nn は 01～50）。

注※1 **トレース管理ファイルは，以下のトレースファイルのファイル名，およびトレースファイルの書き出し位置などを管理しているファイルです。**

- 解析トレースファイル
- 実行トレースファイル
- ユーザトレースファイル

次の条件の場合，トレース管理ファイルで管理するトレースファイルが増えるためトレース管理ファイルの容量が増えます。
ファイル名の異なるスクリプトを実行する
Message コマンドで Target 引数に「Target_File」を指定して新規のファイルに出力する
ファイル名の異なるスクリプトを実行した場合，そのスクリプトファイルの解析トレースファイル，および実行トレースファイルを管理するため，容量が増えます。同じスクリプトファイルを複数回実行する場合は，トレース管理ファイルに出力されている情報を使用しますので容量が増えることはありません。
Message コマンドで Target 引数に「Target_File」を指定して新規のファイルに出力する場合，容量が増えます。
Message コマンドの OutputName 引数に指定したファイルが，すでに Message コマンドで Target 引数に「Target_File」を指定して出力済のファイルの場合，トレース管理ファイルに出力されている情報を使用しますので容量が増えることはありません。
Message コマンドで Target 引数に「Target_File」を指定して，ユニークなファイル名称を持ったユーザトレースファイルを大量に作成する場合，大量に作成されたファイル名，およびトレースファイルの書

き出し位置等を管理するための情報が増えるため，トレース管理ファイルの容量が増えてしまいます。
トレース管理ファイルの容量が増えると，以下の現象が発生します。
スクリプトファイルの実行性能が劣化する
スクリプトファイルの実行が終了コード「20」で異常終了する
コマンドの実行がメモリ不足でエラーになる
ユニークなファイル名称を持ったユーザトレースファイルを大量に作成する場合は，
TextOpen/TextWrite/TextClose コマンドを使用して，トレースファイルを作成してください。
TextOpen/TextWrite/TextClose コマンドで作成するファイルは，トレース管理ファイルの管理対象外のファイルになるため，トレース管理ファイルの容量が増えることはありません。

1.3.2 ファイルの容量

JP1/Script で扱う各ファイル（予約語ルールファイル，ワークファイル，システム環境ファイル，および実行環境構文ファイル以外のファイル）の容量の計算方法，およびおよその容量を次に示します。

(1) 実行ファイル

スクリプトファイル（.SPT）
スクリプトファイルは，ユーザがどれだけの量のスクリプトを記述するのかによって，容量が異なります。したがって，スクリプトファイルのファイル容量は，ユーザ任意となります。
実行環境ファイル（.SPV）
ファイル容量は，8～12KB です。

(2) トレースファイル

解析トレースファイル（.SPA）
計算方法を次に示します。（単位：バイト）
（最大列数+1）×最大行数
最大行数，および最大列数は，実行環境ファイルに設定した値です。
実行トレースファイル（.SPX）
計算方法を次に示します。（単位：バイト）
（最大列数+1）×最大行数
最大行数，および最大列数は，実行環境ファイルに設定した値です。
ユーザトレースファイル（トレースファイル）
計算方法を次に示します。（単位：バイト）
（LineLength で指定した値+1）×（MaxLines で指定した値+1）
LineLength，および MaxLines を省略した場合，最大行数，および最大列数は，実行環境ファイルに設定した値です。LineLength，および MaxLines の詳細は，「6.6 メッセージ出力コマンド」の「Message」を参照してください。

(3) その他のファイル

トレース管理ファイル（.SPB）
初期に割り当てるサイズは，12KB です。また，拡張サイズは 12KB です。なお，初期割り当てサイズで，8 個のトレースファイルを格納できます。
グローバル変数ファイル（.SPG）

初期に割り当てるサイズは，20KB です。また，拡張サイズは 20KB です。
システム情報ファイル
ファイル容量は，最大 6,200KB です。
計算方法を次に示します。（単位：バイト）
列数 124（固定）×最大行数 1,024×最大個数 50

1.3.3 ラージファイル

(1) ファイルシステム

ラージファイルを使用するためには，ファイルシステムの属性がラージファイルに対応している必要があります。ファイルシステムの属性については，プラットフォームのリファレンスを参照してください。
また，作成可能なファイルサイズについては，システム資源の制限が設定されている場合があります。
詳細は，各プラットフォームの ulimit コマンドなどのリファレンスを参照してください。
なお，下記の場合は，ラージファイルは未対応になります。
HP-UX で Disk Layout Version 2 を使用している場合

(2) ファイルの最大サイズ

ファイルの最大サイズは，プラットホームのファイルシステムに依存しますが，JP1/Script として扱えるファイルの最大サイズは，9,223,372,036,854,775,807 バイトです。それ以上の場合の動作は保証しません。

(3) エラーメッセージ

ラージファイル対応のエラーメッセージを，表 1-3 に示します。

表 1-3 ラージファイル対応のエラーメッセージ

No.	コマンド	エラーメッセージ	予約変数（値）
1	GetFileSize	取得した値が変数の上限値を超えているため，値を変数に格納できません。	_ERR_FILE_SIZE_（536904784）
2	TextFileReplace	指定したファイルの容量が制限値を超えています。	_ERR_NOT_LARGE_FILE_（536904785）
3	GetTextPosition	読み書き開始位置が，2,147,483,648 を超えています。	_ERR_FILE_POSITION_（536904786）

1.3.4 システム環境ファイル

システム環境ファイルの内容を，表 1-4 に示します。

表 1-4 システム環境ファイルの内容

No.	キーワード	指定内容	備考
1	[Extensions]	拡張子により起動する実行ファイル名。	変更できません
2	spt	スクリプトファイルの実行ファイル名。	変更できません
3	sh	シェルファイルの実行ファイル名。	変更できません
4	[OPTION]	オプション。	変更できません
5	%USER_DATA%	ユーザが変更できるデータ。	変更できません
6	CommandLine	実行するコンピュータですべてのスクリプトファイルに対して，有効になるコマンドラインの値（デフォルトは指定無し）。	
7	NoEvLog_Opt	-SPT:NOSYSLOG，または-SPT:NOSYSLOG=n,n,…指定時の扱い。 1 以外：位置変数として扱う 1　：位置変数として扱わない（デフォルト）	
8	DQString_Opt	n個のダブルクォーテーション(")を文字としてそのままパラメタに含める場合の指定方法。 1 以外：n＊4 個指定※1 1　：n＊2 個指定（デフォルト）	
9	AlreadyRun	指定されたスクリプトファイルがすでに起動されている場合の終了コード※2。デフォルトは 16。	
10	GrammarError	文法エラーが発生した場合の場合の終了コード※2。デフォルトは 19。	
11	ExAbortError	JP1/Script のプロセスを中断する実行エラー（メモリ不足，未定義の変数を参照している，プロシージャが見つからないエラーなど）が発生した場合の終了コード※2。デフォルトは 20。	
12	Error	JP1/Script のプロセス開始前にエラー（指定されたスクリプトファイルが見つからないエラーなど）が発生した場合の終了コード※2。デフォルトは 99。	

13	ErrorEventLog	解析トレース，および実行トレース未出力時の syslog 出力指定。 1 以外：出力しない（デフォルト） 1　：出力する（ログ ID 98 で出力される）	
14	Trace_Folder	複数起動時に出力するトレースファイルの出力先フォルダを指定します。 デフォルトは/opt/jp1script/data。	
15	Env_Directory	システム情報ファイルの出力先（デフォルトは指定なし）。	
16	GetPath_Opt	GetPath で取得したパスの末尾の（/）付加指定。 1 以外：付加しない 1　：付加する	
17	EngineVersion	スクリプトファイルを実行するときに使用するスクリプトエンジンのバージョンを指定します。実行するスクリプトファイルの先頭に#FileVersion が記述されていない場合，または記述に誤りがあった場合に使用されます。	変更できません
18	%SYSTEM_DATA%	システムで使用するデータ	変更できません
19	CurrentVersion	カレントバージョン	変更できません
20	ExCurrentVersion	拡張カレントバージョン	変更できません
21	ProductId	PP 形名	変更できません
22	Path01	パス	変更できません
23	Path02	パス	変更できません
24	Path04	パス	変更できません
25	Path05	パス	変更できません
26	DB_Folder	DB 出力先フォルダ	変更できません
27	DB_InitCount	DB 初期割り当て値	変更できません
28	ReStartCount	SPV,SPT,SPR ファイルオープン時にアクセスエラーが起きた場合のリトライ回数	変更できません
29	ReStartWaitTime	SPV,SPT,SPR ファイルオープン時にアクセスエラーが起きた	変更できま

		場合のリトライ間隔	せん

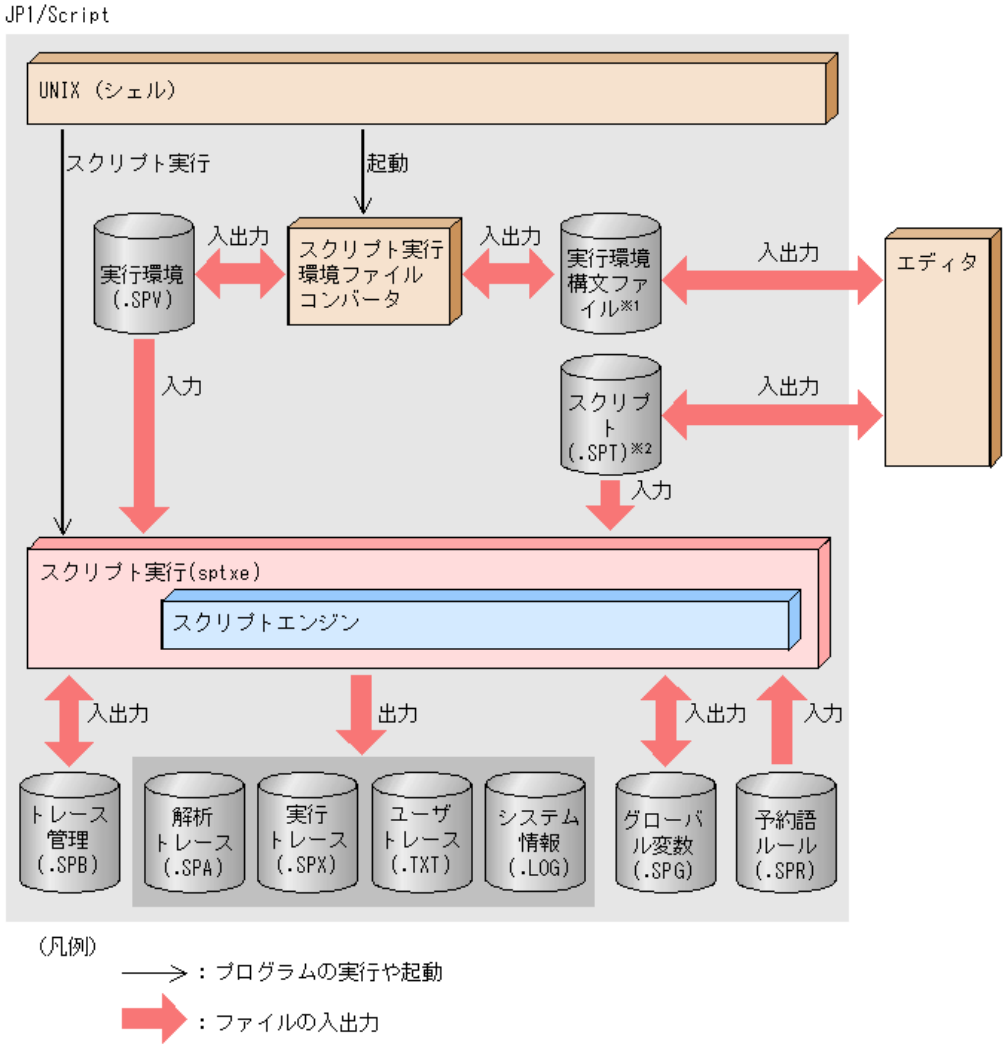
注※1　ユーザプログラムなどから実行形式（sptxe）でコマンドラインを指定してスクリプトファイルを実行する場合の値です。Exec コマンドからパラメタを指定してスクリプトファイルを呼び出す場合は，（n＊4）個とパラメタ全体を囲むダブルクォーテーションのそれぞれを 2 倍したダブルクォーテーションを指定してください。

注※2　スクリプトの実行時にエラーになった場合に呼び元に返される値です。Exec コマンドでスクリプトファイルを呼び出した場合，この値は_EXEC_RTN_に設定されます。

1.4 JP1/Script の全体構成

JP1/Script は，スタンドアロンで動作するプログラムです。JP1/Script の全体構成を図 1-1 に示します。

図 1-1 JP1/Script の全体構成



注※1実行環境構文ファイルは，Windows版JP1/Scriptで作成した実行環境ファイルをWindows版のスクリプト実行環境ファイルコンバータで実行環境構文ファイルに変換し，転送して運用できます。
注※2Windows版JP1/Scriptで作成したスクリプトファイルを転送して運用できます。

注意事項

ハードウェア構成は，クラスタ構成には対応しません。

1.5 スクリプトを作成する前に，実行する前に

(1) Message コマンドについて

Message コマンドで Target 引数に「Target_File」を指定して，ユニークなファイル名称を持ったユーザトレースファイルを大量に作成する場合，大量に作成されたファイル名，およびトレースファイルの書き出し位置などを管理するための情報が増えるため，トレース管理ファイルの容量が増えます。トレース管理ファイルの容量が増えると，以下の現象が発生します。

スクリプトファイルの実行性能が劣化する

スクリプトファイルの実行が終了コード「20」で異常終了する

コマンドの実行がメモリ不足でエラーになる

ユニークなファイル名称を持ったユーザトレースファイルを大量に作成する場合は，TextOpen/TextWrite/TextClose コマンドを使用して，トレースファイルを作成してください。TextOpen/TextWrite/TextClose コマンドで作成するファイルは，トレース管理ファイルの管理対象外のファイルになるため，トレース管理ファイルの容量が増大することはありません。

(2) 解析トレースファイルおよび実行トレースファイルについて

解析トレースファイル，および実行トレースファイルは標準でスクリプト実行時に発生するエラー情報を出力します。実行環境の設定によりトレースファイルを出力しない設定も可能ですが，スクリプト実行時に発生するステートメントやコマンドのエラーが出力されないため，原因が特定できない場合があります。このため，トレースファイルはできるだけ出力するようにしてください。トレースファイルを出力したくない場合は，各コマンドの実行結果を判定して，エラーの場合は，エラーとなったコマンド名，および_RTN_予約変数の内容をファイル等に出力して原因を特定する方法など，エラー処理設計を考慮したスクリプトを作成してください。

2. JP1/Script を利用するための準備

この章では，JP1/Script のインストールとアンインストール方法，および JP1/Script のセットアップについて説明します。

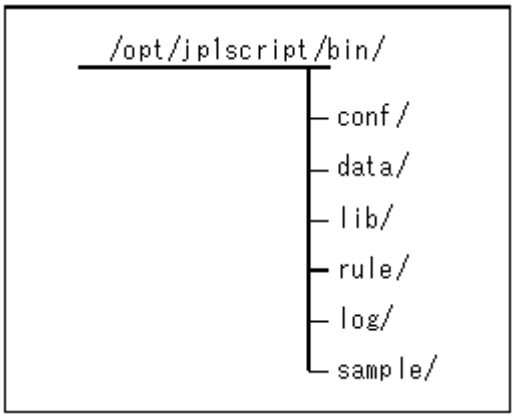
2.1 JP1/Script のインストールとアンインストール方法

ここでは，JP1/Script のインストールとアンインストール方法について説明します。

2.1.1 プログラムのインストール先のフォルダ

JP1/Script プログラムのインストール先のフォルダを次に示します。

図 2-1　JP1/Script プログラムのインストール先のフォルダ構成



2.1.2　インストール方法

UNIX ホストに JP1/Script をインストールする方法を説明します。
インストールする方法には，JP1/NETM/DM を使用してリモートインストールする方法と，提供媒体を使用してインストールする方法があります。
JP1/NETM/DM を使用する方法では，次に示すソフトウェアが必要です。なお，各ソフトウェアの詳細は次のマニュアルを参照してください。
JP1/NETM/DM Manager
マニュアル「JP1 Version 6 JP1/NETM/DM Manager」
JP1/NETM/DM SubManager
マニュアル「JP1 Version 7i JP1/NETM/DM SubManager（UNIX(R)用）」
マニュアル「JP1 Version 8 JP1/NETM/DM SubManager（UNIX(R)用）」
JP1/NETM/DM Client
マニュアル「JP1 Version 7i JP1/NETM/DM Client（UNIX(R)用）」
マニュアル「JP1 Version 8 JP1/NETM/DM Client（UNIX(R)用）」
ここでは，提供媒体を使用する場合のインストール手順を OS 別に説明します。
インストールには，HITACHI PP Installer を使用します。

注意事項

- JP1/Script をインストールする場合は，必ずローカルホストのスーパーユーザのアカウントを使用してください。

JP1/Script を初めてインストールする場合は，インストール先ディレクトリにほかのファイルやディレクトリがないことを確認してください。
JP1/Script を上書きしてインストールする場合は，JP1/Script のプログラムをすべて停止してください。
JP1/Script を上書きしてインストールする場合，/opt/jp1script 下のファイルはすべて上書きされます。
/opt/jp1script 下に必要なファイル（システム環境ファイルを更新している場合など）がある場合は，別

のディレクトリに退避して，インストール後に回復してください。

(1) HP-UX の場合のインストール方法

JP1/Scriptをインストールするホストに，スーパーユーザでログインするか，またはsuコマンドでユーザの権限をスーパーユーザに変更する。
ローカルホストで JP1/Script のプログラムが起動中の場合は，すべて停止する。
ほかに起動中のアプリケーションプログラムがあれば，すべて終了する。
JP1/Script の媒体をセットする。
mount コマンドを実行して，CD-ROM 装置をマウントする。※
例えば，CD-ROM 装置を/cdrom にマウントする場合，次のように指定してコマンドを実行します。
/usr/sbin/mount -F cdfs -r デバイススペシャルファイル名 /cdrom
なお，指定するコマンドは，使用している環境によって異なります。
次のコマンドを実行して，Hitachi PP Installer を起動する。※
HP-UX の場合

/cdrom/HPUX/setup /cdrom

HP-UX(IPF)の場合

/cdrom/IPFHPUX/setup /cdrom

HITACHI PP Installer が起動されて，初期画面が表示されます。
初期画面で「I」を入力する。
インストールできるプログラムの一覧が表示されます。
JP1/Script を選択して，「I」を入力する。
選択したプログラムがインストールされます。
プログラムを選択する場合は，カーソルを移動させて，スペースバーで選択します。
インストールが正常終了したら，「Q」を入力する。
Hitachi PP Installer の初期画面に戻ります。

(2) AIX の場合のインストール方法

1. JP1/Script をインストールするホストに，スーパーユーザでログインするか，または su コマンドでユーザの権限をスーパーユーザに変更する。
2. ローカルホストで JP1/Script のプログラムが起動中の場合は，すべて停止する。
3. ほかに起動中のアプリケーションプログラムがあれば，すべて終了する。
4. JP1/Script の媒体をセットする。
5. mount コマンドを実行して，CD-ROM 装置をマウントする。※
   例えば，CD-ROM 装置を/cdrom にマウントする場合，次のように指定してコマンドを実行します。
   /mount -r -v cdrfs /dev/cd0 /cdrom

6. 次のコマンドを実行して，Hitachi PP Installer を起動する。※
   /cdrom/AIX/setup /cdrom
   HITACHI PP Installer が起動され，初期画面が表示されます。
7. 初期画面で「I」を入力する。
   インストールできるプログラムの一覧が表示されます。
8. JP1/Script を選択して，「I」を入力する。
   選択したプログラムがインストールされます。
   プログラムを選択する場合は，カーソルを移動させて，スペースバーで選択します。
9. インストールが正常終了したら，「Q」を入力する。
   Hitachi PP Installer の初期画面に戻ります。

(3) Solaris の場合のインストール方法

1. JP1/Script をインストールするホストに，スーパーユーザでログインするか，または su コマンドでユーザの権限をスーパーユーザに変更する。
2. ローカルホストで JP1/Script のプログラムが起動中の場合は，すべて停止する。
3. ほかに起動中のアプリケーションプログラムがあれば，すべて終了する。
4. JP1/Script の媒体をセットする。
5. CD-ROM 装置は，通常は自動マウントされていますが，マウントされていない場合は，mount コマンドを実行して，CD-ROM 装置をマウントする。※
   例えば，CD-ROM 装置を/cdrom にマウントする場合，次のように指定してコマンドを実行します。
   mount -r -v -F hsfs デバイススペシャルファイル名 /cdrom
6. 次のコマンドを実行して，Hitachi PP Installer を起動する。※
   /cdrom/SOLARIS/setup /cdrom
   HITACHI PP Installer が起動され，初期画面が表示されます。
7. 初期画面で「I」を入力する。
   インストールできるプログラムの一覧が表示されます。
8. JP1/Script を選択して，「I」を入力する。
   選択したプログラムがインストールされます。
   プログラムを選択する場合は，カーソルを移動させて，スペースバーで選択します。
9. インストールが正常終了したら，「Q」を入力する。
   Hitachi PP Installer の初期画面に戻ります。

(4) Linux の場合のインストール方法

1. JP1/Script をインストールするホストに，スーパーユーザでログインするか，または su コマンドでユーザの権限をスーパーユーザに変更する。
2. ローカルホストで JP1/Script のプログラムが起動中の場合は，すべて停止する。
3. ほかに起動中のアプリケーションプログラムがあれば，すべて終了する。

4. JP1/Script の媒体をセットする。
5. mount コマンドを実行して，CD-ROM 装置をマウントする。※
   例えば，CD-ROM 装置を/mnt/cdrom にマウントする場合，次のように指定してコマンドを実行します。
   mount -r -o mode=0544 /dev/cdrom /mnt/cdrom
6. 次のコマンドを実行して，Hitachi PP Installer を起動する。※
   Linux の場合
7.
8. /mnt/cdrom/LINUX/setup /mnt/cdrom

   Linux(IPF)の場合

   /mnt/cdrom/IPLINUX/setup /mnt/cdrom

   Hitachi PP Installer が起動され，初期画面が表示されます。
9. 初期画面で「I」を入力する。
   インストールできるプログラムの一覧が表示されます。
10. JP1/Script を選択して「I」を入力する。
    選択したプログラムがインストールされます。
    プログラムを選択する場合は，カーソルを移動させて，スペースバーで選択します。
11. インストールが正常終了したら，「Q」を入力する。
    Hitachi PP Installer の初期画面に戻ります。

注※

インストールの作業をするホスト上に，すでに HITACHI PP Installer がインストールされている場合，/etc/hitachi_setup コマンドを次のように指定して HITACHI PP Installer を起動することもできます。

/etc/hitachi_setup -i /mnt/cdrom

2.1.3 アンインストール方法

UNIX ホストから JP1/Script をアンインストールする方法を説明します。

1. JP1/Script をアンインストールするホストに，スーパーユーザでログインするか，または su コマンドでユーザ権限をスーパーユーザに変更する。
2. JP1/Script のプログラムが起動中の場合は，すべて停止する。
3. ほかに起動中のアプリケーションプログラムがあれば，すべて終了する。

4. 次のコマンドを実行して，Hitachi PP Installer を起動する。

   /etc/hitachi_setup

   HITACHI PP Installer が起動されて，初期画面が表示されます。

5. 初期画面で「D」を入力する。

   アンインストールできるプログラムの一覧が表示されます。

6. JP1/Script を選択して，「D」を入力する。

   選択したプログラムがアンインストールされます。

   プログラムを選択する場合は，カーソルを移動させて，スペースバーで選択します。

7. アンインストールが正常終了したら，「Q」を入力する。

   Hitachi PP Installer の初期画面に戻ります。

注意

- JP1/Script をアンインストールする場合は，必ずローカルホストのスーパーユーザのアカウントを使用してください。

JP1/Script をアンインストールする場合は，JP1/Script のすべてのプログラムを停止してください。

2.2 JP1/Script のセットアップ

ここでは，JP1/Script を運用するためのセットアップについて説明します。

2.2.1 セットアップ項目

JP1/Script を運用するためには，JP1/Script をインストールしたホストコンピュータの環境変数を設定する必要があります。
運用のために設定する環境変数を，表 2-1 に示します。

表 2-1　運用のために設定する環境変数

設定する環境変数	AIX 版 JP1/Script	HP-UX 版 JP1/Script	Solaris 版 JP1/Script	Linux 版 JP1/Script
LANG	○	○	○	○
PATH	○	○	○	○
LIBPATH	○	―	―	―
LD_LIBRARY_PATH	―	―	―	○

（凡例）

○：設定する。

―：設定しない。

2.2.2　環境変数の設定

(1)　環境変数 LANG の設定

環境変数 LANG で使用する言語を決定します。JP1/Script で設定できる環境変数 LANG を表 2-2 に示します。

表 2-2　JP1/Script で設定できる環境変数 LANG

OS	言語の種類	環境変数 **LANG の設定値**
AIX	日本語（Shift-JIS コード）	Ja_JP.IBM-943, Ja_JP
	日本語（EUC コード）	ja_JP
	英語（日本語なし）	C
HP-UX	日本語（Shift-JIS コード）	ja_JP.SJIS
	日本語（EUC コード）	ja_JP.eucJP
	英語（日本語なし）	C
Solaris	日本語（Shift-JIS コード）	ja_JP.PCK
	日本語（EUC コード）	ja, japanese
	英語（日本語なし）	C
Linux	日本語（EUC コード）	ja_JP.eucJP, ja_JP.EUC
	日本語（UTF-8）※1	ja_JP.UTF-8
	英語（日本語なし）	C

スクリプトファイル中に全角の変数名，全角の文字列を指定している場合，またはスクリプトで全角文字データを扱う場合，環境変数 LANG とスクリプトファイルで使用している全角文字の文字コードを統一してください。なお，環境変数 LANG に英語('C')を設定している場合，
AIX,HP-UX,Solaris,HP-UX11iV2(IPF)では Shift-JIS コード，Linux では EUC コードとして扱います。
言語の種類で日本語を設定していても，一部のメッセージは，英語で出力されます。言語の種類が指定できるメッセージと言語の種類が英語固定になるメッセージは下記のとおりです。
環境変数 LANG で言語の種類が設定できるメッセージ
・実行トレースファイルに出力されるメッセージ
・解析トレースファイルに出力されるメッセージ
・スクリプト実行中に障害が起きた時に標準出力に出力されるエラーメッセージ
言語が英語固定で出力されるメッセージ
・syslog に出力されるメッセージ

注※1　BOM 付きファイルの扱いについて

BOM（Byte Order Mark）は，データのバイトの並び順序を表すコードで，UTF-8 で記述されたテキストファイルの先頭に付加されている場合があります。JP1/Script のファイル・ディレクトリ操作コマンドのうちファイルの内容を編集するコマンドは BOM を無視し以下の動きになります。

TextRead コマンドは読み込んだ行に BOM が付いている場合は無視し，BOM 以降のデータを読み込みます。
TextWriteコマンドはBOMを付加して出力することはできません。また，BOM付きのファイルの先頭行を上書きする場合は BOM を上書きします。
TexeFileReplace コマンドは変換前文字列，および変換後文字列に BOM を指定できません。
その他，以下のファイル・ディレクトリ操作コマンドはファイルの内容に依存しないため BOM を文字データとみなして処理します。

TextSeek,GetTextposition,Rename,SetFileTime,GetFileTime,GetFileSize,SplitFile,CatFilse,SetStanderdFile,ResetStandardFile，および Copy コマンド

(2) 環境変数 PATH の設定

JP1/Script を使用する場合は，B シェルファイル，または C シェルファイルに，環境変数 PATH で JP1/Script の設定されているディレクトリのパスを追加します。

環境変数 PATH を追加しないで JP1/Script を実行する場合は，JP1/Script の実行ファイルを完全名形式（/opt/jp1script/bin/sptxe）で指定してください。

PATH 環境変数の設定例を次に示します。

B シェル（K シェル）の場合

```
PATH=${PATH}:/opt/jp1script/bin
export　PATH
```

Cシェルの場合

setenv　PATH　${PATH}:/opt/jp1script/bin

(3) 環境変数 LIBPATH の設定

Bシェルファイル，またはCシェルファイルに，環境変数 LIBPATH で JP1/Script のライブラリが設定されているディレクトリのパスを追加します。

環境変数 LIBPATH の設定例を次に示します。

Bシェル（Kシェル）の場合

LIBPATH=${LIBPATH}:/opt/jp1script/lib

export　LIBPATH

Cシェルの場合

setenv　LIBPATH　${LIBPATH}:/opt/jp1script/lib

(4) 環境変数 LD_LIBRARY_PATH の設定

bash ファイル，または C シェルファイルに，環境変数 LD_LIBRARY_PATH で JP1/Script のライブラリが設定されているディレクトリのパスを追加します。

環境変数 LD_LIBRARY_PATH の設定例を次に示します。

bash の場合

LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:/opt/jp1script/lib

export　LD_LIBRARY_PATH

Cシェルの場合

setenv　LD_LIBRARY_PATH　${LD_LIBRARY_PATH}:/opt/jp1script/lib

3. JP1/Script の操作

この章では，JP1/Script のツールの操作方法，ツールによって作成されるファイルの内容，および Windows 上で作成したスクリプトを移行する際の注意事項について説明します。

3.1 Script 実行環境ファイルコンバータ

Script 実行環境ファイルコンバータとは，Windows 上の実行環境ファイル（.SPV）を UNIX 上に移行する，または UNIX 上で実行環境ファイルを作るためのツールです。Script 実行環境ファイルコンバータを使用すると，Windows 上の実行環境ファイル（.SPV）を変換して，実行環境構文ファイル（.SPU）を作成できます。

実行環境ファイル(.SPV)とは，スクリプトの実行環境が設定されているファイルです。また，実行環境構文ファイル(.SPU)とは，実行環境ファイル(.SPV)の内容がテキスト形式で記述されているファイルです。
この節では，Script実行環境ファイルコンバータの使用手順，およびScript実行環境ファイルコンバータの実行方法について説明します。

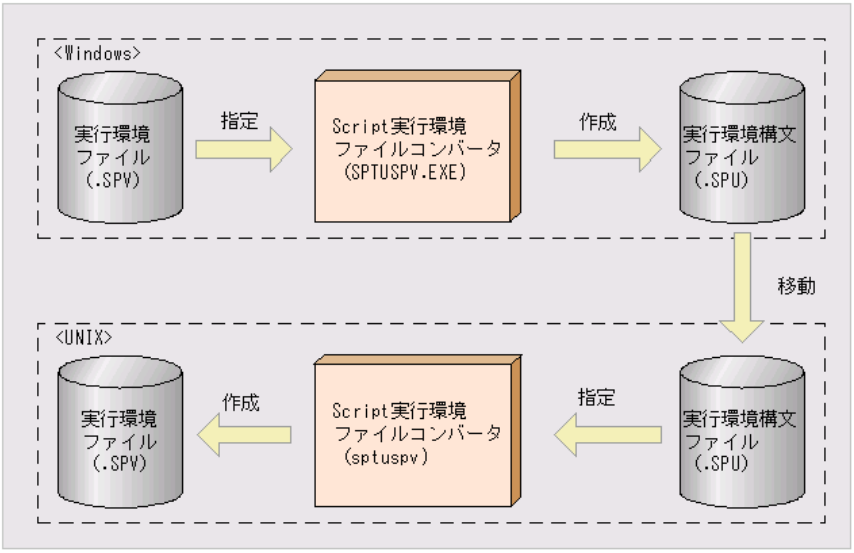
3.1.1 Script 実行環境ファイルコンバータの使用手順

Script 実行環境ファイルコンバータを使用して，次の作業ができます。
Windows 上の実行環境ファイル(.SPV)を UNIX 上に移行する。
UNIX 上で実行環境ファイル(.SPV)を編集する。
UNIX 上で実行環境ファイル(.SPV)を新規作成する。
それぞれの作業の手順を，次に説明します。

(1) Windows 上の実行環境ファイル(.SPV)を UNIX 上に移行する

Windows 上の実行環境ファイル(.SPV)を UNIX 上に移行する手順の概要を，次の図に示します。

図 3-1 実行環境ファイル(.SPV)の移行手順



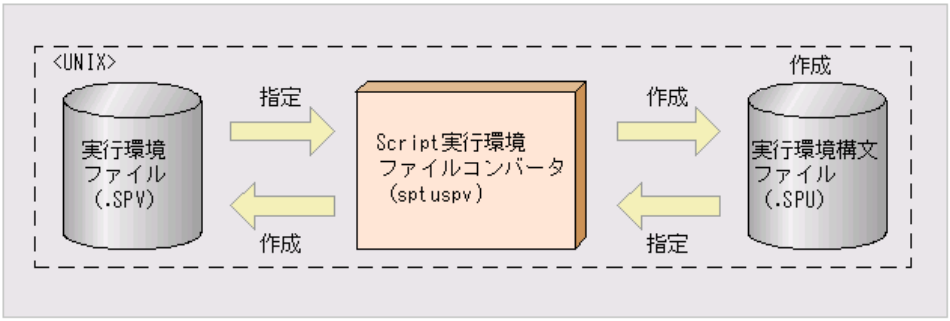
Windows 上の実行環境ファイル(.SPV)を，UNIX 上に移行する手順を次に示します。

1. 移行する Windows 上の実行環境ファイル(.SPV)を，Script 実行環境ファイルコンバータに指定して実行する。
   実行環境構文ファイル(.SPU)が作成されます。
2. 作成された実行環境構文ファイル(.SPU)を FTP 転送で UNIX 上に移動する。
3. 手順 2.で移動した実行環境構文ファイル(.SPU)を，Script 実行環境ファイルコンバータに指定して実行する。
   UNIX 上で使用できる実行環境ファイル(.SPV)が作成されます。

(2) UNIX 上で実行環境ファイル(.SPV)を編集する

UNIX 上で実行環境ファイル(.SPV)を編集する手順の概要を，次の図に示します。

図 3-2　実行環境ファイル(.SPV)の編集手順



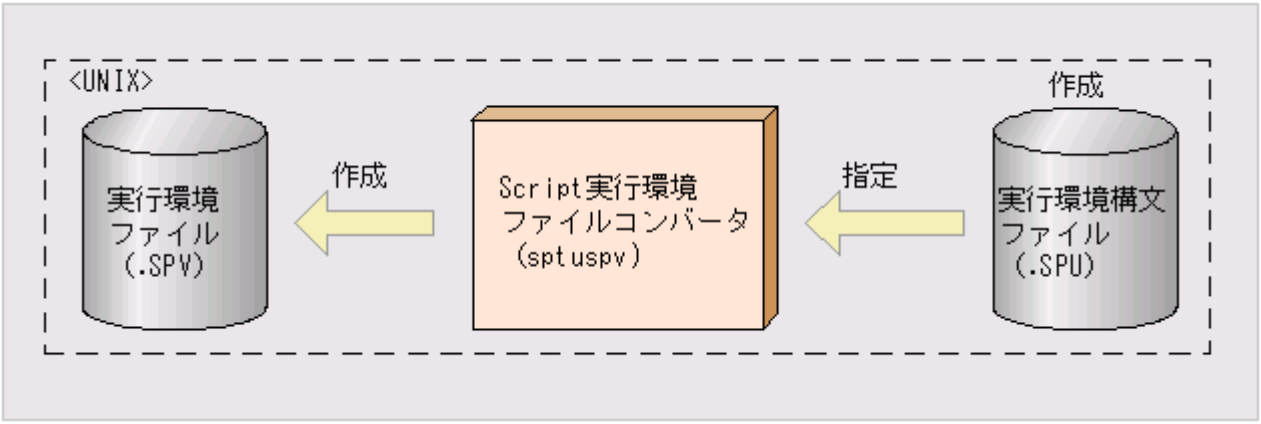
UNIX 上で実行環境ファイル(.SPV)を編集する手順を次に示します。

1. 編集したい実行環境ファイル(.SPV)を，Script 実行環境ファイルコンバータに指定して実行する。
   実行環境構文ファイル(.SPU)が作成されます。
2. 作成された実行環境構文ファイル(.SPU)を編集する。
3. 手順 2.で編集した実行環境構文ファイル(.SPU)を，Script 実行環境ファイルコンバータに指定して実行する。
   編集された実行環境ファイル(.SPV)が作成されます。

(3)　UNIX 上で実行環境ファイル(.SPV)を新規作成する

UNIX 上で実行環境ファイル(.SPV)を新規作成する手順の概要を，次の図に示します。

図 3-3　実行環境ファイル(.SPV)の新規作成手順



UNIX 上で実行環境ファイル(.SPV)を新規作成する手順を次に示します。

1. 新規作成したい実行環境ファイル(.SPV)の，実行環境構文ファイル(.SPU)を作成する。
2. 作成した実行環境構文ファイル(.SPU)を，Script 実行環境ファイルコンバータに指定して実行する。
   UNIX 上で使用できる実行環境ファイル(.SPV)が作成されます。

3.1.2 Script 実行環境ファイルコンバータの実行方法

Script 実行環境ファイルコンバータを実行するには，コマンドラインに次の形式で入力します。

(1) 実行環境構文ファイル（.SPU）から実行環境ファイル（.SPV）への変換

変換元ファイルの実行環境構文ファイルの拡張子に「.SPU」を指定した場合，この変換を行います。

形式

sptuspv△-i△変換元ファイル名（△-o△出力先ディレクトリ名）（△オプション）

説明

-i△変換元ファイル名

実行環境ファイル（.SPV）に変換する実行環境構文ファイル（.SPU）を指定します。ラベル名にワイルドカード指定ができます。拡張子に指定できるのは「.SPU」，または「.SPV」だけです。これらの拡張子以外を指定した場合はエラーになります。-i と変換元ファイル名の間には，スペースを指定してください。

-o△出力先ディレクトリ名

変換後の実行環境ファイル（.SPV）が出力されるフォルダを指定します。指定を省略した場合，変換元ファイルと同じフォルダに出力されます。変換後のファイルがすでにある場合は，今までの内容はすべて失われます。-o と出力先ディレクトリ名の間には，スペースを指定してください。

オプション

次のオペランドを指定できます。

-SPT:ALL

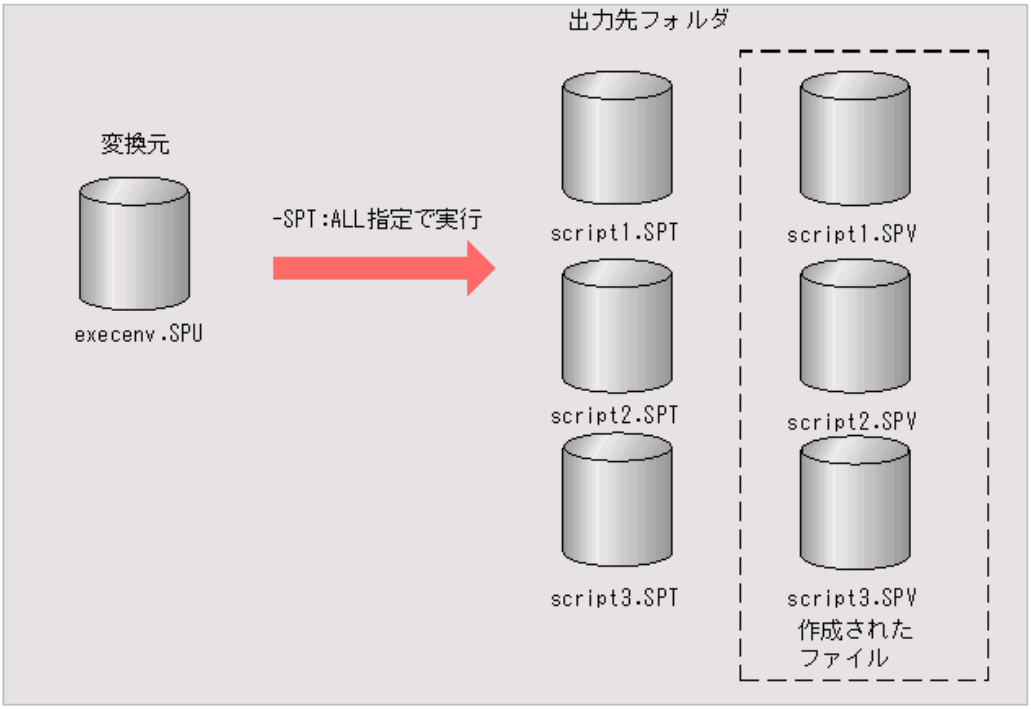
一つの実行環境構文ファイル（.SPU）から出力先フォルダに存在するすべてのスクリプトファイル（.SPT）に適用させたい場合に指定します。

-SPT:ALL を指定した場合，変換元ファイル名にワイルドカードは指定できません（オペランドの組み合わせエラーになります）。

ただし，ワイルドカードを指定した場合でも，ワイルドカードの対象となるファイルが 1 個だけの場合は，エラーになりません。

-SPT:ALL を指定した場合の実行環境ファイル作成の流れを次の図に示します。

図 3-4 -SPT:ALL を指定した場合の実行環境ファイル作成の流れ



変換元ファイルにexecenv.SPU，出力先フォルダにDir，オペランド-SPT:ALLを指定して実行した場合，出力先に存在したscript1～script3.SPTに対して，実行環境ファイルscript1～script3.SPVがexecenv.SPUの内容で作成されます。
オペランドに-SPT:ALLを指定しない場合は，出力先フォルダにexecenv.SPUだけが作成されます。

(2) 実行環境ファイル(.SPV)から実行環境構文ファイル(.SPU)への変換

変換元ファイルの実行環境構文ファイルの拡張子に「.SPV」を指定した場合，この変換を行います。

形式

sptuspv△-i△変換元ファイル名(△-o△出力先ディレクトリ名)

説明

-i△変換元ファイル名

実行環境構文ファイル(.SPU)に変換する実行環境ファイル(.SPV)を指定します。ラベル名にワイルドカード指定ができます。拡張子に指定できるのは「.SPU」，または「.SPV」だけです。これらの拡張子以外を指定した場合はエラーになります。-i と変換元ファイル名の間には，スペースを指定してください。

-o△出力先ディレクトリ名

変換後の実行環境構文ファイル(.SPU)が出力されるフォルダを指定します。指定を省略した場合，変換元ファイルと同じフォルダに出力されます。変換後のファイルがすでにある場合は，今までの内容はすべて失われます。-o と出力先ディレクトリ名の間には，スペースを指定してください。

戻り値

正常に終了した場合は，0 が返されます。正常に終了しなかった場合は，そのエラーコードが返されます。

表示するメッセージ

sptuspv では，表 3-1 に示すメッセージがコンソールに表示されます。コンソールがない状態で

実行する場合などは，実行時のオペランドに標準出力装置を割り当て，実行後に割り当てたファイルを参照してエラー内容を確認します。メッセージはファイルの変換単位で表示されます。実行環境ファイル，実行環境構文ファイルにエラーがある場合は，そのファイルの変換を中止し，次のファイルを変換します。

表 3-1 Script 実行環境ファイルコンバータの戻り値とメッセージ

項番	戻り値	戻り値の意味	メッセージ ID 処理	メッセージ
1	0	正常に終了しました（ワイルドカード指定時，対象ファイルすべての変換が正常に終了しました）。	KBSC5100-I	ファイルを変換しました。（*変換されたファイル名*）
2	1	実行環境構文ファイルの構文に誤りがあります。	KBSC5101-E	実行環境構文ファイルの構文に誤りがあります。（*エラーとなったファイル名*）
3	2	範囲外の数値が指定されています。	KBSC5102-E	実行環境構文ファイル中に範囲外の値が指定されています。（*エラーとなったファイル名*）
4	3	ファイル形式に誤りがあります。	KBSC5103-E	ファイルが壊れているかまたはファイルの形式が異なります。（*エラーとなったファイル名*）
5	－	出力先フォルダに変換するファイルがすでに存在します。	KBSC5104-I	変換後のファイルが出力先ディレクトリに既に存在しました。このファイルの内容はすべて失われました。（*上書きされたファイル名*）
6	4	入力ファイルが存在しません。	KBSC5105-E	指定されたファイルが見つかりません。（*エラーとなったファイル名*）
7	5	入力ファイルアクセスエラー。	KBSC5106-E	システムエラーメッセージ。（*エラーとなったファイル名*）
8	20	出力先ディレクトリが見つかりません。	KBSC5107-E	出力先のディレクトリが見つかりません。（*エラーとなったディレクトリ名*）
9	21	出力ファイルアクセスエラー。	KBSC5108-E	システムエラーメッセージ。（*エラーとなったファイル名*）
10	22	SPV，SPU 以外のファイルを指定しました。	KBSC5109-E	指定できないファイルです。（*エラーとなったファイル名*）
11	23	実行時のコマンドラインに変換元ファイル名が指定されていません。または，コマンドラインに指定した内容が不正です。	KBSC5110-E	パラメタの指定が不正です。

12	24	オペランドが不正です。	KBSC5111-E	指定したパラメタの組み合わせに誤りがあります。
13	25	メモリが不足しています。	KBSC5112-E	メモリが不足しました。
14	26	対象 SPT ファイルが存在しません。	KBSC5113-W	出力先のディレクトリに対象となるスクリプトファイルが存在しませんでした。
15	27	実行環境構文ファイル中のディレクトリの指定が不正です。	KBSC5114-E	実行環境構文ファイル中に指定しているディレクトリが見つかりません。
16	28	上書き時の障害でファイルを削除しました。	KBSC5115-I	• 既に存在するファイルに対して書き込みエラーが発生したため，ファイルを削除しました。 • 変更後のファイルが出力先のディレクトリに存在しましたが，書き込みエラーが発生したため削除しました。
17	99	一部変換が失敗しました。※	ー	ー

注※

入力ファイルのワイルドカード指定時，項番 2，3，4，6，7 が発生した場合に返ります。
そのファイルの変換は行われなかったが，そのほかの対象ファイルの変換は行った場合に返ります。
項番 2，3，4，6，7 以外のエラー時は，その時点で処理が中断されます。

3.2 実行環境構文ファイル(.SPU)

実行環境構文ファイル(.SPU)とは，実行環境ファイル(.SPV)の内容をテキスト形式で記述したファイルです。このファイルは，実行環境をテキストエディタから設定することを目的としています。

(1) 実行環境構文ファイル(.SPU)の形式

実行環境構文ファイル(.SPU)の形式を次の図に示します。

図 3-5 実行環境構文ファイル(.SPU)の形式

```
Start_CommandLine=-SPT:SPXLV=2 "日立 太郎"  ･･･1.
Start_Work_Dir=                             ･･･2.
Trace_Dir=                                  ･･･3.
Trace_MaxLines=1024                         ･･･4.
Trace_MaxColumns=128                        ･･･5.
Trace_User_MaxLines=1024                    ･･･6.
Trace_User_MaxColumns=128                   ･･･7.
Start_MultiRun=1                            ･･･8.
```

図の説明

1. コマンドライン
   スクリプトファイルを実行するためのコマンドラインを指定します。
2. 作業フォルダ
   実行環境構文ファイル(.SPU)から実行環境ファイル(.SPV)に変換した場合は，この値は有効とします。存在しないフォルダ名が設定されていると実行環境ファイル(.SPV)変換時にエラーになり，実行環境ファイル(.SPV)は作成されません。
3. トレース出力先フォルダ
   実行環境構文ファイル(.SPU)から実行環境ファイル(.SPV)に変換した場合は，この値は有効とします。存在しないフォルダ名が設定されていると実行環境ファイル(.SPV)変換時にエラーになり，実行環境ファイル(.SPV)は作成されません。
4. トレース最大行数
   指定できる値は，100 から 9,999 までです。範囲外の値が指定されている場合は，実行環境ファイル(.SPV)変換時にエラーになり，実行環境ファイル(.SPV)は作成されません。
5. トレース最大列数
   指定できる値は，128 から 1,024 までです。範囲外の値が指定されている場合は，実行環境ファイル(.SPV)変換時にエラーになり，実行環境ファイル(.SPV)は作成されません。
6. ユーザトレース最大行数
   指定できる値は，100 から 9,999 までです。範囲外の値が指定されている場合は，実行

環境ファイル(.SPV)変換時にエラーになり，実行環境ファイル(.SPV)は作成されません。

7. ユーザトレース最大列数
   指定できる値は，128 から 1,024 までです。範囲外の値が指定されている場合は，実行環境ファイル(.SPV)変換時にエラーになり，実行環境ファイル(.SPV)は作成されません。
8. 複数起動の許可
   指定できる値は 0(複数起動の禁止)，または 1(複数起動の許可)です。これら以外の値が指定されている場合は，実行環境ファイル(.SPV)変換時にエラーになり，実行環境ファイル(.SPV)は作成されません。
   なお，同時実行数の上限はありません。しかし，同時実行数が多くなれば，その分システムリソースを消費します。そのため，同時実行するスクリプト数に注意してください。

(2) 実行環境構文ファイル(.SPU)の記述規則

実行環境構文ファイル(.SPU)の記述規則は，次のとおりです。

すべての項目は記述を省略できます。記述を省略した項目には，デフォルト値が設定されます。
同じ項目を二つ以上記述した場合は，後に記述されている値が有効になります。
指定項目を記述する順序は，任意です。
行の先頭に「#」を記述することで，その行をコメントとして扱えます。
#を含む文字列を指定する場合は，文字列の前後に引用符(")を指定します。
n 個の引用符(")を文字としてそのまま表す場合は，文字列中に(n×2)個の引用符(")を指定します。
コマンドラインオプションに全角の文字列を指定する場合，スクリプトの実行時に指定する言語と同じコードで指定してください。

(3) その他

実行環境構文ファイルのひな型は「/opt/jp1script/sample/spu」に格納されています。

3.3 Windows 上で作成したスクリプトファイル(.SPT)を UNIX 上に移行する際の注意事項

Windows 上で作成したスクリプトファイル(.SPT)中に全角の変数名，および全角の文字列を使用している場合は，移行先の UNIX の日本語環境に合わせてスクリプトファイルをコード変換する必要があります。使用するファイル転送ユティリティにコード変換機能が付いている場合は転送時に変換できます。ファイル転送ユティリティにコード変換機能がない場合は UNIX 上に転送した後に iconv コマンドを使ってコード変換を行ってください。Linux を対象に，iconv コマンドを起動して Shift-JIS から UTF-8 のコード変換を行うスクリプトの例を下記に示します。iconv コマンドで指定するコードセットの値はプラットフォームによって異なりますので，詳細は各プラットフォームのオンラインマニュアルなどを参照してください。

```
dim RTNCD
dim FROMDIR
dim TODIR
dim CONVFILE
FROMDIR = "/home/SJIS_SPT/"
TODIR = "/home/UTF8_SPT/"
For CONVFILE = FROMDIR + "*.SPT" Do
  SetStdFile(TODIR + CONVFILE,StdOutput,Create)
  RTNCD = Exec("/usr/bin/iconv",True,"-f","SHIFT-JIS","-t","UTF-8",FROMDIR +
CONVFILE)
  if RTNCD = FALSE then
    Message(Target_File,_SCF_FIL_ + ".TXT", "Exec Error : ConvFile=" +
CONVFILE)
    Exit(1)
  else
    if _EXEC_RTN_ = 0 then
      Message(Target_File,_SCF_FIL_ + ".TXT", "iconv normal end : ConvFile="
+ CONVFILE)
    else
      Message(Target_File,_SCF_FIL_ + ".TXT", "iconv error end : ConvFile=" +
CONVFILE)
    End if
  End if
  ResetStandardFile(StdOutput)
End For
```

4. JP1/Script の規則

この章では，スクリプトに関する規則，およびコマンドラインに関する規則について説明します。

4.1 スクリプトに関する規則

コマンドを使用してスクリプトを作成する場合の，次の規則について説明します。
変数
定数
数字の記述規則
文字列の記述規則
演算規則
コーディング規則
終了コード

4.1.1 変数

(1) 変数名の付け方

変数名の長さは 32 バイト以内で指定します。全部を半角文字で指定した場合は 32 文字以内，全部を全角文字で指定した場合は 16 文字以内で指定します。指定できる文字数以上を指定しても，指定できる文字数以降の文字は無視されます。
変数名の先頭 1 文字は半角の英字（A～Z および a～z），または全角文字で指定します。全角文字は，全角の英字（A～Z および a～z），数字（0～9），記号，かたかな，ひらがな，および漢字です。なお日本語 EUC，および UTF-8 では半角のかなも全角文字として扱います。
変数名には，半角の英字（A～Z および a～z），数字（0～9），アンダスコア（_），および全角文字を指定できます。なお日本語 EUC，および UTF-8 では半角のかなも全角文字として扱います。
変数名は 2 行以上にわたって記述できません。
変数名に指定する半角文字は，大文字と小文字が区別されません。全角文字は大文字と小文字が区別されます。

- 変数名にはキーワード（JP1/Script の予約語）を使用できません。また，キーワードは大文字と小文字が区別されません。キーワードの詳細は，「4.1.1 (3) 変数名として扱えないキーワード」を参照してください。

変数宣言をした適用範囲内で，同じ変数名を複数使用できます。

- 変数名には，プロシージャ※名と同じ名前を指定できません。同じ名前を指定した場合は，プロシージャ名が優先されます。

注※
プロシージャとは，実行時に一つの単位として処理されるコマンドの集まりのことをいいます。JP1/Script では，Function ステートメントと Sub ステートメントで定義された一連の処理がこれに該当します。サブルーチンと同意語です。Function ステートメントと Sub ステートメントの詳細は，「5.2 ステートメントの詳細」の「Function」，および「Sub」を参照してください。

(2) 使用できる変数の個数と容量

一つのプロシージャで使用できる変数は，最大 1,024 個です。

プロシージャの外部で使用できる変数（予約変数および位置変数を除く）は，最大 1,024 個です。

変数に格納できる容量は，最大 1,024 バイトです。1,024 バイト以降のデータは無視されます。

(3) 変数名として扱えないキーワード

変数名として扱えないキーワード（JP1/Script の予約語）を，表 4-1 に示します。

表 4-1 変数名として扱えないキーワード

見出し	変数名として扱えないキーワード
A	Abort※，AbortAll※，AbortRetryIgnore※，Abs※，AddStr，After，Alert※，Alertness※，AllDq，AllGV，Alt※，And，Anyway，Append，AppliModal，ApplicationModal，Array※，Asc※，AscB※，AscW※，Atn※，ATTR_ARCHIVE，ATTR_COMPRESSED，ATTR_HIDDEN，ATTR_NORMAL，ATTR_OFF，ATTR_OFFLINE，ATTR_ON，ATTR_READONLY，ATTR_SUBDIR，ATTR_SYSTEM，ATTR_TEMPORARY，AuditFailure，AuditSuccess
B	Backup，Beep，Before，BitmapHide※，BitmapShow※，Boolean※，ByRef※，Byte， ByVal※
C	CalcDate，CalcTime，Call，CallDll※，CallSpt※，Cancel※，CancelStartUp※，CancelUserErr，Case，CatFiles，CBool※，CByte※，CDate※，CDbl※，CDROM※，CheckDirName，CheckDriveType※，Chr※，ChrB※，ChrW※，Clnt※，Clear※，CLng※，Close※，Command※，CompDate，CompTime，Continue，Copy，CopyEX，CopyFile，Cos※，Create，CreateObject※，Critical※，CSng※，CStr※，Ctrl※，Ctrl_Alt※，Currency※，Cur_Desktop※，Cur_Program※，Cur_Startmenu※，Cur_Startup※
D	Date，DateSerial※，DateValue※，Day，DayU，Debug※，Delete，DeleteDir，DeleteFile，DeleteGroup※，DeleteGV，DeleteShortcut※，DependG※，DependM※，Description※，Destroy，Dim，DISABLE，DispName※，Do，Double※
E	Each※，Else，ElseIf，Emergence※，Emergency※，Empty，End，Enter※，EntryStartUp※，Eqv※，Equal，Erase※，Err，Errctl※，Error，ErrSkip，ErrSkip2，Esc※，Ex，Exclamation，ExclDir，Exec，EXEC_RUNNING※，EXEC_STOPPED※，Exit※，ExitWindows※，Exp※，Expand，Explicit※
F	False，FileTime，Fix※，FIXED※，For，Force※，Format，FreeExt，Function，F1※，F2※，F3※，F4※，F5※，F6※，F7※，F8※，F9※，F10※，F11※，F12※
G	GetArrayCount，GetDateCount，GetDiskFreeSpace，GetEnv，GetEnvironment，GetErrorMessage，GetExecStatus※，GetFileAttr※，GetFileAttribute※，GetFileSize，GetFileTime，GetGV，GetPath，GetProcessCount※，GetProcessInfo※，GetServiceName※，GetTextPosition，GetTimeCount，GetVerInfo※，GetVersionInfo※，GetVolLabel※，GetVolumeLabel※，GoTo，Group※

H	HelpContext※, HelpFile※, Hex※, HIDE, HKEY_CLASSES_ROOT※, HKEY_CURRENT_USER※, HKEY_LOCAL_MACHINE※, HKEY_USERS※, Hour, HourU,
I	IMEventMessage※, IniRead※, IniWrite※, InputBox※, InputLine, InStr, InStrB※, Int※, Integer※, Is※, IsArray※, IsDate※, IsDef, IsDefine, IsEmpty, IsEmptyDir, IsEmptyGroup※, IsEmptyReg※, IsExistDir, IsExistFile, IsExistRegKey※, IsExistService※, IsFileAttr※, IsFileAttribute※, IsLeapYear, IsLower, IsMultiChar, IsNew, IsNull※, IsNumeric, IsObject※, IsSingleChar, IsUpper, IsWriteableDir
J	JOBCancel※, JOBHold※, JOBSubmit※, JOBWait※, JP1Script※
K	KB
L	LBound※, LCase, Lcl_Desktop※, Lcl_Program※, Lcl_Startmenu※, Lcl_Startup※, Left, LeftB※, Len, LenB※, Log※, Logoff※, Logon※, LOGON_FAILED※, Long※, Loop, LTrim
M	MakeDir, MakeGroup※, MakePath, MakeShortcut※, Max※, MB, Menu※, Message, MessageBox※, MessageEventLog※, Mid, MidB※, Min※, Minus, Minute, MinuteU, Mod, Mod=, Modify, Month, MonthU, Move
N	Name※, NeedDq, NetExec※, NetExecEX, Next, No, NoExec※, None, NoOverwrite, NOPREFIX, NoReplace, Normal※, Not, NotEqual, Nothing※, Notice, Now※, Null※, Number※
O	Object※, Oct※, OK※, OKCancel※, On, Option※, Or, Overwrite, OverwriteOnly,
P	Password※, Path※, Pause, Pile, Plus, Poweroff※, Preserve※, Private※, ProcessEnv, Public※
Q	Question
R	Raise※, RAMDISK※, Randomize※, ReadOnly, ReadWrite, Reboot※, ReDim※, REG_BINARY※, RegDelete※, RegDeleteKey※, REG_DWORD※, REG_DWORD_BIG_ENDIAN※, REG_EXPAND_SZ※, REG_LINK※, REG_MULTI_SZ※, REG_NONE※, RegRead※, REG_RESOURCE_LIST※, REG_SZ※, RegWrite※, Rem, REMOTE※, REMOVABLE※, Release, Remove, Rename, Replace, ResetStandardFile, ResetStdFile, Restart, Resume※, Retry, RetryCancel, Right, RightB※, Rnd※, RTrim
S	Second, SecondU, Security, Select, SeparateStr, SeparateStrCount, Service, SERVICE_AUTO_START※, SERVICE_BOOT_START※, ServiceChange※, ServiceContinue※, SERVICE_CONTINUE_PENDING※, ServiceControl※, SERVICE_CONTROL_CONTINUE※, SERVICE_CONTROL_PAUSE※, SERVICE_CONTROL_STOP※, ServiceCreate※, ServiceDelete※, SERVICE_DEMAND_START※, SERVICE_DISABLED※, SERVICE_ERROR_CRITICAL※, SERVICE_ERROR_IGNORE※,

	SERVICE_ERROR_NORMAL ※, SERVICE_ERROR_SEVERE ※, SERVICE_FILE_SYSTEM_DRIVER ※, ServiceGetValue ※, SERVICE_KERNEL_DRIVER ※, ServicePause ※, SERVICE_PAUSE ※, SERVICE_PAUSE_PENDING ※, ServiceQuery ※, ServiceRefer ※, SERVICE_RUNNING ※, ServiceSetValue ※, ServiceStart ※, SERVICE_START_PENDING ※, ServiceStop ※, SERVICE_STOPPED ※, SERVICE_STOP_PENDING ※, SERVICE_SYSTEM_START ※, SERVICE_WIN32_OWN_PROCESS※, SERVICE_WIN32_SHARE_PROCESS※, Set※, SetEnv, SetEnvironment, SetFileAttr※, SetFileAttribute※, SetFileTime, SetGV, SetPath, SetStandardFile, SetStdFile, SetVolLabel※, SetVolumeLabel※, Shift※, Shift_Alt※, Shift_Ctrl※, Shift_Ctrl_Alt※, Shutdown※, Sgn※, Sin※, Single※, Skip, Sleep, Source※, Space, SplitFile, SplitPath, Sqr※, Start※, StartName※, StdError, StdInput, StdOutput, Step, Stop, Str, StrComp※, String, StringJ, Sub, SubDirToo, Submit, Syntax※, Sysmodal※, SystemEnv, SystemModal,
T	Tan※, Target_DispClear, Target_DispOff, Target_DispOn, Target_File, Target_SPAFile, Target_SPXFile, TempDir, TempFile, TerminateProcess※, TextClose, TextFileReplace, TextOnly, TextOpen, TextRead, TextSeek, TextWrite, Then, Time, Timeout※, TimeSerial※, TimeValue※, To, ToBegin, ToEnd, Trace, Trim, True, Twice, Type, TypeOf※
U	UBound※, UCase, UnSubmit, Until, Update, UserEnv, UserErr
V	Val※, Variant※, VarType※, Version, VersionUp
W	Wait※, WaitAll※, WaitForExec※, Warning, Weekday, Wend※, While, WriteOnly
X	Xor※
Y	Year, Yes, YesNo, YesNoCancel
記号※	^, ^=, -, -=, *, *=, /, /=, ¥, ¥=, +, +=, &, &=, ?, =, <>, <, >, <=, >=

注※

これらのキーワードは，今後，キーワードとしてサポートしていく予定です。すでにキーワードとして確保していますので，変数名として扱えません。

(4) 予約変数

JP1/Script には，特定のデータ（システム情報，およびコマンドの戻り値）を参照できる**予約変数**があります。予約変数の先頭には，アンダスコア（_）を付けます。

予約変数を表 4-2 に示します。

表 4-2　予約変数

分類	予約変数	意味
システム予約変数	_BIN_	実行時の起動ディレクトリ名です。末尾に/が付きます。
	COMP	現在のシステムにログインしているコンピュータ名です。
	SCF	実行中のスクリプトファイルのディレクトリ名です。末尾に/が付きます。
	_SCF_FIL_	実行中のスクリプトファイルのファイル名です。拡張子は付きません。
	_SCF_EXT_	スクリプトファイルの拡張子（.SPT）です。
	_SVF_EXT_	実行環境ファイルの拡張子（.SPV）です。
	TEMP	一時ファイル用のディレクトリ名です。末尾に/が付きます。
	USER	現在のシステムにログインしているユーザ名です。
プロセス予約変数	_PROC_ID_	実行中のスクリプトのプロセス識別子です。
	ARGV	%1 以降の位置変数を格納した配列変数です。%0 は含みません。_ARGV_(n)（n は 1 以上の数字）の形式で参照できます。
	_ARGV_CNT_	%1 以降の位置変数の合計数です。%0 は含みません。
コマンド戻り値予約変数	_COPY_RTN_	Copy コマンドまたはコマンドの実行結果です。
	_COPY_CNT_	Copy コマンドまたはコマンドでコピーしたファイルの数です。
	_COPY_SKIP_CNT_	Copy コマンドまたはコマンドでコピーされなかったファイルの数です。
	_COPY_SKIP2_CNT_	Copy コマンドまたはコマンドの ErrSkip2 指定で無視されたファイルの数です。
	_EXEC_RTN_	Exec，およびコマンドの戻り値です。符号付きの数値（0～255）です。
	RTN	エラー詳細コードです。符号付きの数値です。
文字コード予約変数	_NL_	改行文字です。
	TAB	タブ文字です。
エラー詳細コード予約変数	_NO_ERR_	エラーはありません。
	_ERR_EOF_	ファイルの終わりに達しました。
	_ERR_TIMEOUT_	タイムアウト時間を経過しました。
	_ERR_FILE_	ファイルが見つかりません。

	_ERR_ACCESS_	アクセスが拒否されました。
	_ERR_READY_	デバイスが準備できていません。
	_ERR_EXCLUSIVE_	ファイルはほかでアクセス中です。
	_ERR_FILE_SIZE_	取得した値が変数の上限値を超えているため，値を変数に格納できません。
	_ERR_NOT_LARGE_FILE_	指定したファイルの容量が制限値を超えています。
	_ERR_FILE_POSITION_	読み書き開始位置が 2,147,483,647 を超えています。

(5) 配列変数

JP1/Script には，データの順をインデックス番号で指定する配列変数があります。

配列変数には一次元配列変数と二次元配列変数があります。

一次元配列変数の場合は，指定するインデックス番号は要素を表します。

二次元配列変数の場合は，一番目に指定するインデックス番号は行要素を，二番目に指定するインデックス番号は列要素を表します。二次元配列変数「T(5，6)」のデータ構造例を図 4-1 に示します。

図 4-1 二次元配列変数「T(5，6)」のデータ構造例

T(1,1)	T(1,2)	T(1,3)	T(1,4)	T(1,5)	T(1,6)
T(2,1)	T(2,2)	T(2,3)	T(2,4)	T(2,5)	T(2,6)
T(3,1)	T(3,2)	T(3,3)	T(3,4)	T(3,5)	T(3,6)
T(4,1)	T(4,2)	T(4,3)	T(4,4)	T(4,5)	T(4,6)
T(5,1)	T(5,2)	T(5,3)	T(5,4)	T(5,5)	T(5,6)

(a) 配列変数の記述規則

配列変数名は，変数名の付け方の規則に従って設定します。変数名の付け方の詳細は，「4.1.1(1) 変数名の付け方」を参照してください。

配列変数の要素（二次元配列の場合は，行要素および列要素）を表すインデックス番号は 1 から始まります。( )または[ ]で囲んで指定します。

配列変数の宣言時にインデックス番号を指定する場合は，配列変数の要素数は固定となります。インデックス番号を指定しない場合は，配列変数の要素数は可変となります。

配列変数の次元数は，一次元，および二次元まで定義できます。

配列変数の最大要素数は，一つの配列変数につき 256 個，一つのスクリプトファイルにつき 512 個です。

一度宣言した配列変数の次元数は変更できません。

（例）

Dim A(5)

Dim A(5,10) ←（エラー）次元数は変更できない。

一度宣言した配列変数の次元数が等しい場合は，要素数を変更できます。また，要素数が固定な配列変数から可変な配列変数へ，または可変な配列変数から固定な配列変数へ変更できます。変更した場合，今までの設定値は Empty 値となります。

（例）

Dim A(10)

Dim A(5) ←要素数を変更できる。

Dim A()　←可変な要素数に変更できる。

非配列変数から配列変数へ，または配列変数から非配列変数への再定義はできません。

（例）

Dim A

Dim A() ←（エラー）A は非配列変数として定義されているため再定義できない。

代入文で配列変数のすべての要素を一括して代入できます。ただし，左辺と右辺の配列変数の要素数が異なる場合，一括して代入することはできません。

（例）

Dim A(5),B(),C(10)

For cnt = 1 To 5

A(cnt) = Time

Next

B = A ←A のすべての要素を一括して B に代入できる。

C = B ←（エラー）B と C の要素数が異なるため一括して代入できない。

代入文で配列変数の一つの行要素だけを一括して代入できます。ただし，一つの列要素だけを一括して代入できません。

（例）

Dim A(2,5),B()

For cnt = 1 To 5

A(1,cnt) = Time

Next

B = A(1) ←A の第 1 行目の要素を一括して B に代入できる。

比較文で配列変数のすべての要素が等しいかどうかを判定できます。ただし，大小比較はできません。

（例 1）

Dim A(5),B(5)

:

If A = B Then ←A と B が等しいかどうかを比較できる。

（例 2）

Dim A(5),B(5)

:

If A < B Then ←（エラー）A と B の大小比較はできない。

要素数が可変な配列変数で値が未設定な中間の要素に対し代入をした場合，先頭から中間の要素までの未設定であった要素は Empty 値になります。

（例 1）

Dim A()

A(1) = Time

A(5) = Time ← A(2) から A(4) までの要素は Empty 値になる。

（例 2）

Dim A(,)

A(2,5) = Time ← A(1, 1)と A(2,1)か A(2,4) の要素は Empty 値になる。

（例 3）

Dim A(2,)

A(2,5) = Time ← A(2,1)から A(2,4)の要素は Empty 値になる。

（例 4）

Dim A(,5)

A(2,5) = Time ← A(1,1)から A(1,5)と A(2,1)から A(2,4)の要素は Empty 値になる。

(b) 配列変数のデータ構造例

一次元の配列変数の場合，インデックス番号は要素を表します。

（例）

Dim A(5)

Dim A(5)と宣言した場合，次の図のような五つ分の要素が割り当てられます。

第1要素	第2要素	第3要素	第4要素	第5要素
A(1)	A(2)	A(3)	A(4)	A(5)

二次元の配列変数の場合，1番目のインデックス番号は行要素を，2番目のインデックス番号は列要素を表します。

（例 1）

Dim B(1,5)

Dim B(1,5)と宣言した場合，次の図のような 1 行 5 列分の要素が割り当てられます。

	（第1列）	（第2列）	（第3列）	（第4列）	（第5列）
（第1行）	B(1, 1)	B(1, 2)	B(1, 3)	B(1, 4)	B(1, 5)

（例 2）

Dim C(3,4)

Dim C(3,4)と宣言した場合，次の図のような 3 行 4 列分の要素が割り当てられます。

	（第1列）	（第2列）	（第3列）	（第4列）
（第1行）	C(1, 1)	C(1, 2)	C(1, 3)	C(1, 4)
（第2行）	C(2, 1)	C(2, 2)	C(2, 3)	C(2, 4)
（第3行）	C(3, 1)	C(3, 2)	C(3, 3)	C(3, 4)

4.1.2 定数

JP1/Script の定数の種類と意味を，表 4-3 に示します。

表 4-3 定数一覧

定数	意味
Empty	空の値（""）を示します。
True	真（0 以外）の値を示します。
False	偽（0）の値を示します。

4.1.3 数字の記述規則

0，＋0，－0 は同値とみなします。
0 だけが連続している数字は，0 とみなします。
扱える数字は，整数だけです。
扱える数値は，-2,147,483,647～2,147,483,647 の範囲です。
コンマ（，）の有無は，数字を引用符（"）で囲んだ場合は区別されません。
（例）"10,000"と 10000 は同値とみなします。
数字の先頭の「+」の有無は区別されません。
（例）600 と+600 は同値とみなします。

- 数字を引用符（"）で囲んでも文字列としては扱われません。文字列として扱うには Str コマンドを使用してください。Str コマンドの詳細は，「6.3 文字列操作コマンド」の「Str（値を文字列で返す）」を参照してください。

  （例）"10000"と 10000 は同じ値とみなします。

4.1.4 文字列の記述規則

数値以外の文字列を記述する場合は引用符（"）で囲みます。
変数，予約語などを引用符（"）で囲むと文字列として扱われます。
扱える文字列の長さは最大 1,024 文字です。
何も囲んでいない引用符（"）は Empty 値とみなします。
引用符（"）で囲まれている数値にコンマ（，）以外の文字列が含まれると文字列として扱われます。

4.1.5 演算規則

(1) 規則

( )内の演算は，最も内側のものから行われます。

変数は，基本的に文字列型として扱われますが，数字だけで構成されている文字列同士の演算の場合は数字として扱われ，演算が行われます。

単項演算子は「＋」，「－」だけです。

単項演算子は乗算，および除算よりも優先されます。

(2) 演算の優先順位

演算の優先順位を表 4-4 に示します。

表 4-4 演算の優先順位

優先度	演算子	内容
↑高い	^	指数演算
	+，－	単項演算子
	*，/，￥，Mod	乗算，除算，整数除算，剰余演算
	+，－	加算，減算
	&	文字列連結
	=，>，<，<=，>=，<>	比較演算
	Not	論理否定
↓低い	And	論理積
	Or	論理和

4.1.6 コーディング規則

1 文（ステートメントを除く）が複数行にわたる場合は，アンダスコア（_）を文の最後に記述します。アンダスコアは，前に 1 個以上の半角スペースを入れてから記述します。なお，引数を伴うコマンド文で引数を括弧で囲んでいる場合は，アンダスコアを記述する必要はありません。

（例）引数を括弧で囲む場合

```
Command (A,B,C,
D,E,F)
```

（例）引数を括弧で囲まない場合

```
Command A,B,C, _
```

D,E,F

コマンド名を，複数行にわたって記述することはできません。
コマンドの大文字と小文字は区別されません。
ファイル名，および環境変数名の大文字と小文字は区別されます。
1 行に記述できるスクリプトは最大 10,240 バイトです。10,240 バイトを超えた部分のスクリプトは無視されます。
引用符（"）内に記述できる文字列は半角文字で 1,024 文字以内，全角文字で 512 文字以内です。それ以降の文字は無視されます。
コマンドの引数の記述は，次に示す形式のどれかで記述します。引数が複数個ある場合は，コンマ（，）または半角スペースで区切ります。

引数の記述形式（△：半角スペース）

Command(A)

Command△(A)

Command△A

Command(A,B)

Command△(A,B)

Command△A△B

Command△A,B

コマンドまたはプロシージャの引数に，引数を伴うコマンドまたはプロシージャを記述する場合は，引数として記述したコマンドまたはプロシージャの引数を，括弧で囲む必要があります。

（例）必ず括弧で囲む場合の記述

Command(Function(A,B,C),D,E)

Command Function(A,B,C),D,E

演算子の左辺または右辺に，引数を伴うコマンドまたはプロシージャを記述する場合は，引数を括弧で囲む必要があります。

（例）

M=Command(A,B,C)＋Function(D)

- スクリプトを実行させるスクリプトエンジンのバージョン※を指定する場合は，各スクリプトファイルの先頭に，#FileVersion=VVRR（VV：JP1/Script のバージョン，RR：JP1/Script のリビジョン）と記述する必要があります。この記述がない場合，または存在しないバージョンが指定された場合は，#FileVersion=0700 を仮定して，スクリプトが実行されます。

注※

スクリプトを実行させるスクリプトエンジンのバージョンは，JP1/Script のインストールバージョンと同一ではありません。

（例）

#FileVersion = 0700

GoTo ステートメントや On Error ステートメントなどの制御の分岐先は，ラベル名の直後にコロン（:）を付けて指定する必要があります。

（例）

LabelName:

文字列中に含まれる"¥r"，"¥n"，"¥t"，"¥¥"をそれぞれ対応するコントロールコードに変換しない場合は，スクリプトファイルの先頭に，#Option = NOCHANGE と記述する必要があります（ただし，#FileVersion = VVRR が指定されている場合はその下の行に記述します）。この記述がない場合は，文字列中に含まれる"¥r"，"¥n"，"¥t"，"¥¥"はそれぞれに対応するコントロールコードに変換されます。"¥r"は復帰，"¥n"は改行と復帰，"¥t"はタブ，"¥¥"は一つの"¥"に変換されます。ただし，コンソール画面では変換されません。

（例）

#FileVersion = 0700

#Option = NOCHANGE

Message(Target_File,"TESTFILE","¥¥1,000")

[TESTFILE に出力される文字列]

¥¥1,000

（例）

#FileVersion = 0700

Message(Target_File,"TESTFILE","¥¥1,000")

[TESTFILE に出力される文字列]

¥1,000

n 個の引用符（"）を文字としてそのまま表す場合は，文字列中に（n×2）個の引用符（"）を指定します。

（例）

Message (Target_DispOn,”display”,"Error code: ""99""")

[コンソール画面に出力される文字列]

Error code: "99"

4.1.7 終了コード

スクリプト実行の終了コードは，Exit コマンドで設定します。Exit コマンドまたはこのコマンドの引数を省略した場合，スクリプト実行の終了コードには「0」が設定されます。Exit コマンドの詳細は，「6.11 その他のコマンド」の「Exit（スクリプトを終了する）」を参照してください。
ただし，スクリプト実行がエラーになった場合は，**JP1/Script の終了コード**を返すことがあります。これらの値は，JP1/Script のインストールディレクトリにあるシステム環境ファイル（jp1script.cf）に設定されています。システム環境ファイルに設定されている JP1/Script の終了コードを表 4-5 に示します。

表 4-5 システム環境ファイルに設定されている JP1/Script の終了コード

値	デフォルト値	意味
AlreadyRun	16	指定されたスクリプトファイルはすでに起動されています。
GrammarError	19	文法エラーが発生しました。
ExAborterror	20	JP1/Script のプロセスを中断する実行エラーが発生しました（メモリ不足，未定義の変数を参照しているエラー，プロシージャが見つからないエラー，ステートメントで発生するエラーなど）。
Error	99	JP1/Script のプロセス開始前にエラーが発生しました（指定されたスクリプトファイルが見つからないエラーなど）。
	21 ～ 32	予備

注意事項

Exit コマンドで終了コードを設定する場合は，JP1/Script の終了コードと重複しないように注意してください。

4.2 コマンドラインに関する規則

スクリプトファイルを実行する場合の，コマンドラインの指定方法について説明します。

4.2.1 コマンドラインの形式

コマンドラインの形式について次に示します。形式中の△は半角スペースを示します。指定したスクリプトファイルの先頭行に「#!△/opt/jp1script/bin/sptxe」の記述がある場合，（1），および（3）の先頭の「sptxe」を省略できます。なお，位置変数については「4.2.2 コマンドラインのパラメタの説明」を参照してください。

(1) コンソール画面から直接実行形式（sptxe）を指定する場合

sptxe△スクリプトファイル名〔△位置変数〕〔△-SPT:SPALV=n〕〔△-SPT:SPXLV=n〕〔△-SPT:NOSYSLOG=n,n,…〕〔△-SPT:GRM〕〔△-SPT:NODSP〕

(2) Exec コマンドでパラメタを指定してスクリプトファイルを呼び出す場合

Exec(スクリプトファイル名,終了待ちの有無〔，"位置変数"〕〔,"-SPT:SPALV=n"〕〔,"-SPT:SPXLV=n"〕〔,"-SPT:NOSYSLOG=n,n,…"〕〔，"-SPT:GRM"〕〔，"-SPT:NODSP"〕)

(3) ユーザプログラムなどから実行形式（sptxe）でコマンドラインを指定してスクリプトファイルを実行する場合

sptxe△スクリプトファイル名〔△位置変数〕〔△-SPT:SPALV=n〕〔△-SPT:SPXLV=n〕〔△-SPT:NOSYSLOG=n,n,…"〕〔△-SPT:GRM〕〔△-SPT:NODSP〕

この場合，スクリプトファイル名はコマンドラインの先頭に指定します。

4.2.2 コマンドラインのパラメタの説明

(1) 位置変数

スクリプト実行時に指定されたコマンドラインを参照できる定数として，位置変数があります。位置変数は%n（n は正の整数）で表され，%0 はスクリプトファイル名，%1 は第 1 位置変数（%n は第 n 位置変数）となります。

複数の位置変数を指定する場合は，コマンドラインの記述規則に従ってください。コマンドライン

の記述規則については，「4.2.3 コマンドラインの記述規則」を参照してください。

コマンドラインの形式例を次に示します。△は半角スペースを示します。

（例 1）

コンソール画面から直接実行形式（sptxe）を指定する場合

[コマンドラインの指定形式]

sptxe△スクリプトファイル名△ABC△123△"/WORK/"

[位置変数に指定される値]

%0：スクリプトファイル名

%1：ABC

%2：123

%3：/WORK/

（例 2）

Exec コマンドからパラメタを指定してスクリプトファイルを呼び出す場合

[コマンドの指定形式]

Exec (スクリプトファイル名, true, "ABC", 123, "/WORK/")

[呼び出されるスクリプトファイルの位置変数]

%0：スクリプトファイル名

%1：ABC

%2：123

%3：/WORK/

（例 3）

ユーザプログラムなどから実行形式（sptxe）でコマンドラインを指定してスクリプトファイルを実行する場合

[ユーザプログラムの指定形式]

sptxe△スクリプトファイル名△ABC△123△"/WORK/"

[位置変数に指定される値]

%0：スクリプトファイル名

%1：ABC

%2：123

%3：/WORK/

(2) -SPT:SPALV=n（または-spt:spalv=n）

解析トレースファイルの出力の有無を指定します。指定を省略した場合は，解析トレースファイ

ルを出力します。n には 0 または 1 の整数を入力します。

n の値が 0 の場合：解析トレースファイルを出力しません。

n の値が 1 の場合：解析トレースファイルを出力します。
上記以外の値を指定した場合はエラーとなり，スクリプト実行を中断します。

(3) -SPT:SPXLV=n（または-spt:spxlv=n）
実行トレースファイルの出力の有無，および出力レベルを指定します。指定を省略した場合は，実行トレースファイルにエラー時の結果だけを出力します。n には 0～3 の整数を入力します。

n の値が 0 の場合：実行トレースファイルを出力しません。

n の値が 1 の場合：実行トレースファイルにエラー時の結果だけが出力されます。

n の値が 2 の場合：実行トレースファイルに正常時の結果も出力されます。

n の値が 3 の場合：実行トレースファイルに，コマンド開始時刻と終了時刻が出力されます。
上記以外の値を指定した場合はエラーとなり，スクリプト実行を中断します。

(4) -SPT:NOSYSLOG（または-spt:nosyslog）
syslog への出力を抑止する場合に指定します。このパラメタを省略すると，syslog にすべてのログが出力されます。

-SPT:NOSYSLOG または-spt:nosyslog

ログの種類がエラーの場合にだけログを出力します。

-SPT:NOSYSLOG=n,n,…または-spt:nosyslog=n,n,…

n で指定したログ ID は出力しません。n には複数のログ ID を順不同で指定できます。複数のログ ID をコンマで区切って指定します。

ログ ID の詳細は，「7.1.2 syslog ファイルから原因を調査する」を参照してください。

(5) -SPT:GRM（または-spt:grm）
文法チェックをする場合に指定します。

(6) -SPT:NODSP（または-spt:nodsp）
画面に出力されるメッセージを抑止する場合に指定します。

4.2.3 コマンドラインの記述規則

(1) コンソール画面から直接実行ファイル名（sptxe）を指定する場合の記述規則
実行形式は小文字で sptxe と指定します。また，スクリプトファイルの拡張子を省略した場合は，".SPT"（大文字）を仮定します。

複数のパラメタを指定する場合は，半角スペースで区切ります。

区切りではない半角スペースを含むパラメタを指定する場合は，パラメタ全体を引用符（"）または¥"で囲む必要があります。パラメタに含ませる文字とその実行例を表 4-6 に示します。

表 4-6 パラメタに含ませる文字とその実行例

パラメタに含ませる文字	実行例	%1 **に渡る文字列**
半角スペース	sptxe△abc.SPT△"ABC△DEF"	ABC△DEF
	sptxe△abc.SPT△¥"ABC△DEF¥"	
引用符(')	sptxe△abc.SPT△"ABC'DEF"※1	ABC'DEF
	sptxe△abc.SPT△¥"ABC'DEF¥"※2	

(凡例)

△:半角スペース

注※1

位置変数を引用符(')で囲む場合で，文字列に引用符(')を含んだ値を指定する場合は，値の引用符(')の前に「'¥'」を付けます。

注※2

位置変数を¥"で囲む場合で，文字列に引用符(')(または引用符(''))を含んだ値を指定する場合は，引用符(')を¥'(引用符(")を¥")とします。

n 個の引用符(")を文字としてそのまま表す場合は，文字列中に(n×2)個の引用符(")を指定します。

一つのスクリプトファイルで使用できる位置変数は，最大 255 個(%254 まで)です。それ以降の指定は無視されます。

位置変数に格納できる文字数は半角文字で 1,024 文字以内です。それ以降のデータは無視されます。

"-SPT:SPALV=n"などのパラメタが同種で複数指定されている場合は，あとに指定されているパラメタが有効になります。

指定したパラメタの値はスクリプトファイル単位で有効です。実行コンピュータのすべてのスクリプトに対して有効にしたい場合は，システム環境ファイル(/opt/jp1script/conf/jp1script.cf)の CommandLine にパラメタの値を設定します。

注意

実行時のコマンドラインとシステム環境ファイルの両方にパラメタを指定している場合は，両方の指定が有効になります。ただし，コマンドラインに"-SPT:SPXLV=3"，システム環境ファイルに"-SPT:SPXLV=0"のように，同種で異なる指定が両方にある場合はコマンドラインの指定が優先されます。

位置変数，および-SPT:GRM は有効になりません。

- スクリプトファイルの先頭に「実行ファイル指定:#!△/opt/jp1script/bin/sptxe」を記述しておくことで，コンソール画面からの直接実行では，ファイル名(sptxe)の記述を省略できます。

(2) Exec コマンドでパラメタを指定してスクリプトファイル，実行可能ファイルおよびシェルスクリプトファイルを呼び出す場合の記述規則

複数のパラメタを指定する場合は，半角スペースで区切ります。

空白を含む文字列を渡す場合は，引用符（”）で空白を含む文字列を囲ってください。JP1/Script で引用符（”）を記述する場合は，引用符（”）を二つ連続して記述します。ただし，囲んだ引用符（”）は呼び出した実行可能ファイルまたはシェルスクリプトファイルには渡されないので注意してください。なお，Exec コマンドにスクリプトファイル(.SPT)を指定した場合は，囲んだ引用符（”）が呼び出したスクリプトファイルに渡されます。引用符（”）文字が不要な場合は，文字列操作コマンドによって（”）文字を削除する処理を組み込んでください。パラメタの指定方法と呼び出し先のファイルに渡される文字列の例を表 4-7 に示します。

表 4-7 パラメタの指定とその実行例

パラメタの指定方法	実行可能ファイル，シェルスクリプトファイルに渡る文字列	スクリプトファイルに渡るる文字列
“ABC"	ABC	ABC
“”“ABC""""	ABC	"ABC"
“”“A△B△C””””	A△B△C	“A△B△C”
“ABC""DEF"	ABC"DEF	ABC"DEF
“ABC'DEF"	ABC'DEF	ABC'DEF

（凡例）

△：半角スペース

- Exec コマンドで実行可能ファイル，およびシェルスクリプトファイルを指定する場合は，パラメタの「”」から「”△」（△は半角スペースを意味しています）までを一つのパラメタと認識します。文字列中に「”△」を含む文字列を渡すことはできません（引数が分解されます）ので一つの文字列中に「”△」を含めないでください。

一つのスクリプトファイルで使用できる位置変数は，最大 255 個（%254 まで）です。それ以降の指定は無視されます。

位置変数に格納できる文字数は半角文字で 1,024 文字以内です。それ以降のデータは無視されます。

"-SPT:SPALV=n"などのパラメタが同種で複数指定されている場合は，あとに指定されているパラメタが有効になります。
指定したパラメタの値はスクリプトファイル単位で有効です。実行コンピュータのすべてのスクリプトに対して有効にしたい場合は，システム環境ファイル（/opt/jp1script/conf/jp1script.cf）の

CommandLine にパラメタの値を指定します。

注意

実行時のコマンドラインとシステム環境ファイルの両方にパラメタを指定している場合は，両方の指定が有効になります。ただし，コマンドラインに"-SPT:SPXLV=3"，システム環境ファイルに"-SPT:SPXLV=0"のように，同種で異なる指定が両方にある場合はコマンドラインの指定が優先されます。

位置変数，および-SPT:GRM は有効になりません。

4.2.4 コマンドラインに対するエラー処理

コマンドラインの解析時，および実行時に次のエラー処理をします。
なお，正常に実行できた場合はコンソールには何も出力しません。

(1) コンソール画面から直接実行ファイル名（sptxe）を指定する場合

(a) 実行ファイル名（sptxe）だけを指定した場合

次のエラーメッセージと，使用方法を説明したメッセージをコンソールに出力します。

KBSC4006-E スクリプトファイルが指定されていません。

使用方法を説明したメッセージを図 4-2 に示します。

図 4-2 使用方法を説明したメッセージ

```
使用方法：sptxe スクリプトファイル名 ［位置変数］ ［オプション］

位置変数                  //位置変数に格納するパラメタを指定する
オプション
 -SPT:GRM                 //文法チェックだけ行う場合に指定する
 -SPT:SPALV=n             //解析トレースファイルの出力の有無を指定する
                          // n=0:出力しない
                          // n=1:出力する
 -SPT:SPXLV=n             //実行トレースファイルの出力の有無およびレベルを
                          //指定する
                          // n=0:出力しない
                          // n=1:エラー時の結果だけ出力する
                          // n=2:正常時の結果も出力する
                          // n=3:コマンド開始時刻と終了時刻も出力する
 -SPT:NOSYSLOG            //ログの種類がエラーの場合にだけsyslogに出力する
 -SPT:NOSYSLOG=n[,n…]     //nで指定したログIDはsyslogに出力しない
 -SPT:NODSP               //画面に出力されるメッセージを抑止する場合に指定する
```

(b) スクリプトファイル名とオプションにエラーがある場合

存在しないスクリプトファイル名を指定した場合
次のエラーメッセージをコンソールに出力します。

KBSC4007-E **指定されたスクリプトファイルが見つかりません。（エラーのスクリプトファイル名）**

スクリプトファイルの拡張子に SPT 以外を指定した場合

次のエラーメッセージをコンソールに出力します。

KBSC4008-E **指定されたスクリプトファイル名が不正です。（エラーのスクリプトファイル名）**

存在しないオプションを指定またはオプションの記述に誤りがある場合

次のエラーメッセージと，使用方法を説明したメッセージをコンソールに出力します。

KBSC4009-E **指定された値が無効です。（エラーのオプション）**

使用方法を説明したメッセージについては，図 4-2 を参照してください。

(c) コマンドラインの実行結果に対するエラー

スクリプトファイルの解析エラーの場合

次のエラーメッセージをコンソールに出力します。

KBSC4004-E **文法エラーが発生しました。（エラーのスクリプトファイル名）**

スクリプトファイルの実行エラーの場合

次のエラーメッセージをコンソールに出力します。

KBSC4005-E **実行エラーが発生しました。（エラーのスクリプトファイル名）**

なお，解析エラーと実行エラーが同一スクリプトファイルで発生した場合は，解析エラーのメッセージだけを出力します。

(2) Exec コマンドでパラメタを指定してスクリプトファイルを呼び出す場合，およびユーザプログラムなどから実行ファイル名（sptxe）でコマンドラインを指定してスクリプトファイルを実行する場合

(a) コマンドラインの実行結果に対するエラー

スクリプトファイルの解析エラーの場合

次のエラーメッセージをコンソールに出力します。

KBSC4002-E **文法エラーがありました。（エラーのスクリプトファイル名）**

スクリプトファイルの実行エラーの場合

次のエラーメッセージをコンソールに出力します。

KBSC40**05-E 実行エラーが発生しました。（エラーのスクリプトファイル名）**

なお，解析エラーと実行エラーが同一スクリプトファイルで発生した場合は，解析エラーのメッセージだけを出力します。

5. ステートメント

この章では，スクリプトを作成するときに使用できるステートメントについて説明します。

5.1 ステートメント一覧

スクリプトを作成するときに使用できる，ステートメントの一覧を表 5-1 に示します。

表 5-1 ステートメントの一覧

ステートメント	意味
=	右辺の値を左辺の変数に代入します。
Do...Loop	指定した条件が真（True）である間，または条件が真（True）になるまで，一連のステートメントを繰り返します。
For...Next	指定した回数だけ，一連のステートメントを繰り返します。
For...End For	指定したディレクトリ内のすべてのファイルに対して，一連のステートメントを繰り返します。
If...Then...Else	式の値に従って，条件を実行します。
Select Case	条件式の値に従って，複数のステートメントブロックのどれかを実行します。
While...End	指定した条件が真（True）である間，一連のステートメントの実行を繰り返します。
Function	Function プロシージャの名前，および引数を宣言し，Function プロシージャの始まりを示します。
Sub	Sub プロシージャの名前，および引数を宣言して，Sub プロシージャの始まりを示します。
Call	Sub プロシージャ，および Function プロシージャに制御を渡します。
Exit xx	Do...Loop ループ，For...End For ループ，For...Next ループ，While...End ループ，Function プロシージャ，または Sub プロシージャから抜け出します。
GoTo	指定したラベルに処理を分岐させます。
Continue	Do...Loop ループ，For...End For ループ，For...Next ループ，While...End ループの先頭に戻ります。
On Error	エラー発生時に，指定したラベルに制御を移します。エラー発生時に指定したラベルに制御を移さないようにすることもできます。

5.2 ステートメントの詳細

ステートメントの詳細を表 5-1 の順に説明します。

=

機能

右辺の値を左辺の変数に代入します。

形式

Result = Expression

指定項目

Result

値を受け取る変数名を指定します。

Expression

任意の式を指定します。

説明

式 Expression の値を変数 Result に代入します。

式 Expression の値が Empty 値のときは，変数 Result には Empty 値が代入されます。

Dim コマンドで宣言していない変数に値を代入した場合は，自動的に変数を割り当てて値を代入します。

例

```
' 変数 result1 には"ABCDE"が格納される。
result1 = "ABCDE"

' 変数 result2 には 200 が格納される。
result2 = 50 + 150

' 変数 result3 には現在の日付が格納される。
result3 = Date
```

Do...Loop

機能

指定した条件が真（True）である間，または条件が真（True）になるまで，一連のステートメントを繰り返します。

形式

Do

〔Statements〕

〔Exit Do〕

〔Statements〕

Loop〔{ While | Until } Condition〕

指定項目

Statements

Conditoin が真（True）である間，または Conditoin が真（True）になるまで繰り返し実行される一つ，または複数のステートメントを指定します。複数のステートメントを記述する場合は，ステートメントごとに改行します。

Condition

真（True），または偽（False）を評価する条件式を指定します。

説明

キーワード While が記述されている間は Condition が真（True）である間，一連のステートメントが繰り返し実行され，キーワード Until が記述されている場合は Condition が真（True）になるまで，一連のステートメントが繰り返し実行されます。

Exit Do ステートメントは，制御構造 Do ...Loop 内だけで使われ，Condition で指定した以外の条件で Do ...Loop を終了させることができます。Exit Do ステートメントは，Do ...Loop 内の任意の場所に何回でも指定できます。Exit Do は条件の評価（例えば If...Then ステートメントなど）とともに使われることが多く，Loop ステートメントの直後のステートメントに制御を渡します。

Do ...Loop ステートメントはネスト構造にできます。つまり，Do ...Loop の内部に別の Do ...Loop を記述できます。ネスト構造の場合に Exit Do が実行されると，その Exit Do を囲んでいる最も内側のループから抜け出します。

Do ...Loop ステートメントは While...End ステートメントとは異なり，一連のステートメントが必ず

一度は実行される後判定の制御構造となっています。一連のステートメントが実行される前に条件を判定したい場合は，While...End ステートメントを使用してください。

例

```
' Reverse は文字列を反転した値を返す。
' result には"EDCBA"が格納される。
result = Reverse ( "ABCDE" )
Message( Target_DispOn,"display",result )

Function    Reverse ( chrValue )
   Dim    chrString ,chrLength
   cnt = 0
   chrLength = Len ( chrValue )
   Do
         chrString = chrString + Mid ( chrValue ,chrLength - cnt , 1 )
         cnt = cnt + 1
   Loop While ( cnt < chrLength )
   Reverse = chrString
End Function
```

For...Next

機能

指定した回数だけ，一連のステートメントを繰り返します。

形式

For　Counter = Start　To　End　〔Step Step〕

　〔Statements〕

　〔Exit For〕

　〔Statements〕

Next

指定項目

Counter

ループカウンタに使う数値変数を指定します。この変数には，配列やユーザ定義型の要素を指定できません。

Start

Counter の初期値を指定します。

End

Counter の最終値を指定します。

Step

ループを繰り返すごとに *Counter* に加算される値を指定します。*Step* を省略すると，ループを繰り返すごとに *Counter* に 1 が加算されます。*Counter* には正の数，または負の数を指定できます。*Step* で指定した値によりループの実行は次のように制御されます。

値	実行条件
正の数または 0	Counter ≦ End
負の数	Counter ≧ End

Statements

ループ内で実行される一連のステートメントを指定します。For と Next の間にこれらのステートメントを記述します。ここに記述したステートメントは，For...Next で指定した回数だけ実行されます。*Statements* には複数の文を記述できます。複数の文を記述する場合は，文ごとに改行します。

説明

プログラムの実行がループに移り，ループ内の一連のステートメントがすべて実行されると，*Step* の値が *Counter* に加算されます。この時点で終了条件を満たしていない場合は，ループ内のステートメントが再び実行されます。終了条件が満たされると，ループから抜け出して Next ステートメントの次のステートメントに制御が移ります。

Exit For ステートメントは，制御構造 For...Next と For...End For ループ内だけで使用できます。Exit For ステートメントを使用すると，指定した回数以外の条件で For...Next ループを終了させることができます。Exit For ステートメントは，For...Next ループ内の任意の位置で何回でも指定できます。Exit For ステートメントは，条件の評価（例えば，If...Then ステートメントなど）とともに使われることが多く，Next の直後のステートメントに制御を渡します。

For...Next ループはネスト構造にすることができます。つまり，For...Next ループの内部に別の For...Next ループを入れることができます。ループをネストさせる場合は，それぞれの Counter に別の変数名を指定してください。

例

```
For　I = 1　To　10
    For　J = 1　To　10
      For　K = 1　To　10
        :
      Next
    Next
Next
```

For...End For

機能

指定したディレクトリ内のすべてのファイルに対して，一連のステートメントを繰り返します。

形式

```
For　VarName = Filemask　Do
　〔Statements〕
　〔Exit For〕
　〔Statements〕
End〔For〕
```

指定項目

VarName

ファイル名を格納する変数名を指定します。変数名は変数名の付け方の規則に従って指定します。変数名の付け方の詳細は，「4.1.1 (1) 変数名の付け方」を参照してください。

Filemask

ファイル名をフルパスで指定します。ファイル名にはワイルドカードを含めることもできます。フルパス名は，引用符（"）で囲みます。

Statements

ループ内で実行される一連のステートメントを指定します。For と End For の間にこれらのステートメントを記述します。ここに記述したステートメントは，For...End For で指定したパスに該当するファイルが見つからなくなるまで繰り返されます。

Statements には複数の文を記述できます。複数の文を記述する場合は，文ごとに改行します。

説明

指定したディレクトリ内のすべてのファイルに対して，一連のステートメントを繰り返します。指定したパスに該当するファイルが見つからなくなったとき，End For ステートメントの次のステートメントに制御が移ります。

Exit For ステートメントは，制御構造 For...End For と For...Next ループ内だけで使用できます。Exit For ステートメントを使用すると，直ちに For...End For ループを終了させることができます。Exit For ステートメントは，For...End For ループ内の任意の位置で何回でも指定できます。

For…End For ループのファイル検索順序は保証していません。

For...End For ループはネスト構造にすることができます。つまり，For...End For ループの内部に別の For...End For ループを入れることができます。ループをネストさせる場合は，それぞれの VarName に別の変数名を指定してください。

指定したディレクトリが存在しない場合，一連のステートメントを実行しないで，次のステートメント，またはコマンドを実行します。

filemask に "*"を指定した場合，varname には "."，".." 以外のファイル名，およびディレクトリ名が格納されます。

例

```
' 実行ディレクトリ下のスクリプトファイルを別ディレクトリにバックアップする。
Dim　path1 ,bkupDir
bkupDir = _BIN_+"BKUP/"
MakeDir ( bkupDir )

' スクリプトファイルを検索
For　path1 = _BIN_+"*.SPT"　Do
  ' ディレクトリは無視
  If　IsExistDir ( _BIN_+path1 ) = False　Then
    ' 別ディレクトリにバックアップ
    Copy ( _BIN_+path1 ,bkupDir+path1 )
  End If
End For
```

If...Then...Else

機能

式の値に従って，条件を実行します。

形式 1

If　Condition　Then

　〔Statements〕

〔Else

　〔ElseStatements〕〕

End〔If〕

形式 2

If　Condition　Then

　〔Statements〕

〔ElseIf　Condition-n　Then

　〔ElseifStatements〕〕...

〔Else

　〔ElseStatements〕〕

End〔If〕

指定項目

Condition

真（True）か偽（False）を評価する条件式です。

Statements

Condition が真（True）の場合，実行される文です。

Statements には複数の文を記述できます。複数の文を記述する場合は，文ごとに改行します。

Condition-n

真（True）か偽（False）を評価する条件式です。

ElseifStatements

Condition-n が真（True）の場合に実行される一連のステートメントを指定します。

ElseifStatements には複数の文を記述できます。複数の文を記述する場合は，文ごとに改行します。

ElseStatements

Else 以前に定義されている条件（Condition または Condition-n）がどれも真（True）でない場合に実行される一連のステートメントを指定します。

ElseStatements には複数の文を記述できます。複数の文を記述する場合は，文ごとに改行します。

説明

最初に Condition が評価されます。Condition が真（True）の場合，Then に続くステートメントが実行されます。Condition が偽（False）の場合，形式 1 では Else 節があるとき，Else 節が実行されます。形式 2 では ElseIf で指定した条件（Condition-n）が評価されます。どれかの条件が真（True）の場合，対応する Then に続くステートメントが実行されます。ElseIf で指定された条件式がどれも偽（False）の場合（または，ElseIf 節がない場合）は，Else に続くステートメントが実行されます。Then，または Else に続くステートメントの実行が終わると，End の次のステートメントからプログラムの実行が続けられます。

Else 節と ElseIf 節はどちらも必要に応じて定義します。また，形式 2 の If では，ElseIf 節は幾つでも指定できます。ただし，Else 節の後ろに ElseIf 節を指定することはできません。また，If はネスト構造にすることができます。

Select Case

機能

条件式の値に従って，複数のステートメントブロックのどれかを実行します。

形式

Select Case　TestExpression

〔Case　ExpressionList-n

　〔Statements-n〕〕...

〔Case Else

　〔ElseStatements-n〕〕

End〔Select〕

指定項目

TestExpression

任意の条件式を指定します。

ExpressionList-n

Case 節を記述する場合は必ず指定します。一つ，または複数の式をコンマ（，）で区切って指定します。

Statements-n

TestExpression が ExpressionList-n のどれかと一致した場合，一致した Statements-n のステートメントが実行されます。Statements-n には複数の文を記述できます。複数の文を記述する場合は，文ごとに改行します。

ElseStatements-n

TestExpression が Case 節のどれとも一致しない場合，Elsestatements のステートメントが実行されます。Elsestatements-n には複数の文を記述できます。複数の文を記述する場合は，文ごとに改行します。

説明

TestExpression が Case 節のどれかの式 ExpressionList-n に一致すると，次の Case 節，または

End Select ステートメントまでの Statements が実行されます。ブロックの実行が終わると，制御は End Select ステートメントの次のステートメントに移ります。TestExpression が複数の Case 節に一致するときは，最初に一致した Case 節に続くステートメントだけが実行されます。

Case Else 節には，TestExpression がどんな Case 節の ExpressionList-n にも一致しなかった場合に実行するステートメント ExpressionList-n を指定します。Case Else ステートメントは必ずしも必要ではありませんが，予測できない TestExpression の値を処理するために，Select Case ブロックに Case Else ステートメントを記述することをお勧めします。Case 節の式 ExpressionList が TestExpression と一致しない場合で Case Else ステートメントが指定されていない場合は，End Select ステートメントの次のステートメントから実行が続けられます。

Select Case ステートメントは，ネスト構造できます。このとき，各 Select Case ステートメントには，それぞれ対応する End Select ステートメントが必要です。

While...End

機能

指定した条件が真（True）である間，一連のステートメントの実行を繰り返します。

形式

While　Condition

　〔Statements〕

　〔Exit While〕

End〔While〕

指定項目

Condition

真（True），または偽（False）を評価する条件式を指定します。

Statements

Condition が真（True）の間に実行する一つ，または複数のステートメントを指定します。複数のステートメントを記述する場合は，ステートメントごとに改行します。

説明

Condition が真（True）の場合は，End ステートメントに達するまで，Statements 内のすべてのステートメントが実行されます。実行が End ステートメントに達すると，制御は再び While ステートメントに戻り，Condition が評価されます。Condition が真（True）の間，この処理が繰り返されます。真（True）でない場合は，End ステートメントの次のステートメントに制御が移ります。

Exit While ステートメントは，制御構造 While...End ループ内だけで使用できます。Exit While ステートメントを使用すると，指定した以外の条件で While...End ループを終了させることができます。Exit While ステートメントは，While...End ループ内の任意の位置で何回でも指定できます。

While...End ループは，任意のレベルでネスト構造にすることができます。End ステートメントは最後に実行された While ステートメントに対応します。

Function

機能

Function プロシージャの名前，および引数を宣言して，Function プロシージャの始まりを示します。

形式

Function　Name〔(ArgList)〕

　〔Statements〕

　〔Name = Expression〕

　〔Exit Function〕

　〔Statements〕

　〔Name = Expression〕

End〔Function〕

指定項目

Name

定義する Function プロシージャの名前を指定します。名前は変数名の付け方の規則に従って指定します。変数名の付け方の詳細は，「4.1.1 (1) 変数名の付け方」を参照してください。

注意

プロシージャ名には，変数名と同じ名前を指定できません。同じ名前を指定した場合は，プロシージャ名が優先されます。

ArgList

Function プロシージャを呼び出すときに，Function プロシージャに渡す引数を表す変数のリストを指定します。複数の変数を指定する場合はコンマ(，)で区切ります。引数がある場合には，引数を必ず括弧で囲んでください。

Statements

Function プロシージャ内で実行される一連のステートメントを指定します。

Expression

Function プロシージャの戻り値を指定します。

説明

Function プロシージャはパブリックプロシージャなので，スクリプト内のほかのすべてのプロシージャからも参照できます。

実行可能なコードはすべてプロシージャ内に記述する必要があります。また，Function プロシージャをほかの Function プロシージャ，または Sub プロシージャの中で定義することはできません。

Exit Function ステートメントは，Function プロシージャを直ちに終了します。プログラムの実行は，その Function プロシージャを呼び出したステートメントの次のステートメントから継続されます。Exit Function ステートメントは，Function プロシージャ内の任意の位置で何回でも指定できます。

Function プロシージャの呼び出し方法については，Call ステートメントの説明に記述しています。Call ステートメントの詳細は，「5.2 ステートメントの詳細」の「Call」を参照してください。

Function プロシージャから値を返すには，値を Function プロシージャ名の *Name* に入力します。Function プロシージャ内の任意の場所で，必要に応じて何回でも *Name* に値を入力できます。*Name* に値を入力しない場合は，既定の戻り値が返されます。既定の戻り値は，長さ 0 の文字列("")です。

Function プロシージャで使う変数には，Function プロシージャ内で明示的に宣言される変数と，それ以外の変数の 2 種類があります。プロシージャ内で Dim コマンドなどを使用して明示的に宣言された変数は，そのプロシージャの中だけで有効なローカル変数になります。プロシージャ内で明示的に宣言しないで使われている変数も，そのプロシージャの外部のさらに上のレベルで明示的に宣言されないかぎりは，ローカル変数となります。Function プロシージャ内のローカル変数の値は，プロシージャの実行が終了すると破棄されます。

プロシージャ内で明示的に宣言していない変数をプロシージャ内で使うことはできますが，その変数と同じ名前の変数などがスクリプトレベルで定義されている場合，その変数はグローバル変数として扱われます。

例

```
Function   BinarySearch (...)
  :
  ' 値が見つかりません。値として False を返します。
  If   lower > upper   Then
       BinarySearch = False
       Exit Function
  End
  :
End Function
```

Sub

機能

Sub プロシージャの名前，および引数を宣言して，Sub プロシージャの始まりを示します。

形式

Sub　Name〔(ArgList)〕

〔Statements〕

〔Exit Sub〕

〔Statements〕

End〔Sub〕

指定項目

Name

定義するSubプロシージャの名前を指定します。名前は変数名の付け方の規則に従って指定します。変数名の付け方の詳細は，「4.1.1 (1) 変数名の付け方」を参照してください。

注意

プロシージャ名には，変数名と同じ名前を指定できません。同じ名前を指定した場合は，プロシージャ名が優先されます。

ArgList

Sub プロシージャを呼び出すときに，Sub プロシージャに渡す引数を表す変数のリストを指定します。複数の変数を指定するときは，コンマ(，)で区切ります。引数がある場合は，引数を必ず括弧で囲んでください。

Statements

Sub プロシージャ内で実行される，一連のステートメントを指定します。

説明

Sub プロシージャはパブリックプロシージャなので，スクリプト内のほかのすべてのプロシージャからでも参照できます。

実行可能なコードはすべてプロシージャ内に記述する必要があります。また，Sub プロシージャをほかの Sub プロシージャ，Function プロシージャの中で定義することはできません。

Exit Sub ステートメントは，Sub プロシージャを直ちに終了します。プログラムの実行は，その Sub プロシージャを呼び出したステートメントの次のステートメントから継続されます。Exit Sub ステートメントは，Sub プロシージャ内の任意の場所に必要に応じて幾つでも指定できます。

Sub プロシージャと Function プロシージャは，引数を受け取って一連のステートメントを実行し，引数の値を変えられる点では似ています。しかし，Sub プロシージャは，値を返す Function プロシージャとは異なり，値を返さないため，式の中に記述することはできません。

Sub プロシージャを呼び出すには，プロシージャ名の後ろに引数リストを付けて指定します。Sub プロシージャの呼び出し方法については，Call ステートメントの説明に記述しています。Call ステ

ートメントの詳細は，「5.2 ステートメントの詳細」の「Call」を参照してください。

Sub プロシージャで使う変数には，Sub プロシージャ内で明示的に宣言される変数と，それ以外の変数の 2 種類があります。プロシージャ内で Dim コマンドなどを使用して明示的に宣言された変数は，そのプロシージャの中だけで有効なローカル変数になります。プロシージャ内で明示的に宣言しないで使われている変数も，そのプロシージャの外部のさらに上のレベルで明示的に宣言されないかぎりは，ローカル変数となります。Sub プロシージャ内のローカル変数の値は，プロシージャの実行が終了すると破棄されます。

プロシージャ内で明示的に宣言されていない変数をプロシージャ内で使うことはできますが，その変数と同じ名前の変数などがスクリプトレベルで定義されている場合，その変数はグローバルな変数として扱われます。

Call

機能

Sub プロシージャ，および Function プロシージャに制御を渡します。

形式

〔Call〕Name〔(ArgumentList)〕

指定項目

Name

呼び出すプロシージャの名前を指定します。

ArgumentList

プロシージャに引き渡す変数リスト，または式のどちらかを指定します。*ArgumentList* を複数指定する場合は，コンマ（，）で区切ります。

説明

プロシージャを呼び出す場合，キーワードの Call を省略できます。引数を必要とするプロシージャを呼び出す場合は，*ArgumentList* を括弧で囲んでも囲まなくてもかまいません。ただし，プロシージャをほかのプロシージャやコマンドの引数として呼び出したり，式の途中で呼び出したりする場合には，引数を必ず括弧で囲みます。Call ステートメントの構文で組み込み関数，またはユーザ定義型関数を呼び出す場合，その関数の戻り値を取得できません。

例

Call　MyProc (0)　または　MyProc 0

Exit xx

機能

Do...Loop ループ，For...End For ループ，For...Next ループ，While...End ループ，Function プロシージャ，または Sub プロシージャから抜け出します。

形式

Exit While

Exit For

Exit Do

Exit Function

Exit Sub

説明

各 Exit ステートメントの機能を次に示します。

Exit While

While...End ループを抜け出し，End ステートメントの次のステートメントに制御を移します。While...End ステートメントの中だけで使用できます。While...End ステートメントがネスト構造になっている場合は，Exit While のあるループの一つ外側のループに制御を移します。

Exit For

For...Next ループを抜け出し，Next ステートメントの次のステートメントに制御を移します。For...Next と For...End For ループの中だけで使用できます。For...Next，または For...End For ループがネスト構造になっている場合は，Exit For ステートメントが含まれているループの一つ外側のループに制御を移します。

Exit Do

Do...Loop ループを抜け出し，Loop ステートメントの次のステートメントに制御を移します。Do...Loop ステートメントの中だけで使用できます。Do...Loop ステートメントがネスト構造になっている場合は，Exit Do ステートメントのあるループの一つ外側のループに制御を移します。

Exit Function

このステートメントのある Function プロシージャを直ちに抜け出します。制御は Function プロシージャを呼び出したステートメントの次のステートメントに移ります。

Exit Sub

このステートメントのある Sub プロシージャを直ちに抜け出します。制御は Sub プロシージャを呼び出したステートメントの次のステートメントに移ります。

GoTo

機能

指定したラベルに処理を分岐させます。

形式

GoTo　LabelName

指定項目

LabelName

分岐先のラベル名を指定します。ラベル名は変数名の付け方の規則に従って指定します。変数名の付け方の詳細は，「4.1.1 (1) 変数名の付け方」を参照してください。

説明

指定されたラベル先に実行の制御を移して，ラベルの次の行から実行が再開されます。ラベル名に同位のレベルで同じ名前が複数個指定されている場合，文法エラーになります。

分岐先のラベルは任意の位置に記述できますが，GoTo ステートメントがプロシージャの内部で指定されている場合，分岐先を同じプロシージャの内部にする必要があります。

例

```
If Exec("/user/bin/vi",True,_BIN_+"Loging.TXT") = FALSE Then
GoTo ErrorExit
End

ErrorExit:                    ' 分岐先
:
```

Continue

機能

Do...Loop ループ，For...End For ループ，For...Next ループ，または While...End ループの先頭に戻ります。

形式

Continue

説明

Do...Loop ループ，For...End For ループ，For...Next ループ，または While...End ループの中だけで記述できます。それ以外の位置で記述した場合は文法エラーとなります。

Continue ステートメントは Do...Loop ループ，For...End For ループ，For...Next ループ，または While...End ループ内の任意の位置で何回でも指定できます。

Do...Loop ループ，For...End For ループ，For...Next ループ，または While...End ループがネスト構造になっている場合は，Continue ステートメントを含んでいる最も内側のループの先頭に制御を移します。

例

```
For　I = 1　To　10
  For　J = 1　To　10          ' ここに戻る
      :
    Continue
      :
  Next
Next
```

On Error

機能

エラー発生時に，指定したラベルに制御を移します。エラー発生時に，指定したラベルに制御を移さないようにすることもできます。

形式

On Error GoTo　LabelName

指定項目

LabelName

分岐先のラベル名を指定します。ラベル名は変数名の付け方の規則に従って指定します。変数名の付け方の詳細は，「4.1.1 (1) 変数名の付け方」を参照してください。

LabelName に 0 を指定すると，エラー発生時に，指定したラベルに制御が移らなくなります。

説明

On Error ステートメントは，ラベルの次の行から始まるエラー発生時の処理を有効にします。実行時にエラーが発生すると，ここで設定した分岐先に制御が移り，処理が実行されます。ただし，構文エラーが発生した場合，設定した分岐先には制御が移りません。分岐先のラベルを On Error ステートメントと同じプロシージャ内にする必要があります。

On Error GoTo 0 を指定しない場合は，ラベルの次の行から始まるエラー発生時の処理はプロシージャの終了時に自動的に無効になります。

補足

次の場合は，実行エラーとなっても On Error ステートメントで指定したラベルに分岐しません。

TextRead コマンドで「ファイルの終わりに達したエラー」の場合
また，IsDefine コマンドや IsExistFile コマンドのように実行エラーにならないコマンドは，コマンドの戻り値にかかわらず分岐しません。

例

```
On Error GoTo ErrHandler '以降エラー発生時にラベルの次の行に制御が移る
  Copy ( InDir+"CTL3D32.TXT" ,OutDir+"CTL3D32.TXT" ,VersionUp )
    :
  On Error GoTo 0   '以降エラー発生時にラベルの次の行に制御を移さない
    :
```

ErrHandler:　　'エラー発生時の分岐先

:

6. 基本コマンド

この章では，スクリプトを作成する場合に使用できる基本コマンドについて説明します。

6.1 基本コマンド一覧

スクリプトを作成する場合に使用できる，基本コマンドの一覧を表 6-1 に示します。

表 6-1 基本コマンド一覧

分類	コマンド名	意味
変数操作	Dim	変数を宣言してメモリ領域を割り当てます。
	Dim（配列）	配列変数を宣言してメモリ領域を割り当てます。
	SetEnvironment または SetEnv	環境変数を設定します。
	GetEnvironment または GetEnv	環境変数を取得します。
	SetGV	グローバル変数を設定します。
	GetGV	グローバル変数を取得します。
	DeleteGV	グローバル変数を削除します。
	GetArrayCount	配列変数の要素数（二次元配列の場合は行数または列数）を取得します。
文字列操作	InStr	文字列の中から指定した文字列を検索し，最初に検索した文字の文字位置（先頭からその位置までの文字数）を返します。
	InArray	配列変数の要素の中から指定した文字列を検索し，最初に検索したインデックス番号を返します。
	Len	文字列の文字数を返します。
	LCase	文字列の半角英字の大文字を小文字に変換します。
	UCase	文字列の半角英字の小文字を大文字に変換します。
	Left	文字列の左端から指定した文字数分の文字列を返します。
	Mid	文字列から指定した文字数分の文字列を返します。
	Right	文字列の右端から指定した文字数分の文字列を

		返します。
	Space	指定した数の半角スペースから成る文字列を返します。
	LTrim	文字列の左からスペースを削除した文字列を返します。
	RTrim	文字列の右からスペースを削除した文字列を返します。
	Trim	文字列の左右からスペースを削除した文字列を返します。
	+（文字列連結）	二つの式に対して文字列の連結を行います。
	&（文字列連結）	二つの式に対して文字列の連結を行います。
	&=（文字列連結）	変数と式の値に対して文字列の連結を行い，連結した文字列を変数に代入します。
	AddStr	指定した二つ以上の文字列に対して，指定した区切り文字を挿入し連結した文字列を返します。
	SeparateStrCount	指定した区切り文字によって分割された文字列の数を返します。
	SeparateStr	指定した区切り文字によって分割された文字列を返します。
	Str	指定した値を文字列で返します。
	Format	指定した値を書式化した文字列で返します。
	IsLower	文字列が半角英字の小文字かどうかを調べて，結果を真（True）または偽（False）で返します。
	IsUpper	文字列が半角英字の大文字かどうかを調べて，結果を真（True），または偽（False）で返します。
	IsSingleChar	文字列が半角文字かどうかを調べて，結果を真（True），または偽（False）で返します。
	IsMultiChar	文字列が全角文字かどうかを調べて，結果を真（True），または偽（False）で返します。
日付操作	Date	現在の日付を返します。
	Time	現在の時刻を返します。
	Year	指定した日付の年を表す値を，4 けたの数値で返します。
	Month	指定した日付の月を表す値を，1～12 の範囲の 2 けたの数値で返します。
	Day	指定した日付の日を表す値を，1～31 の範囲の 2 けたの数値で返します。
	Weekday	指定した日付の曜日を表す値を，1（日曜）～7（土

		曜）の範囲の 1 けたの数値，または曜日を表す文字列で返します。
	Hour	指定した時刻の時を表す値を，0～23 の範囲の 2 けたの数値で返します。
	Minute	指定した時刻の分を表す値を，0～59 の範囲の 2 けたの数値で返します。
	Second	指定した時刻の秒を表す値を，0～59 の範囲の 2 けたの数値で返します。
	CalcDate	指定した日付に，指定した値を加算，または減算した結果（日付）を返します。
	CompDate	指定した二つの日付を比較し，結果を真（True），または偽（False）で返します。
	GetDateCount	指定した二つの日付の経過日数を返します。
	CalcTime	指定した時刻に，指定した値を加算，および減算した結果（時刻）を返します。
	CompTime	指定した二つの時刻を比較し，結果を真（True），または偽（False）で返します。
	GetTimeCount	指定した二つの時刻の経過時間を返します。
	IsLeapYear	指定した西暦がうるう年かどうかを調べて，結果を真（True），または偽（False）で返します。
ファイル・ディレクトリ操作	TextFileReplace	テキストファイルの中の，特定の文字列を置き換えます。
	TextOpen	テキスト形式ファイルをオープンします。
	TextClose	テキスト形式ファイルをクローズします。
	TextRead	テキスト形式ファイルの 1 行分のデータを読み込みます。
	TextWrite	テキスト形式ファイルにデータを書き込みます。
	TextSeek	テキスト形式ファイルのデータの読み書き開始位置を移動します。
	GetTextPosition	テキスト形式ファイルの現在の読み書き開始位置を返します。
	MakeDir	ディレクトリを作成します。
	DeleteDir	ディレクトリを削除します。
	DeleteFile	ファイルを削除します。
	Rename	ファイル名を変更します。
	TempDir	一時ディレクトリを取得します。
	TempFile	一時ファイル名を取得します。

	SetFileTime	ファイルに日付と時刻を設定します。
	GetFileTime	ファイルの日付と時刻を取得します。
	GetFileSize	ファイルの容量を取得します。
	SplitFile	指定したサイズでファイルを分割します。
	CatFiles	複数のファイルを一つに統合します。
	SetStandardFile または SetStdFile	Exec コマンドで呼び出すプロセスの標準入力，標準出力，および標準エラーファイルを設定します。
	ResetStandardFile または ResetStdFile	Exec コマンドで呼び出すプロセスの標準入力，標準出力，および標準エラーファイルを解除します。
	SplitPath	フルパスを解析します。
	MakePath	フルパスを作成します。
	SetPath	実行ディレクトリのパスを設定します。
	GetPath	実行ディレクトリのパスを取得します。
	GetDiskFreeSpace	ディスクの空き容量を取得します。
	Copy	ファイルをコピーします。
メッセージ出力	Message	ファイル，またはコンソール画面へ指定したテキストを出力します。また，テキストを出力したコンソール画面を消去します。
	InputLine	コマンドラインメッセージを出力して，コマンドライン上からテキスト入力できるようにして，入力された値を変数へ格納します。
演算処理	+演算子（加算）	二つの数値の和を求めます。
	+=演算子（加算）	変数と式の値の和を求め，結果を変数に代入します。
	-演算子（減算）	二つの数値の差を求めます。数式の符号を反転した値を指定します。
	-=演算子（減算）	変数と式の値の差を求め，結果を変数に代入します。
	Mod 演算子（剰余演算）	二つの数値の除算を行い，その剰余を返します。
	Mod=演算子（剰余演算）	変数と式の値の除算を行い，その剰余を変数に代入します。
	*演算子（乗算）	二つの数値の積を求めます。
	*=演算子（乗算）	変数と式の値の積を求め，結果を変数に代入します。
	/演算子（除算）	二つの数値の商を計算し，結果を整数で返します。
	/=演算子（除算）	変数と式の値の商を計算し，結果を整数で変数に

		代入します。
	￥演算子（整数除算）	二つの数値の商を計算し，結果を整数で返します。
	￥=演算子（整数除算）	変数と式の値の商を計算し，結果を整数で変数に代入します。
	^演算子（べき乗）	二つの数値のべき乗を求めます。
	^=演算子（べき乗）	変数と式の値のべき乗を求め，結果を変数に代入します。
	比較演算子（=，<>，<，<=，>，>=）	二つの式を比較します。
	論理積（And）	二つの式の論理積を求めます。
	論理和（Or）	二つの式の論理和を求めます。
	論理否定（Not）	式の論理否定を求めます。
チェック処理	IsEmpty	変数が Empty 値かどうかを調べて，結果を真（True），または偽（False）で返します。
	IsDefine または IsDef	変数が定義されているかどうかを調べて，結果を真（True），または偽（False）で返します。
	IsNumeric	値が数値として評価できるかどうかを調べて，結果を真（True），または偽（False）で返します。
	IsEmptyDir	ディレクトリの中身が空かどうかを調べて，結果を真（True），または偽（False）で返します。
	IsExistDir	ディレクトリが存在するかどうかを調べて，結果を真（True），または偽（False）で返します。
	IsExistFile	ファイルが存在するかどうかを調べて，結果を真（True），または偽（False）で返します。
	IsWriteableDir	ディレクトリが書き込みできるかどうかを調べて，結果を真（True），または偽（False）で返します。
	IsNew	二つのファイルの日付の新旧を比較し，結果を真（True），または偽（False）で返します。
	CheckDirName	ディレクトリの末尾（￥）かどうかを調べます。
外部プログラム呼び出し	Exec	実行ファイルを呼び出します。複数パラメタを指定できます。
コメント	Rem または'	プログラム内にコメントを記述する場合に指定します。
その他	Sleep	指定した時間中，処理を中断します。
	Beep	スピーカからビープ音を鳴らします。
	Exit	スクリプトの実行を終了します。

	GetErrorMessage	指定したエラー詳細コードのエラーメッセージを返します。

6.2 変数操作コマンド

変数操作コマンドの詳細を説明します。

Dim（変数を宣言しメモリ領域を割り当てる）

機能

変数を宣言してメモリ領域を割り当てます。

形式

Dim VarName〔, VarName , ...〕

指定項目

VarName

定義する変数名を指定します。変数名は変数名の付け方の規則に従って指定します。変数名の付け方の詳細は，「4.1.1(1) 変数名の付け方」を参照してください。

説明

このコマンドを使ってスクリプトで宣言した変数は，宣言したスクリプト内のすべてのプロシージャから参照できます。プロシージャで宣言した変数は，宣言したプロシージャ内だけで参照できます。

ただし，スクリプト，またはプロシージャの Do...Loop，For...Next，For...End For，While...End ループ内で変数の宣言，または宣言していない変数への代入により割り当てた変数はループ内だけで参照できます。Dim で宣言していない変数に値を代入した場合は，自動的に変数を割り当て値の代入を行います。

変数の宣言時は，変数は Empty 値になります。

Dim（配列）（配列変数を宣言しメモリ領域を割り当てる）

機能

配列変数を宣言してメモリ領域を割り当てます。

形式

Dim VarName (〔Number1〕〔, Number2〕)〔, VarName (〔Number1〕〔, Number2〕) , ...〕

Dim VarName ［〔Number1〕〔, Number2〕］〔, VarName ［〔Number1〕〔, Number2〕］, …〕

指定項目

VarName

定義する配列変数の変数名を指定します。変数名は変数名の付け方の規則に従って指定します。変数名の付け方の詳細は，「4.1.1(1) 変数名の付け方」を参照してください。

Number1

配列変数の要素数（二次元配列の場合は行要素数）を数値，または値を格納した変数で指定します。省略した場合，要素数は可変となります。

Number2

二次元配列の列要素数を数値，または値を格納した変数で指定します。省略した場合，要素数は可変となります。

説明

このコマンドを使ってスクリプトで宣言した配列変数は，宣言したスクリプト内のすべてのプロシージャから参照できます。プロシージャで宣言した配列変数は，宣言したプロシージャ内だけで参照できます。

配列変数の宣言時は，配列変数のすべての要素が Empty 値になります。

例

```
' 二次元の固定な配列変数 Table1 を定義する。
Dim   Table1( 5 ,6 )

' 一次元の固定な配列変数 Table2 から一次元の可変な配列変数 Table3 に一括代入する。
Dim   Table2( 6 ) ,Table3( )
For   I = 1 To 6
   Table2( I ) = Time( )
Next
Table3 = Table2

' 一次元の固定な配列変数 Table4 と配列変数 Table5 を比較する。
```

```
Dim　Table4( 6 ) ,Table5( 6 )
　:
If　Table4 = Table5　Then
　:
' 次のような大小比較はできないため,実行エラーになる。
If　Table4 <= Table5　Then
　:
```

SetEnvironment または SetEnv（環境変数を設定する）

機能

環境変数を設定します。

形式

SetEnvironment (Type , EnvironmentName 〔, Value〕)

SetEnv (Type , EnvironmentName 〔, Value〕)

指定項目

Type

設定する環境変数を次の値で指定します。

値	意味
ProcessEnv	現在のプロセスの環境変数 設定する環境変数はコマンド実行後の現在のプロセス内で有効になります。

EnvironmentName

設定する環境変数名を文字列，または値を格納した変数名で指定します。

Value

設定する値を文字列，または値を格納した変数名で指定します。

この値は省略できます。省略した場合，長さ 0 の文字列（""）を仮定します。

説明

指定した環境変数に，指定した値を設定します。コマンドが正常に実行された場合は真（True）を，エラーが発生した場合は偽（False）を，コマンドの実行結果として返します。

指定した環境変数が存在しない場合は，新規に環境変数を作成して値を設定します。環境変数が存在する場合は，指定した値で更新します。

注意事項

SetEnvironment，または SetEnv コマンドは EnvironmentName で指定した環境変数名長+Value で指定した文字列長+2 バイトのメモリを使用します。なお，使用したメモリは実行が終了するまで解放されません。

例

'プロセス変数：I1_PATH に/ABC/を設定する。

SetEnvironment (ProcessEnv ,"I1_PATH" ,"/ABC/")

Exec ("ABC" ,True)

GetEnvironment または GetEnv（環境変数を取得する）

機能

環境変数を取得します。

形式

GetEnvironment (Type , EnvironmentName)

GetEnv (Type , EnvironmentName)

指定項目

Type

設定する環境変数を次の値で指定します。

値	意味
ProcessEnv	現在のプロセスの環境変数

EnvironmentName

取得する環境変数を文字列，または値を格納した変数名で指定します。

説明

指定した環境変数を取得して，値をコマンドの実行結果として返します。

指定した環境変数が存在しない場合は，長さ 0 の文字列（""）を返します。

例

' 環境変数 PATH の値が"/ABC/"の場合,変数 buff1 には"/ABC/"が格納される。

Dim　buff1

buff1 = GetEnvironment (ProcessEnv ,"PATH")

SetGV（グローバル変数を設定する）

機能

グローバル変数を設定します。

形式

SetGV (GlobalName ,〔Value〕〔, CompName〕)

指定項目

GlobalName

設定するグローバル変数を文字列，または値を格納した変数名で指定します。

Value

設定する値を文字列，数値，または値を格納した変数名で指定します。

この値は省略できます。省略した場合，長さ 0 の文字列（""）を仮定します。

CompName

設定するグローバル変数が存在するコンピュータ名を文字列，または値を格納した変数名で指定します。

コンピュータ名には，このコマンドを実行するコンピュータ名だけを指定できます。ほかのコンピュータ名を指定した場合は実行エラーになります。

この値は省略できます。省略した場合，このコマンドを実行するコンピュータ名を仮定します。

説明

指定したグローバル変数に，指定した値を設定します。コマンドが正常に実行された場合は真（True）を，エラーが発生した場合は偽（False）を，コマンドの実行結果として返します。

指定したグローバル変数が存在しない場合は，新規にグローバル変数を作成して値を設定します。グローバル変数が存在する場合は，指定した値で更新します。

補足

このコマンドは，"/opt/jp1script/data"ディレクトリ中に，グローバル変数ファイル（SPTGV.SPG）を作成します。このファイルはスクリプトの実行が終了しても残ります。グローバル変数を初期化する場合は，このファイルを削除してください。

例

' 自コンピュータ上のグローバル変数を設定する。

SetGV ("Debug_Mode" ,TRUE)

GetGV（グローバル変数を取得する）

機能

グローバル変数を取得します。

形式

GetGV (GlobalName〔, CompName〕)

指定項目

GlobalName

取得するグローバル変数を文字列，または値を格納した変数名で指定します。

CompName

取得するグローバル変数が存在するコンピュータ名を文字列，または値を格納した変数名で指定します。

コンピュータ名には，このコマンドを実行するコンピュータ名だけを指定できます。ほかのコンピュータ名を指定した場合は実行エラーになります。

この値は省略できます。省略した場合，このコマンドを実行するコンピュータ名を仮定します。

説明

指定したグローバル変数を取得して，値をコマンドの実行結果として返します。

指定したグローバル変数が存在しない場合は，長さ 0 の文字列（""）を返します。

例

' 自コンピュータ上のグローバル変数を取得する。

GetGV ("Debug_Mode")

DeleteGV（グローバル変数を削除する）

機能

グローバル変数を削除します。

形式

DeleteGV (*GlobalName*〔, *CompName*〕)

指定項目

GlobalName

削除するグローバル変数を文字列，または値を格納した変数名で指定します。

すべてのグローバル変数を削除する場合は次の値を指定します。

値	意味
AllGV	すべてのグローバル変数を削除します。ただし，グローバル変数ファイル（SPTGV.SPG）は残ります。

CompName

削除するグローバル変数が存在するコンピュータ名を文字列，または値を格納した変数名で指定します。

コンピュータ名には，このコマンドを実行するコンピュータ名だけを指定できます。ほかのコンピュータ名を指定した場合は実行エラーになります。

この値は省略できます。省略した場合，このコマンドを実行するコンピュータ名を仮定します。

説明

指定したグローバル変数を削除します。コマンドが正常に実行された場合は真（True）を，エラーが発生した場合は偽（False）を，コマンドの実行結果として返します。

指定したグローバル変数が存在しない場合は，何もしないで常に真（True）を返します。

例

'自コンピュータ上のグローバル変数をすべて削除する。

DeleteGV (AllGV)

GetArrayCount（配列変数の要素数を取得する）

機能

配列変数の要素数（二次元配列の場合は行数または列数）を取得します。

形式

GetArrayCount (ArrayName)

指定項目

ArrayName

配列変数名を指定します。

説明

指定した配列変数の要素数（二次元配列の場合は行数または列数）を取得します。

コマンドが正常に実行された場合は要素数（二次元配列の場合は行数または列数）を，エラーが発生した場合は長さ 0 の文字列（""）を，コマンドの実行結果として返します。

指定した配列変数が可変である場合は，すでに設定された要素数（二次元配列の場合は行数または列数）を返します。

例

```
' ＜一次元の固定な配列変数 A の場合＞
' 変数 result1 には要素数 10 が格納される。
Dim　A( 10 )
result1 = GetArrayCount ( A )

' ＜二次元の固定な配列変数 B の場合＞
' 変数 result2 と変数 result3 には列数 10 が,変数 result4 には行数 5 が格納される。
Dim　B( 5 ,10 )
result2 = GetArrayCount ( B( 1 ) )
result3 = GetArrayCount ( B( 5 ) )
result4 = GetArrayCount ( B )
```

```
' ＜二次元の可変な配列変数 C の場合＞
' 変数 result5 には列数 1 が,変数 result6 には列数 7 が,変数 result7 には行数 2 が格納される。
Dim　C( , )
C( 2 ,1 ) = "SUN"
C( 2 ,2 ) = "MON"
C( 2 ,3 ) = "TUE"
C( 2 ,4 ) = "WED"
C( 2 ,5 ) = "THU"
C( 2 ,6 ) = "FRI"
C( 2 ,7 ) = "SAT"
result5 = GetArrayCount ( C( 1 ) )
result6 = GetArrayCount ( C( 2 ) )
result7 = GetArrayCount ( C )
```

6.3 文字列操作コマンド

文字列操作コマンドの詳細を説明します。

InStr（文字列を検索し文字位置を返す）

機能

指定した文字列の中から指定した文字列を検索して，最初に見つかった文字位置（先頭からその位置までの文字数）を返します。

形式

```
InStr ( String , SearchStr ,　〔Start〕　〔, Compare〕　)
```

指定項目

String

検索対象となる文字列を文字列，または値を格納した変数名で指定します。この値が長さ 0 の文字列（""）の場合は，0 を返します。

SearchStr

検索する文字列を文字列，または値を格納した変数名で指定します。この値が長さ 0 の文字列（""）の場合は，0 を返します。

Start

String で指定した文字列の先頭を 1 として，検索開始する位置を先頭からの文字数で指定します。この値が *String* で指定した文字列の文字数を超える場合は，0 を返します。この値は省略できます。省略した場合，1 を仮定します。

Compare

文字列の比較方法を次の値で指定します。この値は省略できます。省略した場合，Twice を仮定します。

値	意味
真（True）	半角文字も全角文字も大文字と小文字を区別して比較します。
偽（False）	半角文字も全角文字も大文字と小文字を区別しないで比較します。
Twice	半角文字の大文字と小文字は区別しないで全角文字の大文字と小文字は区別して比較します。

説明

指定した文字列の中から指定した文字列を検索して，最初に見つかった文字位置（先頭からその位置までの文字数）を返します。全角文字も半角文字も同じ 1 文字として扱います。指定した文字列が見つからない場合は，0 を返します。

例

```
' 変数 point1 には 6 が格納される。
Dim　point1
point1 = InStr ( "ファイルを ABC 順にならべる" ,"abc" ,3 ,False )
```

InArray（配列変数の要素から文字列を検索し，インデックス番号を返す）

機能

指定した配列変数の要素の中から指定した文字列を検索し，最初に見つかったインデックス番号を返します。

形式

InArray (ArrayName , SearchStr ,　〔Start〕　〔, Compare〕　)

指定項目

ArrayName

検索対象となる配列変数を変数名で指定します。この配列変数の要素数が 0 の場合は，0 を返し

ます。

SearchStr

検索する文字列を文字列，または値を格納した変数名で指定します。この値が長さ 0 の文字列（""）の場合は，0 を返します。

Start

ArrayName で指定した配列変数の検索開始する位置を，インデックス番号の数値，または値を格納した変数名で指定します。

この値が *ArrayName* で指定した配列変数の要素数を超える場合は，0を返します。この値は省略できます。省略した場合，先頭の要素を仮定します。

Compare

文字列の比較方法を次の値で指定します。この値は省略できます。省略した場合，Twice を仮定します。

値	意味
真（True）	半角文字も全角文字も大文字と小文字を区別して比較します。
偽（False）	半角文字も全角文字も大文字と小文字を区別しないで比較します。
Twice	半角文字の大文字と小文字は区別しないで，全角文字の大文字と小文字は区別して比較します。

説明

指定した配列変数の要素の中から指定した文字列を検索して，最初に見つかったインデックス番号（1 から始まる数値）をコマンドの実行結果として返します。

指定した文字列が見つからない場合は，0 を返します。

例

' 次の図のように値を格納した配列変数 closeDay から指定のデータを検索する。

	（第1列）	（第2列）	（第3列）	・・・	（第11列）	（第12列）
（第1行）	"January"	"February"	"March"	・・・	"November"	"December"
（第2行）	5	2	1	・・・	1	6
（第3行）	19	16	15	・・・	15	20

```
Dim    closeDay( 3 ,12 )
       :
(配列変数 closeDay への値格納処理)
       :
```

```
monthName = "March"
buff = InArray ( closeDay( 1 ) ,monthName ,1 ,False )
If   buff > 0   Then
   firstDay = closeDay( 2 ,buff )
   Message(Target_DispOn, "実行結果",  monthName+"の第 1 定休日は"+firstDay+"です。
" )
End If


' 二次元の配列変数の全要素の中から指定のデータを検索する。
Dim   array1( , )
      :
(配列変数 array1 への値格納処理)
      :
allCnt = GetArrayCount ( array1 )
For   line = 1 To allCnt
       buff = InArray ( array1( line ) ,"1999" ,1 ,False )
       If   0 < buff   Then
             Exit For
       End If
Next
```

Len（文字列の文字数を返す）

機能

文字列の文字数を返します。

形式

Len (String)

指定項目

String

文字列，または値を格納した変数名を指定します。

説明

指定した文字列の文字数，または指定した変数に格納された文字列の文字数を返します。全角文字も半角文字も同じ 1 文字として扱います。

例

```
' 変数 length1 には 6 が格納される。
Dim　length1
length1 = Len ( "ABCDEF" )

' 変数 length2 には 4 が格納される。
Dim　length2
length2 = Len ( "ファイル" )
```

LCase（半角英字の大文字を小文字に変換する）

機能

文字列の半角英字の大文字を小文字に変換して，変換した文字列を返します。

形式

LCase (String)

指定項目

String

文字列，または値を格納した変数名を指定します。

説明

指定した文字列の半角英字の大文字を小文字に変換して，変換した文字列を返します。

指定した文字列の大文字だけが小文字に変換されます。大文字の半角英字以外の文字は変換されません。

例

```
' 変数 string1 には "abcdef" が格納される。
Dim　string1
string1 = LCase ( "abcDEF" )
```

UCase（半角英字の小文字を大文字に変換する）

機能

文字列の半角英字の小文字を大文字に変換して，変換した文字列を返します。

形式

UCase (String)

指定項目

String

文字列，または値を格納した変数名を指定します。

説明

指定した文字列の半角英字の小文字を大文字に変換して，変換した文字列を返します。

指定した文字列の小文字だけが大文字に変換されます。小文字の半角英字以外の文字は変換されません。

例

```
' 変数 string1 には "ABCDEF" が格納される。
```

```
Dim　string1
string1 = UCase ( "abcDEF" )
```

Left（左端から指定した文字数分の文字列を返す）

機能

文字列の左端から指定した文字数分の文字列を返します。

形式

Left (String , Length)

指定項目

String

文字列，または値を格納した変数名を指定します。

Length

String で指定した文字列から取り出す文字数を数値，または値を格納した変数名で指定します。

この値に 0 を指定した場合は，長さ 0 の文字列（""）を返します。また，String で指定した文字列全体の文字数以上の値を指定した場合は，String で指定した文字列全体を返します。

説明

指定した文字列の左端から指定した文字数分の文字列を返します。全角文字も半角文字も同じ 1 文字として扱います。

文字列の文字数を調べる場合は，Len コマンドを使用します。Len コマンドの詳細は，「6.3 文字列操作コマンド」の「Len（文字列の文字数を返す）」を参照してください。

例

```
' 変数 string1 には "ABCD" が格納される。
Dim　string1
string1 = Left ( "ABCDEFGH" ,4 )

' 変数 string2 には "ABC 順" が格納される。
Dim　string2
string2 = Left ( "ABC 順に並べる",4 )
```

Mid（指定した文字数分の文字列を返す）

機能

文字列の中間から指定した文字数分の文字列を返します。

形式

Mid (String , Start〔, Length〕)

指定項目

String

文字列，または値を格納した変数名を指定します。

Start

String で指定した文字列の先頭の位置を 1 として，取り出す文字列の位置を先頭からの文字数で指定します。

この値が String で指定した文字列の文字数を超える場合は，長さ 0 の文字列（""）を返します。

Length

取り出す文字数を数値，または値を格納した変数名で指定します。

この値を省略した場合，または文字列内の文字数がこの値より短い場合は，Start から後ろのすべての文字列が返されます。

説明

指定した文字列の中間から指定した文字数分の文字列を返します。全角文字も半角文字も同じ 1 文字として扱います。

文字列の文字数を調べるには，Len コマンドを使用します。Len コマンドの詳細は，「6.3 文字列操作コマンド」の「Len（文字列の文字数を返す）」を参照してください。

例

```
' 変数 string1 には "CDEFG" が格納される。
Dim　string1
string1 = Mid ( "ABCDEFGH" ,3 ,5 )
```

' 変数 string2 には "C 順に並べ" が格納される。

Dim　string2

string2 = Mid ("ABC 順に並べる" ,3 ,5)

Right（右端から指定した文字数分の文字列を返す）

機能

文字列の右端から指定した文字数分の文字列を返します。

形式

Right (String , Length)

指定項目

String

文字列，または値を格納した変数名を指定します。

Length

String で指定した文字列から取り出す文字数を，数値，または値を格納した変数名で指定します。

この値に 0 を指定した場合は，長さ 0 の文字列（""）を返します。また，String で指定した文字列全体の文字数以上の値を指定した場合は，String で指定した文字列全体を返します。

説明

指定した文字列の右端から指定した文字数分の文字列を返します。

全角文字も半角文字も同じ 1 文字として扱います。

文字列の文字数を調べる場合は，Len コマンドを使用します。Len コマンドの詳細は，「6.3 文字列操作コマンド」の「Len（文字列の文字数を返す）」を参照してください。

例

' 変数 string1 には "EFGH" が格納される。

Dim　string1

string1 = Right ("ABCDEFGH" ,4)

' 変数 string2 には "に並べる" が格納される。

Dim　string2

string2 = Right ("ABC 順に並べる" ,4)

Space（半角スペースから成る文字列を返す）

機能

指定した数の半角スペースから成る文字列を返します。

形式

Space (Number)

指定項目

Number

スペースの数を数値，または値を格納した変数名で指定します。指定できる値の範囲は，0～1,024 です。0 を指定した場合は，長さ 0 の文字列（""）を返します。また，1,025 以上の値を指定した場合は，1,024 を仮定します。

説明

指定した数の半角スペースから成る文字列を返します。

例

```
' 変数 string1 には "　　"（半角スペース 3 文字）が格納される。
Dim　string1
string1 = Space ( 3 )
```

LTrim（左側のスペースを削除した文字列を返す）

機能

文字列の左から，スペースを削除した文字列を返します。

形式

LTrim (String 〔, Option〕)

指定項目

String

文字列，または値を格納した変数名を指定します。

文字列の左側にスペースが見つからない場合は，*String* で指定した文字列全体を返します。

Option

オプションを次の値で指定します。

値	意味
Twice	文字列の左から，半角スペースと全角スペースを削除します。

説明

オプションに Twice を指定した場合は，文字列の左側の半角スペース，および全角スペースを削除します。

オプションを指定しない場合は，文字列の左側の半角スペースだけを削除します。

例

```
' 変数 string1 には"ABC DEFG　　"が格納される。
Dim　string1
string1 = LTrim ( "　　ABC DEFG　　" ,Twice )
```

RTrim（右側のスペースを削除した文字列を返す）

機能

文字列の右から，スペースを削除した文字列を返します。

形式

RTrim (String 〔, Option〕)

指定項目

String

文字列，または値を格納した変数名を指定します。

文字列の右側にスペースが見つからない場合は，*String* で指定した文字列全体を返します。

Option

オプションを次の値で指定します。

値	意味
Twice	文字列の右から，半角スペースと全角スペースを削除します。

説明

オプションに Twice を指定した場合は，文字列の右側の半角スペース，および全角スペースを

削除します。

オプションを指定しない場合は，文字列の右側の半角スペースだけを削除します。

例

```
' 変数 string1 には"　ABC DEFG"が格納される。
Dim　string1
string1 = RTrim ( "　ABC DEFG　" ,Twice )
```

Trim（左右のスペースを削除した文字列を返す）

機能

文字列の左右から，スペースを削除した文字列を返します。

形式

Trim (String 〔, Option〕)

指定項目

String

文字列，または値を格納した変数名を指定します。

文字列の左右にスペースが見つからない場合は，*String* で指定した文字列全体を返します。

Option

オプションを次の値で指定します。

値	意味
Twice	文字列の左右から，半角スペースと全角スペースを削除します。

説明

オプションに Twice を指定した場合は，文字列の左右の半角スペース，および全角スペースを削除します。

オプションを指定しない場合は，文字列の左右の半角スペースだけを削除します。

例

```
' 変数 string1 と変数 string2 と変数 string3 には"ABC DEFG"が格納される。
Dim　string1, string2, string3
string1 = Trim ( "　ABC DEFG　" ,Twice )
string2 = Trim ( "　　ABC DEFG" ,Twice )
```

string3 = Trim ("ABC DEFG　　" ,Twice)

+（文字列連結）

機能

二つの式に対して文字列の連結を行います。

形式

Result = Expression1 + Expression2

指定項目

Result

結果を受け取る変数名を指定します。

Expression1

任意の式を指定します。

Expression2

任意の式を指定します。

説明

式の形式によって，+演算子の動作は異なります。

式の形式と+演算子で行われる演算を次に示します。

式の形式	行われる演算
両方の式が文字	文字列連結
両方の式が数値	加算
両方の式が数値だけから成る文字列	加算
一方の式が数値，他方が文字列	文字列連結
一方の式が文字列，他方が数値だけから成る文字列	文字列連結
一方の式が数値，他方が数値だけから成る文字列	加算

両方の式が Empty 値の場合は，演算結果 *Result* は数値 0 になります。ただし，一方の式だけが Empty 値の場合は，他方の式がそのまま演算結果 *Result* として返されます。

例

' 変数 result1 には"ABCDEF"が格納される。

result1 = "ABC" + "DEF"

' 変数 result2 には 12 が格納される。

result2 = 7 + 5

&（文字列連結）

機能

二つの式に対して文字列の連結を行います。

形式

Result = Expression1 & Expression2

指定項目

Result

結果を受け取る変数名を指定します。

Expression1

任意の式を指定します。

Expression2

任意の式を指定します。

説明

式の形式によって，&演算子の動作は異なります。

式の形式と&演算子で行われる演算を次に示します。

式の形式	行われる演算
両方の式が文字	文字列連結
両方の式が数値	文字列連結
両方の式が数値だけから成る文字列	文字列連結
一方の式が数値，他方が文字列	文字列連結
一方の式が文字列，他方が数値だけから成る文字列	文字列連結
一方の式が数値，他方が数値だけから成る文字列	文字列連結

両方の式が Empty 値の場合は，演算結果 *Result* は長さ 0 の文字列（""）になります。ただし，一方の式だけが Empty 値の場合は，他方の式がそのまま演算結果 *Result* として返されます。

例

' 変数 result1 には"ABCDEF"が格納される。

result1 = "ABC" & "DEF"

' 変数 result2 には 75 が格納される。

result2 = 7 & 5

&=（文字列連結）

機能

変数の値と式の値に対して文字列の連結を行い，連結した文字列を変数に代入します。

形式

Result &= Expression

指定項目

Result

結果を受け取る変数名を指定します。

Expression

任意の式を指定します。

説明

&=演算子は変数 Result と式 Expression のすべての値を文字列とみなして文字列連結を行います。

両方の値が Empty 値の場合は，変数 Result には Empty 値が代入されます。変数 Result が Empty 値，または未定義の場合は，式 Expression の値がそのまま変数 Result に代入されます。式 Expression の値が Empty 値のときは，変数 Result の値は変わりません。

例

' 変数 result1 には"ABCDEF"が格納される。

result1 = "ABC"

result1 &= "DEF"

' 変数 result2 には"01"が格納される。

result2 = 0

result2 &= 1

```
' 変数 result3 には"100"が格納される。
Dim　result3
result3 &= 100
```

AddStr（指定した連結文字で文字列連結を行う）

機能

指定した二つ以上の文字列に対して，指定した連結文字で区切った文字列連結を行い，連結した文字列を返します。

形式

AddStr (〔SeparateChar〕 , 〔Option〕 , String1 , String2 〔, String3 , ...〕)

指定項目

SeparateChar

連結文字を文字列，または値を格納した変数名で指定します。

この値は省略できます。省略した場合，連結文字を挿入しません。

Option

オプションに次の値を指定します。

この値は省略できます。省略した場合，NeedDq を仮定します。

値	意味
NeedDq	String1～5 で指定した文字列中に連結文字とみなされるコンマ，タブ，スペースを含んでいるときだけ文字列を引用符（"）で囲みます。
AllDq	String1～5 で指定した文字列を無条件に引用符（"）で囲みます。

String1～5

連結する文字列を文字列，または値を格納した変数名で指定します。文字列は五つまで指定できます。

説明

指定した複数の文字列に対して，指定した連結文字で区切った文字列連結を行って，連結した文字列を返します。連結した文字列の文字数が半角文字で 1,024 文字を超える場合，1,025 文字以降は切り捨てられます。全角文字の場合は，512 文字を超える場合，513 文字以降は切り捨てられます。

例

```
' 変数 string1 には""コード　291" 価格 "3,000""が格納される。
Dim　string1
string1 = AddStr ( " " , ,"コード　291" ,"価格" ,"3,000" )

' 変数 string2 には""コード　291" "価格" "3,000""が格納される。
Dim　string2
string2 = AddStr ( " " ,AllDq ,"コード　291" ,"価格" ,"3,000" )
```

SeparateStrCount（文字列分割を行い，分割文字列の数を返す）

機能

指定した文字列に対して，指定した区切り文字で文字列分割を行い，分割した文字列の数を返します。

形式

SeparateStrCount (String , SeparateChar)

指定項目

String

分割する文字列を文字列，または値を格納した変数名で指定します。

SeparateChar

区切り文字を文字列，または値を格納した変数名で指定します。

説明

指定した文字列に対して，指定した区切り文字で文字列分割を行い，分割した文字列の数を返します。区切り文字が見つからず分割できない場合は，1 を返します。

区切り文字が String の先頭や末尾にある場合は，その区切り文字の直前や直後に長さ 0 の文字列（""）があるものとして分割した文字列の数を返します。

また，String で指定した文字列が長さ 0 の文字列（""）である場合は，0 を返します。

分割した文字列を取得するには，SeparateStr コマンドを使用します。分割する文字列に n 個の引用符（"）を文字としてそのまま含む場合は，（n×2）個の引用符（"）を指定します。

例

```
' 変数 count1 には 3 が格納される。
Dim　count1
count1 = SeparateStrCount ( "JP1　Script　01-00,01-01" ,"　" )

' 変数 count2 には 3 が格納される。
Dim　count2 ,param
param　= """コード　100""　価格　300"
count2 = SeparateStrCount ( param ,"　" )

' 変数 count3 には 4 が格納される。
Dim　count3
count3 = SeparateStrCount ( "コード 100;コード 200;コード 300;" ,";" )
```

SeparateStr（文字列分割を行い，分割文字列を返す）

機能

指定した文字列に対して，指定した区切り文字で文字列分割を行い，分割した文字列を返します。

形式

SeparateStr (String , SeparateChar〔, Position〕)

指定項目

String

分割する文字列を文字列，または値を格納した変数名で指定します。

SeparateChar

区切り文字を文字列，または値を格納した変数名で指定します。

Position

分割した文字列の並びの先頭を 1 として，取り出す文字列の位置を先頭からの文字列位置で指定します。区切り文字は含みません。この値が分割した文字列の数よりも大きい場合は，長さ0の文字列（""）が返されます。

この値は省略できます。省略した場合，1 を仮定します。

説明

指定した文字列に対して，指定した区切り文字で文字列分割を行い，指定した位置の分割した文字列を返します。

区切り文字が 2 文字以上連続している場合に分割した文字列は長さ 0 の文字列（""）になります。

分割した文字列の数を取得するには，SeparateStrCount コマンドを使用します。

例

```
' 変数 string1 には"01-00,01-01"が格納される。
Dim　string1
string1 = SeparateStr ( "JP1　Script　01-00,01-01" ,"　" ,3 )
```

```
' 変数 string2 には長さ 0 の文字列""が格納される。
Dim　string2
string2 = SeparateStr ( "JP1,Script,01-00,,01-01" ,"," ,4 )
```

Str（値を文字列で返す）

機能

指定した値を文字列で返します。

形式

Str (Number)

指定項目

Number

文字列に変換する値，または値を格納した変数名で指定します。

この値に文字列を指定した場合は，Number で指定した値をそのまま返します。

説明

指定した値を文字列で返します。

例

```
' 変数 string1 には 2 が格納される。
Dim　string1
string1 = 1 + 2 - 1

' 変数 string2 には"12-1"が格納される。
Dim　string2
string2 = Str ( 1 ) + Str ( 2 ) + Str ( -1 )

' 変数 string3 には 11 が格納される。
Dim　string3
string3 = Str ( 1 ) + 2 - 1
```

Format（値を書式化した文字列を返す）

機能

指定した値を，書式化した文字列で返します。

形式

Format (Form , Arg1 〔, Arg2 , …〕)

指定項目

Form

Arg1～32 で指定した値をどのように書式化するかを指定します。書式化の指示は，文字列，または値を格納した変数名で指定します。

Form の中身の文字はそのまま表されます。書式化の指示の先頭は%で指定します。

書式化の指示は次の値で指定します。

値	意味
%d	数値を 10 進数で表します。
%x	数値を 16 進数（小文字）で表します。
%X	数値を 16 進数（大文字）で表します。
%o	数値を 8 進数で表します。
%s	文字列をそのまま表します。
%c	1 文字をそのまま表します。 なお，全角文字も半角文字も同じ 1 文字として扱われます。また，Argn で指定した書式化する値が 2 文字以上の場合でも，先頭の 1 文字だけを表します。
%5d，%10s	書式化する値の最大けた数を指定します。
%05d	書式化する値の最大けた数を指定する場合に，最大けた数に満たない場合は値の先頭を 0 で埋めます。
%-5d，%-10s	値を左詰めで表します。

なお，%を書式化の指示の始まりとしてではなく文字として表す場合は，%%と指定します。

Arg1～32

書式化する値を文字列，数値，または値を格納した変数名で指定します。*Form* で指定した書式の順に複数の値を指定できます。

説明

指定した値を書式化した文字列で返します。

例

```
' 前ゼロ 10 進数 5 けたの識別子をパラメタとして実行ファイル ABC へ渡す。
Dim　numID ,strID
numID = GetGV ( "seqNo" )
If　IsEmpty ( numID )　Then
　numID = 1
End If
strID = Format ( "%05d" ,numID )
Exec ( _SCF_+"ABC" ,True ,strID )
```

IsLower（半角英字の小文字かどうかを調べる）

機能

指定した文字列が半角英字の小文字かどうかを調べて，結果を真（True）か偽（False）で返します。

形式

IsLower (String)

指定項目

String

調べる文字列，または値を格納した変数名を指定します。

この値に長さ 0 の文字列（""）を指定した場合は，偽（False）を返します。

説明

指定した文字列のすべての文字が半角英字の小文字かどうかを調べて，結果をコマンドの実行結果として返します。すべての文字が半角英字の小文字である場合は真（True）を，それ以外の場合は偽（False）を返します。

例

```
'グローバル変数：Level の内容が小文字の a なのか大文字の A なのかを調べる。
Dim　buff
buff = GetGV ( "Level" )
If　buff = "A"　Then
```

```
    If   IsLower ( buff )   Then
        Message( Target_DisOn, "実行結果","Small A Level." )
    Else
        Message( Target_DisOn, "実行結果","Large A Level." )
    End If
End If
```

IsUpper（半角英字の大文字かどうかを調べる）

機能

指定した文字列が半角英字の大文字かどうかを調べて，結果を真（True）か偽（False）で返します。

形式

IsUpper (String)

指定項目

String

調べる文字列，または値を格納した変数名を指定します。

この値に長さ 0 の文字列（""）を指定した場合は，偽（False）を返します。

説明

指定した文字列のすべての文字が半角英字の大文字かどうかを調べて，結果をコマンドの実行結果として返します。すべての文字が半角英字の大文字である場合は真（True）を，それ以外の場合は偽（False）を返します。

例

```
'グローバル変数：Level の内容が小文字の a なのか大文字の A なのかを調べる。
Dim   buff
buff = GetGV ( "Level" )
If   buff = "A"   Then
    If   IsUpper ( buff )   Then
        Message( Target_DisOn, "実行結果","Large A Level." )
    Else
```

```
        Message( Target_DisOn, "実行結果","Small A Level." )
    End If
End If
```

IsSingleChar（半角文字かどうかを調べる）

機能

指定した文字列が半角文字かどうかを調べて，結果を真（True）か偽（False）で返します。

形式

IsSingleChar (String)

指定項目

String

調べる文字列，または値を格納した変数名を指定します。

この値に長さ 0 の文字列（""）を指定した場合は，偽（False）を返します。

説明

指定した文字列のすべての文字が半角文字かどうかを調べて，結果をコマンドの実行結果として返します。すべての文字が半角文字である場合は真（True）を，それ以外の場合は偽（False）を返します。

Shift-JIS コードでは半角かな文字は 1 バイトのコードで表されますが，日本語 EUC コードでは半角かな文字は 2 バイトのコードで表されます。また，UTF-8 では半角かな文字は 3 バイトのコードで表されます。このコマンドは，Shift-JIS コードの半角かな文字は半角文字とみなしますが，EUC コード，または UTF-8 の半角かな文字は全角文字とみなします。

日本語EUCの半角かな文字

16 進コード	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
8EA0		｡	｢	｣	､	･	ｦ	ｧ	ｨ	ｩ	ｪ	ｫ	ｬ	ｭ	ｮ	ｯ
8EB0	ｰ	ｱ	ｲ	ｳ	ｴ	ｵ	ｶ	ｷ	ｸ	ｹ	ｺ	ｻ	ｼ	ｽ	ｾ	ｿ
8EC0	ﾀ	ﾁ	ﾂ	ﾃ	ﾄ	ﾅ	ﾆ	ﾇ	ﾈ	ﾉ	ﾊ	ﾋ	ﾌ	ﾍ	ﾎ	ﾏ
8ED0	ﾐ	ﾑ	ﾒ	ﾓ	ﾔ	ﾕ	ﾖ	ﾗ	ﾘ	ﾙ	ﾚ	ﾛ	ﾜ	ﾝ	ﾞ	ﾟ

UTF-8の半角かな文字

16 進コード	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
EFBDA0		｡	｢	｣	､	･	ｦ	ｧ	ｨ	ｩ	ｪ	ｫ	ｬ	ｭ	ｮ	ｯ
EFBDB0	ｰ	ｱ	ｲ	ｳ	ｴ	ｵ	ｶ	ｷ	ｸ	ｹ	ｺ	ｻ	ｼ	ｽ	ｾ	ｿ
EFBD80	ﾀ	ﾁ	ﾂ	ﾃ	ﾄ	ﾅ	ﾆ	ﾇ	ﾈ	ﾉ	ﾊ	ﾋ	ﾌ	ﾍ	ﾎ	ﾏ
EFBD90	ﾐ	ﾑ	ﾒ	ﾓ	ﾔ	ﾕ	ﾖ	ﾗ	ﾘ	ﾙ	ﾚ	ﾛ	ﾜ	ﾝ	ﾞ	ﾟ

例

```
'グローバル変数:#の内容が半角文字なのかそれ以外なのかを調べる。
Dim　buff
buff = GetGV ( "#" )
If　IsSingleChar ( buff )　Then
　　Message( Target_DispOn, "実行結果","#" + buff )
Else
　　Message( Target_DispOn, "実行結果","項番" + buff )
End If
```

IsMultiChar（全角文字かどうかを調べる）

機能

指定した文字列が全角文字かどうかを調べて，結果を真（True）か偽（False）で返します。

形式

IsMultiChar (String)

指定項目

String

調べる文字列，または値を格納した変数名を指定します。

この値に長さ 0 の文字列（""）を指定した場合は，偽（False）を返します。

説明

指定した文字列のすべての文字が全角文字かどうかを調べて，結果をコマンドの実行結果として返します。すべての文字が全角文字である場合は真（True）を，それ以外の場合は偽（False）を返します。

Shift-JIS コードでは半角かな文字は 1 バイトのコードで表されますが，日本語 EUC コードでは半角かな文字は 2 バイトのコードで表されます。また，UTF-8 では半角かな文字は 3 バイトのコードで表されます。このコマンドは，Shift-JIS コードの半角かな文字は半角文字とみなしますが，EUC コード，または UTF-8 の半角かな文字は全角文字とみなします。

日本語EUCの半角かな文字

16 進コード	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
8EA0		｡	｢	｣	､	･	ｦ	ｧ	ｨ	ｩ	ｪ	ｫ	ｬ	ｭ	ｮ	ｯ
8EB0	ｰ	ｱ	ｲ	ｳ	ｴ	ｵ	ｶ	ｷ	ｸ	ｹ	ｺ	ｻ	ｼ	ｽ	ｾ	ｿ
8EC0	ﾀ	ﾁ	ﾂ	ﾃ	ﾄ	ﾅ	ﾆ	ﾇ	ﾈ	ﾉ	ﾊ	ﾋ	ﾌ	ﾍ	ﾎ	ﾏ
8ED0	ﾐ	ﾑ	ﾒ	ﾓ	ﾔ	ﾕ	ﾖ	ﾗ	ﾘ	ﾙ	ﾚ	ﾛ	ﾜ	ﾝ	ﾞ	ﾟ

UTF-8の半角かな文字

16 進コード	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
EFBDA0		｡	｢	｣	､	･	ｦ	ｧ	ｨ	ｩ	ｪ	ｫ	ｬ	ｭ	ｮ	ｯ
EFBDB0	ｰ	ｱ	ｲ	ｳ	ｴ	ｵ	ｶ	ｷ	ｸ	ｹ	ｺ	ｻ	ｼ	ｽ	ｾ	ｿ
EFBD80	ﾀ	ﾁ	ﾂ	ﾃ	ﾄ	ﾅ	ﾆ	ﾇ	ﾈ	ﾉ	ﾊ	ﾋ	ﾌ	ﾍ	ﾎ	ﾏ
EFBD90	ﾐ	ﾑ	ﾒ	ﾓ	ﾔ	ﾕ	ﾖ	ﾗ	ﾘ	ﾙ	ﾚ	ﾛ	ﾜ	ﾝ	ﾞ	ﾟ

例

```
'グローバル変数：#の内容が全角文字なのかそれ以外なのかを調べる。
Dim　buff
buff = GetGV ( "#" )
If　IsMultiChar ( buff )　Then
　Message( Target_DispOn, "実行結果","項番" + buff )
Else
　Message(Target_DispOn, "実行結果", "#" + buff )
End If
```

6.4 日付操作コマンド

日付操作コマンドの詳細を説明します。

Date（現在の日付を返す）

機能

現在の日付を返します。

形式

Date

このコマンドに引数はありません。

説明

現在の日付を yyyy/mm/dd の形式で返します。

Time（現在の時刻を返す）

機能

現在の時刻を返します。

形式

Time

このコマンドに引数はありません。

説明

現在の時刻を hh:mm:ss の形式で返します。

Year（指定した日付の年を返す）

機能

指定した日付の，年を表す値を 4 けたの整数値で返します。

形式

Year (〔Date〕)

指定項目

Date

日付を yyyy/mm/dd の形式で指定します。
yyyy に 0～69 の整数値を指定すると，2,000～2,069 に読み替えられます。
mm に存在しない不正な値を指定した場合は，エラーになります。
dd に存在しない不正な値を指定した場合は，存在する日付に置き換えて結果を返します。例えば，1999/12/32（存在しない日付）を指定した場合，2000/1/1（存在する日付）に置き換えて，2000 を返します。
この値は省略できます。省略した場合，現在の日付を仮定します。

説明

指定した日付の年を表す値を 4 けたの数値で返します。

例

```
'今日の日付の年を取得する。
Dim today
today = Date
Year( today )
```

Month（指定した日付の月を返す）

機能

指定した日付の，月を表す値を，1～12 の範囲の 1～2 けたの数値で返します。

形式

Month (〔Date〕)

指定項目

Date

日付を yyyy/mm/dd の形式で指定します。
yyyy に 0～69 の整数値を指定すると，2,000～2,069 に読み替えられます。
mm に存在しない不正な値を指定した場合は，エラーになります。

dd に存在しない不正な値を指定した場合は，存在する日付に置き換えて結果を返します。例えば，1999/12/32（存在しない日付）を指定した場合，2000/1/1（存在する日付）に置き換えて，1 を返します。
この値は省略できます。省略した場合，現在の日付を仮定します。

説明

指定した日付の月を表す値を 1～12 の範囲の 1～2 けたの数値で返します。

例

```
'今日の日付の月を取得する。
Dim today
today = Date
Month( today )
```

Day（指定した日付の日を返す）

機能

指定した日付の，日を表す値を，1～31 の範囲の 1～2 けたの数値で返します。

形式

Day （〔Date〕）

指定項目

Date

日付を yyyy/mm/dd の形式で指定します。
yyyy に 0～69 の整数値を指定すると，2,000～2,069 に読み替えられます。
mm に存在しない不正な値を指定した場合は，エラーになります。
dd に存在しない不正な値を指定した場合は，存在する日付に置き換えて結果を返します。例えば，1999/12/32（存在しない日付）を指定した場合，2000/1/1（存在する日付）に置き換えて，1 を返します。
この値は省略できます。省略した場合，現在の日付を仮定します。

説明

指定した日付の日を表す値を，1～31 の範囲の 1～2 けたの数値で返します。

例

```
'今日の日付の日を取得する。
Dim today
today = Date
Day(today)
```

Weekday（指定した日付の曜日を返す）

機能

指定した日付の，曜日を表す値を，1（日曜）～7（土曜）の範囲の 1 けたの整数値，または曜日を表す文字列で返します。

形式

Weekday (〔Date〕 〔, Option〕)

指定項目

Date

日付を yyyy/mm/dd の形式で指定します。

yyyy に 0～69 の整数値を指定すると，2,000～2,069 に読み替えられます。

mm に存在しない不正な値を指定した場合は，エラーになります。

dd に存在しない不正な値を指定した場合は，存在する日付に置き換えて結果を返します。例えば，1999/12/32（存在しない日付）を指定した場合，2000/1/1（存在する日付）に置き換えて，7（土曜日）を返します。

この値は省略できます。省略した場合，現在の日付を仮定します。

Option

オプションを次の値で指定します。

値	意味
String	曜日を表す値を英字の文字列で返します。
StringJ	曜日を表す値を日本語の文字列で返します。

説明

オプションの指定を省略した場合
指定した日付の曜日を表す値を 1（日曜）～7（土曜）の 1 けたの整数値で返します。返される値とその内容を次に示します。

値	内容
1	日曜

2	月曜
3	火曜
4	水曜
5	木曜
6	金曜
7	土曜

オプションを指定した場合
指定した日付の曜日を表す値を文字列で返します。返される値とその内容を次に示します。

String を指定	StringJ を指定	内容
"SUN"	"日"	日曜
"MON"	"月"	月曜
"TUE"	"火"	火曜
"WED"	"水"	水曜
"THU"	"木"	木曜
"FRI"	"金"	金曜
"SAT"	"土"	土曜

例

```
'今日の日付の曜日を取得する
Dim today
today = Date
Weekday( today )
```

Hour（指定した時刻の時を返す）

機能

指定した時刻の，時を表す値を 0～23 の範囲の 1～2 けたの数値で返します。

形式

Hour (〔Time〕)

指定項目

Time

時刻を hh:mm:ss の形式で指定します。
hh に存在しない不正な値を指定した場合は，エラーになります。

mm と ss に存在しない不正な値を指定した場合は，存在する時刻に置き換えて結果を返します。
例えば，19:61:00（存在しない時刻）を指定した場合，20:01:00（存在する時刻）に置き換えて，20 を返します。
この値は省略できます。省略した場合，現在の時刻を仮定します。

説明

指定した時刻の時を表す値を 0～23 の範囲の 1～2 けたの数値で返します。

例

```
'今日の時刻の時を取得する
Dim today
today = Time
Hour(today)
```

Minute（指定した時刻の分を返す）

機能

指定した時刻の，分を表す値を 0～59 の範囲の 1～2 けたの数値で返します。

形式

Minute (〔Time〕)

指定項目

Time

時刻を hh:mm:ss の形式で指定します。
hh に存在しない不正な値を指定した場合は，エラーになります。
mm と ss に存在しない不正な値を指定した場合は，存在する時刻に置き換えて結果を返します。
例えば，19:61:00（存在しない時刻）を指定した場合，20:01:00（存在する時刻）に置き換えて，1 を返します。
この値は省略できます。省略した場合，現在の時刻を仮定します。

説明

指定した時刻の分を表す値を 0～59 の範囲の 1～2 けたの数値で返します。

例

```
'今日の時刻の分を取得する
Dim today
today = Time
Minute(today)
```

Second（指定した時刻の秒を返す）

機能

指定した時刻の，秒を表す値を 0～59 の範囲の 1～2 けたの数値で返します。

形式

Second (〔Time〕)

指定項目

Time

時刻を hh:mm:ss の形式で指定します。
hh に存在しない不正な値を指定した場合は，エラーになります。
mm と ss に存在しない不正な値を指定した場合は，存在する時刻に置き換えて結果を返します。
例えば，19:00:61（存在しない時刻）を指定した場合，19:01:01（存在する時刻）に置き換えて，1 を返します。
この値は省略できます。省略した場合，現在の時刻を仮定します。

説明

指定した時刻の秒を表す値を 0～59 の範囲の 1～2 けたの数値で返します。

例

```
'今日の時刻の秒を取得する
Dim today
today = Time
Second(today)
```

CalcDate（指定した日付に加算減算を行う）

機能

指定した年数，月数，日数を，指定した日付に足したり，指定した日付から引いたりして求めた日付を返します。

形式

CalcDate (Date , Calc , 〔Years〕 , 〔Months〕 〔, Days〕)

指定項目

Date

日付を yyyy/mm/dd の形式で指定します。

yyyy に 0～69 の整数値を指定すると，2,000～2,069 に読み替えられます。

mm に存在しない不正な値を指定した場合は，エラーになります。

dd に存在しない不正な値を指定した場合は，存在する日付に置き換えます。例えば，1999/12/32（存在しない日付）を指定した場合，2000/01/01（存在する日付）に置き換えます。

Calc

演算方法を次の値で指定します。

値	意味
Minus	Date で指定した日付から，Years，Months，Days で指定した年数，月数，日数を引いた日付を返します（減算）。
Plus	Date で指定した日付に，Years，Months，Days で指定した年数，月数，日数を足した日付を返します（加算）。

Years

年数を 0 以上の数値，または値を格納した変数名で指定します。

この値は省略できます。省略した場合，0 を仮定します。

Months

月数を 0 以上の数値，または値を格納した変数名で指定します。

この値は省略できます。省略した場合，0 を仮定します。

Days

日数を 0 以上の数値，または値を格納した変数名で指定します。

この値は省略できます。省略した場合，0 を仮定します。

説明

指定した年数，月数，日数を，指定した日付に足したり，指定した日付から引いたりして求め，yyyy/mm/dd の日付の形式で返します。

求めた日付が存在しない日付になった場合，その日付以降の存在する日付を返します。例えば，求めた日付が 1999/02/29（存在しない日付）になった場合，1999/03/01（存在する日付）を返します。

エラーが発生した場合は，長さ 0 の文字列（""）を返します。

例

```
' 本日から 10 日前の日付を表示する。
Dim　result
result = CalcDate ( Date() ,Minus , , ,10 )
Message( Target_DispOn, "実行結果","今日から 10 日前は"+result+"です。" )
```

CompDate（二つの日付を比較する）

機能

指定した二つの日付を比較して，結果を真（True），または偽（False）で返します。

形式

CompDate (Date1 , Comp , Date2)

指定項目

Date1

比較する一方の日付を yyyy/mm/dd の形式で指定します。

yyyy に 0～69 の範囲の整数値を指定すると，2,000～2,069 に読み替えられます。

mm に存在しない不正な値を指定した場合は，エラーになります。

dd に存在しない不正な値を指定した場合は，存在する日付に置き換えます。例えば，1999/12/32（存在しない日付）を指定した場合，2000/01/01（存在する日付）に置き換えます。

Comp

比較方法を次の値で指定します。

値	意味
Equal	等しい（ = ）
NotEqual	等しくない（ < > ）
Before	Date1 より，Date2 の方が前の日付である（ > ）
After	Date1 より，Date2 の方があとの日付である（ < ）

Date2

比較するもう一方の日付を yyyy/mm/dd の形式で指定します。

Date1 と同様に，yyyy に 0～69 の整数値を指定すると，2,000～2,069 に読み替えられ，dd に存在しない不正な値を指定した場合は，存在する日付に置き換えます。

説明

二つの日付を *Comp* で指定した方法で比較し，結果をコマンドの実行結果として返します。比較結果が *Comp* で指定した値と一致する場合は真（True）を，それ以外の場合は偽（False）を返します。

エラーが発生した場合は，長さ 0 の文字列（""）を返します。

例

```
' 更新日付が今日から 30 日前のファイル（拡張子.TXT） を削除する。
Dim　fileName ,delDate ,updDate
delDate = CalcDate ( Date() ,Minus ,0 ,0 ,29 )

For　fileName = _TEMP_+"*.TXT"　Do
  GetFileTime ( _TEMP_+fileName ,updDate , ,Update )
  If　CompDate ( updDate ,After ,deldate ) = True　Then
    DeleteFile ( _TEMP_+fileName )
  End If
End For
```

GetDateCount（二つの日付の経過日数を取得する）

機能

指定した二つの日付の経過日数を取得します。

形式

GetDateCount (StartDate , EndDate 〔, UnitOFDate〕)

指定項目

StartDate

経過日数を求めるときの開始日付を yyyy/mm/dd の形式で指定します。

yyyy に 0～69 の範囲の整数値を指定すると，2,000～2,069 と読み替えられます。

mm に存在しない不正な値を指定した場合は，エラーになります。

dd に存在しない不正な値を指定した場合は，存在する日付に置き換えます。例えば，1999/12/32（存在しない日付）を指定した場合，2000/01/01（存在する日付）に置き換えます。

EndDate

経過日数を求めるときの終了日付を yyyy/mm/dd の形式で指定します。

StartDate と同様に，yyyy に 0～69 の整数値を指定すると，2,000～2,069 に読み替えられ，dd に存在しない不正な値を指定した場合は，存在する日付に置き換えます。

UnitOFDate

取得する経過日数の単位を次の値で指定します。この値は省略できます。省略した場合，DayU を仮定します。

値	意味	例
YearU	年の単位で取得します。	開始日付が 2001/4/1 で，終了日付が 2002/4/1 である場合は 1 になり，終了日付が 2002/3/31 である場合は 0 になります。
MonthU	月の単位で取得します。	開始日付が 2001/4/10，終了日付が 2002/5/10 である場合は 1 になり，終了日付が 2001/5/9 である場合は 0 になります。
DayU	日の単位で取得します。	―

説明

指定した二つの日付の経過日数を取得し，コマンドの実行結果として返します。*StartDate* で指定した日付が *EndDate* で指定した日付よりもあとの日付である場合，経過日数は負の値になります。

例

```
' 2001 年 4 月 1 日から 2002 年 3 月 31 日までの経過日数を年,月,日の単位で求める。
Dim　date1 ,date2 ,yBuff ,mBuff ,dBuff
date1 = "2001/4/1"
date2 = "2002/3/31"

' yBuff には 0,mBuff には 11,dBuff には 364 が格納される。
yBuff = GetDateCount ( date1 ,date2 ,YearU )
mBuff = GetDateCount ( date1 ,date2 ,MonthU )
dBuff = GetDateCount ( date1 ,date2 ,DayU )
```

CalcTime（指定した時刻に加算減算を行う）

機能

指定した時刻に，指定した時，分，または秒を足したり引いたりした時刻を返します。

形式

CalcTime (Time , Calc , 〔Hours〕 , 〔Minutes〕 , 〔Seconds〕 〔, DaysBuff〕)

指定項目

Time

時刻を hh:mm:ss の形式で指定します。

hh に存在しない不正な値を指定した場合は，エラーになります。

mm と ss に存在しない不正な値を指定した場合は，存在する時刻に置き換えて結果を返します。例えば，19:00:61（存在しない時刻）を指定した場合，19:01:01（存在する時刻）に置き換えます。

Calc

演算方法を次の値で指定します。

値	意味
Minus	Time で指定した時刻から，Hours，Minutes，Seconds で指定した時，分，秒を引いた日付を返します（減算）。
Plus	Time で指定した時刻に，Hours，Minutes，Seconds で指定した時，分，秒を足した日付を返します（加算）。

Hours

時を 0 以上の数値，または値を格納した変数名で指定します。

この値は省略できます。省略した場合，0 を仮定します。

Minutes

分を 0 以上の数値，または値を格納した変数名で指定します。

この値は省略できます。省略した場合，0 を仮定します。

Seconds

秒を 0 以上の数値，または値を格納した変数名で指定します。

この値は省略できます。省略した場合，0 を仮定します。

DaysBuff

Time で指定した時刻を基点として求めた時刻が，日を超えた回数を受け取るための変数名を指定します。日を超えなかった場合は，0 を受け取ります。この変数名の指定は省略できます。

説明

指定した時刻に，指定した時，分，または秒を足したり引いたりした時刻を求め，求めた時刻を hh:mm:ss の形式で返します。

エラーが発生した場合は，長さ 0 の文字列（""）を返します。

例

```
' 現在の時刻から 7 時間 45 分後の時間を求める。
Dim　resTime ,resdate ,daysBuff
resTime = CalcTime ( Time() ,Plus ,7 ,45 , ,daysBuff )
resDate = CalcDate ( Date() , Plus , , ,daysBuff )
```

CompTime（二つの時刻を比較する）

機能

指定した二つの時刻を比較して，結果を真（True），または偽（False）で返します。

形式

CompTime (Time1 , Comp , Time2)

指定項目

Time1

比較する一方の時刻を hh:mm:ss の形式で指定します。

hh に存在しない不正な値を指定した場合は，エラーになります。

mm と ss に存在しない不正な値を指定した場合は，存在する時刻に置き換えて結果を返します。例えば，19:00:61（存在しない時刻）を指定した場合，19:01:01（存在する時刻）に置き換えます。

Comp

比較方法を次の値で指定します。

値	意味
Equal	等しい（=）
NotEqual	等しくない（<>）
Before	Time1 より，Time2 の方が前の時刻である（>）
After	Time1 より，Time2 の方があとの時刻である（<）

Time2

比較するもう一方の時刻を hh:mm:ss の形式で指定します。

Time1 と同様に，mm と ss に存在しない不正な値を指定した場合は，存在する時刻に置き換えます。

説明

Comp で指定した方法で二つの時刻を比較し，結果をコマンドの実行結果として返します。比較結果が *Comp* で指定した値と一致する場合は真（True）を，それ以外の場合は偽（False）を返します。

エラーが発生した場合は，長さ 0 の文字列（""）を返します。

例

```
' 更新時間が今日の午前中のファイル（拡張子.TXT）を削除する。
Dim　fileName ,updDate ,updTime

For　fileName = _TEMP_+"*.TXT"　Do
  GetFileTime ( _TEMP_+fileName ,updDate ,updTime ,Update )
  If　CompDate ( updDate ,Equal ,Date() )　Then
    If　CompTime ( updTime ,After ,"12:00:00" ) = True　Then
      DeleteFile ( _TEMP_+fileName )
    End If
  End If
End For
```

GetTimeCount（二つの時刻の経過時間を取得する）

機能

指定した二つの時刻の経過時間を取得します。

形式

GetTimeCount (StartTime , EndTime 〔, UnitofTime〕)

指定項目

StartTime

経過時間を求めるときの開始時刻を hh:mm:ss の形式で指定します。

hh に存在しない不正な値を指定した場合は，エラーになります。

mm と ss に存在しない不正な値を指定した場合は，存在する時刻に置き換えて結果を返します。例えば，19:00:61（存在しない時刻）を指定した場合，19:01:01（存在する時刻）に置き換えます。

EndTime

経過時間を求めるときの終了時刻を hh:mm:ss の形式で指定します。

StartTime と同様に，mm と ss に存在しない不正な値を指定した場合は，存在する時刻に置き換えます。

UnitofTime

取得する経過時間の単位を次の値で指定します。この値は省略できます。省略した場合，SecondU を仮定します。

値	意味	例
HourU	時の単位で取得します。	開始時刻が 9:00:00 で，終了時刻が 10:00:00 である場合は 1 になり，終了時刻が 9:59:59 である場合は 0 になります。
MinuteU	分の単位で取得します。	開始時刻が 9:10:00，終了時刻が 9:11:00 である場合は 1 になり，終了時刻が 9:10:59 である場合は 0 になります。
SecondU	秒の単位で取得します。	ー

説明

指定した二つの時刻の経過時間を取得し，コマンドの実行結果として返します。*StartTime* で指定され時刻が *EndTime* で指定した時刻よりもあとの時刻である場合，経過時間は負の値になります。

例

```
' 9:10:30 から 10:09:20 までの経過時間を時,分,秒の単位で求める。
Dim   time1 ,time2 ,hBuff ,mBuff ,sBuff
time1 = "9:10:30"
time2 = "10:09:20"

' hBuff には 0,mBuff には 58,sBuff には 3530 が格納される。
hBuff = GetTimeCount ( time1 ,time2 ,HourU )
mBuff = GetTimeCount ( time1 ,time2 ,MinuteU )
sBuff = GetTimeCount ( time1 ,time2 ,SecondU )
```

IsLeapYear（うるう年かどうかを調べる）

機能

指定した西暦年がうるう年かどうかを調べ，結果を真（True），または偽（False）で返します。

形式

IsLeapYear (Year)

指定項目

Year

西暦年を数値，または値を格納した変数名で指定します。

西暦年に 0～69 の整数値を指定すると，2,000～2,069 に読み替えられます。

説明

指定した西暦年がうるう年かどうかを調べ，結果をコマンドの実行結果として返します。うるう年である場合は真（True）を，うるう年でない場合は偽（False）を返します。

例

```
' 今年がうるう年かどうかを調べる。
Dim　nowYear
nowYear = Year ( Date() )
If　IsLeapYear ( nowYear ) Then
　Message( Target_DispOn, "実行結果", "今年はうるう年です。" )
Else
　Message( Target_DispOn, "実行結果", "今年はうるう年ではありません。" )
End If
```

6.5 ファイル・ディレクトリ操作コマンド

ファイル・ディレクトリ操作コマンドの詳細を説明します。

TextFileReplace（テキストファイルの文字列を置き換える）

機能

テキストファイルの中の，特定の文字列を置き換えます。

形式

TextFileReplace (FileName , OldText , NewText〔, ReplaceCntBuff〕)

指定項目

FileName

テキストファイル名を文字列，または値を格納した変数名で指定します。

OldText

変更前文字列を文字列，または値を格納した変数名で指定します。

NewText

変更後文字列を文字列，または値を格納した変数名で指定します。

ReplaceCntBuff

置き換えた文字列の数を受け取る変数名を指定します。必要がない場合は省略します。

説明

指定した変更前文字列をテキストファイルから検索し，別の文字列に置き換えます。コマンドが正常に実行された場合は真（True）を，エラーが発生した場合は偽（False）を，コマンドの実行結果として返します。

ラージファイルは，使用できません。

2,147,483,648 バイト以上のテキストファイルの文字列を置き換える場合は，SplitFile コマンドを使用してテキストファイルを 2,147,483,647 バイト以下に分割し，分割した各々のテキストファイルに対して TextFileReplace コマンドを実行してください。

その後，CatFiles コマンド使用して分割したファイルを統合してください。

注意事項

変換前文字列に改行コードを指定してもファイル中の改行コードは変換後文字列に変換されません。

例

＜実行前＞

```
' TextFileReplace のテスト
TextFileReplace ( _BIN_+"ABC.TXT" ,"テスト" ,"test" )
```

＜実行後＞

```
' TextFileReplace の test
TextFileReplace ( _BIN_+"ABC.TXT" ,"test" ,"test" )
```

TextOpen（テキストファイルをオープンする）

機能

テキスト形式ファイルをオープンします。JP1/Script07-51 以降では，ラージファイルが使用できます。

形式

TextOpen (FilePath 〔, Mode〕)

指定項目

FilePath

テキスト形式ファイル名のフルパスを文字列，または値を格納した変数名で指定します。

Mode

ファイルオープン時の動作を次の値で指定します。

値	意味
Create	ファイルを無条件に作成する。
ReadOnly	ファイルを読み込みモードでオープンする。
WriteOnly	ファイルを書き込みモードでオープンする。
ReadWrite	ファイルを読み書きモードでオープンする。

Mode に ReadOnly を指定した場合は共用モードで，Cretate，WriteOnly，および ReadWrite を指定した場合は排他モードでオープンします。ただし，HP-UX，Solaris，Linux の場合は，スクリプト内で使用するファイルだけ排他になります。

この値は省略できます。省略した場合，Create を仮定します。

説明

指定したテキスト形式ファイルをオープンします。コマンドが正常に実行された場合はファイル識別子を，エラーが発生した場合は 0 を，コマンドの実行結果として返します。

またコマンドが正常に実行された場合は，現在の読み書き開始位置は先頭の 0 になります。

例

```
' _BIN_+ABC.TXT ファイルをオープンし， 正常なら 1 レコード入力する。
Dim　file1
```

```
file1 = TextOpen ( _BIN_+"ABC.TXT" ,ReadOnly )
If   file1 = 0   Then
   Message( Target_DispOn, "実行結果", _BIN_+"ABC.TXT のオープンに失敗" )
Else
   Dim   buff1
   If   TextRead ( file1 ,buff )   Then
      Message( Target_DispOn, "実行結果", buff )
   End
   TextClose ( file1 )
End
```

TextClose（テキストファイルをクローズする）

機能

テキスト形式ファイルをクローズします。JP1/Script 07-51 以降では，ラージファイルが使用できます。

形式

TextClose (〔FileId〕)

指定項目

FileId

ファイル識別子を数値，または値を格納した変数名で指定します。
このファイル識別子は，TextOpen コマンドで実行結果として返される値です。
この値は省略できます。省略した場合，オープン中のファイルをすべてクローズします。また，この値が 0 の場合も，オープン中のファイルをすべてクローズします。

説明

指定したファイル識別子のテキスト形式ファイルをクローズします。コマンドが正常に実行された場合は真（True）を，エラーが発生した場合は偽（False）を，コマンドの実行結果として返します。

例

```
' _BIN_+ABC.TXT ファイルと_TEMP_+Backup.TXT ファイルをオープンする。
Dim   file1 ,file2
```

```
file1 = TextOpen ( _BIN_+"ABC.TXT" ,ReadOnly )
If   file1 = 0   Then
   Message( Target_DispOn, "実行結果", _BIN_+"ABC.TXT のオープンに失敗" )
   Exit
End
file2 = TextOpen ( _TEMP_+"Backup.TXT" ,Create )
If   file2 = 0   Then
   Message( Target_DispOn, "実行結果",_TEMP_+"Backup.TXT のオープンに失敗" )
   TextClose ( file1 )
   Exit
End

' _BIN_+ABC.TXT ファイルの 1 レコードを_TEMP_+Backup.TXT ファイルにコピーする。
Dim   buff1
If   TextRead ( file1 ,buff1 )   Then
   TextWrite ( file2 ,buff1 )
End
TextClose ( file1 )
TextClose ( file2 )
```

TextRead（テキストファイルから 1 行分のデータを読み込む）

機能

テキスト形式ファイルから 1 行分のデータを読み込みます。JP1/Script 07-51 以降では，ラージファイルが使用できます。

形式

TextRead (FileId , Buff)

指定項目

FileId

ファイル識別子を数値，または値を格納した変数名で指定します。

このファイル識別子は，TextOpen コマンドで実行結果として返される値です。

なお，この項目を指定する場合は TextOpen コマンドのファイルオープン時の動作には，WriteOnly（ファイルを書き込みモードでオープン）以外を指定してください。

Buff

読み込まれた 1 行分のデータを受け取る変数名を指定します。受け取るデータの文字数が半角文字で 1,024 文字を超える場合，1,024 文字で切り捨てられます。

説明

指定したファイル識別子のテキスト形式ファイルの現在の読み書き開始位置から 1 行分のデータを読み込み，指定した変数に格納します。コマンドが正常に実行された場合は真（True）を，エラーが発生した場合は偽（False）を，コマンドの実行結果として返します。

コマンドの実行結果が真（True）の場合は，読み書き開始位置は次行になります。

また，ファイルの終わりに達した場合は，コマンドの実行結果に偽（False）を返し，_RTN_予約変数には_ERR_EOF_予約変数と等しい値が設定されます。しかし，On Error ステートメントで指定したラベルには分岐しません。

注意事項

TextRead コマンドでファイルの終わりに達した場合，JP1/Script ではコマンド実行時に異常が発生した場合と同じようにエラーとして処理します。また，TextRead コマンドで読み込まれた1行分のデータを受け取る変数には，直前まで格納していた値が格納されているので，EOF 検出後に変数内容が未設定になる状態を期待する処理は作り込まないでください。

TextRead で読込んだ行に BOM（Byte Order Mark）が付いている場合は BOM は読み捨てられます。

例

```
' カレントディレクトリ下のテキストファイル"ABC.TXT"の内容を,
' 以下のスクリプトで 1 行ずつ読み込み，内容を表示する。

Dim   file1 ,buff1 ,readRtn

On Error GoTo ErrorBranch
   file01 = TextOpen ( _BIN_+"ABC.TXT" ,ReadOnly )
   TextRead ( file1 ,buff1 )
   readRtn = _RTN_
   while   readRtn <> _ERR_EOF_
      Message( Target_DispOn, "実行結果", _buff1 )
      TextRead ( file1 ,buff1 )
      readRtn = _RTN_
   End

TextClose ( file1 )
Exit ( 0 )

ErrorBranch:
   Message( Target_DispOn, "実行結果", _BIN_+"ABC.TXT のファイル操作失敗" )
   TextClose ( file1 )
   Exit ( -1 )
```

TextWrite（テキストファイルにデータを書き込む）

機能

テキスト形式ファイルにデータを書き込みます。JP1/Script 07-51 以降では，ラージファイルが使用できます。

形式

TextWrite (FileId , Data ,　〔NewLine〕　〔, Position〕　)

指定項目

FileId

ファイル識別子を数値，または値を格納した変数名で指定します。

このファイル識別子は，TextOpen コマンドで実行結果として返される値です。

なお，この項目を指定する場合は，TextOpen コマンドのファイルオープン時の動作には，ReadOnly（ファイルを読み込みモードでオープン）以外を指定してください。

Data

書き込むデータを文字列，または値を格納した変数名で指定します。

NewLine

データを書き込んだあとを改行する場合には真（True）を，改行しない場合には偽（False）を指定します。

この値は省略できます。省略した場合，真（True）を仮定します。

Position

ファイルの先頭を 0 とした読み書き開始位置を数値，または値を格納した変数名で指定します。

この値は省略できます。省略した場合，現在の読み書き開始位置を仮定します。

説明

指定したファイル識別子のテキスト形式ファイルに対しデータを現在の読み書き開始位置，または指定した読み書き開始位置から書き込みます。コマンドが正常に実行された場合は真（True）を，エラーが発生した場合は偽（False）を，コマンドの実行結果として返します。

読み書き開始位置は，コマンドの実行結果が真（True）で NewLine に真（True）が指定してある場合は次行になり，NewLine に偽（False）が指定してある場合は書き込んだデータの直後になります。

注意事項

すでに存在するファイルに対して TextOpen コマンドの Mode オペランドに WriteOnly，または ReadWrite を指定し，TextWrite コマンドを実行した場合，TextWrite コマンドは，単純に指定された文字数分の上書きを行います。したがって，出力する文字数（改行コードを含む）が既存ファイルより少ない場合，少ない文字数分オープン前の内容がファイルに残ります。

TextWrite コマンドは BOM（Byte Order Mark）を付加して出力することはできません。また，BOM 付きのファイルの先頭行を上書きする場合は BOM を上書きします。

例

次のようなスクリプトでカレントディレクトリ下のテキストファイル"ABC.TXT"にデータを書き込む。

```
' スクリプトファイル
Dim　file1
file1 = TextOpen ( _BIN_+"ABC.TXT" ,Create )
If　file1 = 0　Then
　Message( Target_DispOn, "実行結果", _BIN_+"ABC.TXT のオープンに失敗" )
Else
　TextWrite ( file1 ,"JP1/Script についての追加情報" )
　TextWrite ( file1 ,"                    目次" ,False )
　TextClose ( file1)
End

; ABC.TXT
JP1/Script についての追加情報
```

TextSeek（読み書き開始位置を先頭または末尾に移動する）

機能

テキスト形式ファイルの読み書き開始位置をファイルの先頭，または最後に移動します。
JP1/Script 07-51 以降では，ラージファイルが使用できます。

形式

TextSeek (FileId 〔, Point〕)

指定項目

FileId

ファイル識別子を数値，または値を格納した変数名で指定します。

このファイル識別子は，TextOpen コマンドで実行結果として返される値です。

なお，この項目を指定する場合は，TextOpen コマンドのファイルオープン時の動作には，WriteOnly（ファイルを書き込みモードでオープン）以外を指定してください。

Point

読み書き開始位置を次の値で指定します。この値は省略できます。省略した場合，ToBegin を仮定します。

値	意味
ToBegin	ファイルの先頭に移動します。
ToEnd	ファイルの最後に移動します。

説明

指定したファイル識別子のテキスト形式ファイルに対し読み書き開始位置を指定した位置に移動します。コマンドが正常に実行された場合は真（True）を，エラーが発生した場合は偽（False）を，コマンドの実行結果として返します。

例

```
' カレントディレクトリ下のテキストファイル"ABC.TXT"の最後に"** END OF FILE **"と書き込む。
Dim　file1
file1 = TextOpen ( _BIN_+"ABC.TXT" ,ReadWrite )
If　file1 = 0　Then
　Message( Target_DIspOn, "実行結果" ,_BIN_+"ABC.TXT のオープンに失敗" )
```

```
Else
  If  TextSeek ( file1 ,ToEnd )  Then
    TextWrite ( file1 ,"** END OF FILE **" ,False )
  End
  TextClose ( file1 )
End
```

GetTextPosition（読み書きの開始位置を返す）

機能

テキスト形式ファイルの現在の読み書き開始位置を返します。

形式

GetTextPosition (FileId)

指定項目

FileId

ファイル識別子を数値，または値を格納した変数名で指定します。
このファイル識別子は，TextOpen コマンドで実行結果として返される値です。

説明

指定したファイル識別子のテキスト形式ファイルの，現在の読み書き開始位置を取得します。取得する値は，先頭を 0 とします。コマンドが正常に実行された場合は現在の読み書き開始位置を，エラーが発生した場合は長さ 0 の文字列（""）を，コマンドの実行結果として返します。

JP1/Script 07-51 以降では，ラージファイルの使用が可能です。ただし，位置づいている場所が，2,147,483,647 バイト以下の場合は，正常に終了しますが，2,147,483,648 バイト以上の場合は，エラーとなります。

例

```
' カレントディレクトリ下のテキストファイル"ABC.TXT"から読み込んだデータが,
' "JP1/Script について"であれば"JP1/Script の説明"に書き換える。

Dim  file1
file1 = TextOpen ( _BIN_+"ABC.TXT" ,ReadWrite )
```

```
If   file1 = 0   Then
   Message( _Target_DispOn, "実行結果" ,_BIN_+"ABC.TXT のオープンに失敗" )
Else
   Dim   line ,position ,buff
   For   line = 1 To 10
      position = GetTextPosition ( file1 )
      If   TextRead ( file1 ,buff )   Then
         If   buff = "JP1/Script について"   Then
            TextWrite ( file1 ," JP1/Script の説明  " ,True ,position )
         End
      Else
         Exit For
      End
   Next
   TextClose ( file1 )
End
```

MakeDir（ディレクトリを作成する）

機能

ディレクトリを作成します。

形式

MakeDir (DirPath)

指定項目

DirPath

作成するディレクトリパスを文字列，または値を格納した変数名で指定します。

説明

指定したディレクトリを作成します。コマンドが正常に実行された場合は真（True）を，エラーが発生した場合は偽（False）を，コマンドの実行結果として返します。

指定したディレクトリまでのパスが存在しない場合でも，途中のパスを作成してディレクトリを作成します。

例

' WORK ディレクトリを作成する。

MakeDir ("WORK")

DeleteDir（ディレクトリを削除する）

機能

ディレクトリを削除します。

形式

DeleteDir (DirPath 〔, Option〕)

指定項目

DirPath

削除するディレクトリパスを文字列，または値を格納した変数名で指定します。

Option

オプションを次の値で指定します。この値は省略できます。省略した場合，Anyway を仮定します。

値	意味
Anyway	ディレクトリの中身が空でも空でなくても，中身ごとディレクトリを削除します。
IfEmpty	ディレクトリの中身が空の場合はディレクトリを削除し，空でない場合はディレクトリを削除しません。 なお，ディレクトリを削除しなかった場合，偽（False）をコマンドの実行結果として返します。

説明

指定したディレクトリを削除します。コマンドが正常に実行された場合は真（True）を，エラーが発生した場合は偽（False）を，コマンドの実行結果として返します。

また，指定したディレクトリが存在しない場合は，常に真（True）を返します。

注意

オプションに IfEmpty の指定がない場合，ディレクトリの中身が空でなくても，無条件に中身ごとディレクトリを削除してしまいます。指定する場合は十分に注意してください。

例

"WORK"ディレクトリを無条件に削除する。

DeleteDir ("WORK")

'"WORK"ディレクトリ下が空であれば削除する。

DeleteDir ("WORK" ,IfEmpty)

DeleteFile（ファイルを削除する）

機能

ファイルを削除します。JP1/Script 07-51 以降では，ラージファイルが使用できます。

形式

DeleteFile (PathName 〔, Option〕)

指定項目

PathName

削除するディレクトリ，またはファイルのフルパスを文字列，または値を格納した変数名で指定します。ファイル名にワイルドカードを指定することもできます。

指定されたディレクトリ名が複数ディレクトリにわたっている場合は，指定された文字列の末尾に該当するディレクトリが削除の対象になり，上位ディレクトリは対象になりません。

Option

オプションを次の値で指定します。

値	意味
ExclDir	PathName で指定したディレクトリ下のファイルだけを削除し，ディレクトリは残します。

説明

指定したディレクトリ，またはファイルを削除します。オプションが省略された場合，ディレクトリの中身が空ではなくても，中身ごとディレクトリを削除します。オプションに ExclDir が指定されている場合は，指定したディレクトリ下のファイルだけを削除します。

コマンドが正常に実行された場合は真（True）を，エラーが発生した場合は偽（False）を，コマンドの実行結果として返します。

ただし，「指定したファイルが見つからない」（_ERR_FILE_予約変数と等しい値）エラーが発生した場合は，常に真（True）を返します。

例

' 一時ディレクトリ下のファイル"tempfile.tmp"を削除する。

DeleteFile (_TEMP_+"tempfile.tmp")

' 一時ディレクトリ下の拡張子が".tmp"であるファイルを削除する。

DeleteFile (_TEMP_+"*.tmp")

' 一時ディレクトリ下のすべてのファイルを削除し,一時ディレクトリは残す。

DeleteFile (_TEMP_ ,ExclDir)

Rename(ファイル名を変更する)

機能

ファイル名を変更します。JP1/Script 07-51 以降では，ラージファイルが使用できます。

形式

Rename (OldName , 〔NewName〕 〔, Method〕)

指定項目

OldName

変更前ファイル名を文字列，または値を格納した変数名で指定します。

NewName

変更後ファイル名を文字列，または値を格納した変数名で指定します。

Method

変更方法を次の値で指定します。この値は省略できます。省略した場合，Replace が仮定されます。

値	意味
Replace	NewName で指定した名称のファイルがすでに存在する場合もファイルを置き換えます。
NoReplace	NewName で指定した名称のファイルがすでに存在する場合はエラーになります。
FreeExt	拡張子を，.000～.999 の範囲の中で空いているものに変更します。このオプション使用時は，変更後ファイル名を指定しても無視されます。

説明

OldName で指定したファイルを *NewName* のファイル名に変更します。コマンドが正常に実行された場合は真(True)を，エラーが発生した場合は偽(False)を，コマンドの実行結果として返します。

例

```
' ファイル名を変更する。
Dim　outDir1
outDir1 = _BIN_
Rename ( outDir1+"DEFAULT.DAT" ,outDir1+"DEFAULT.000" ,FreeExt )
Rename ( _TEMP_+"WORK.DAT" ,outDir1+"DEFAULT.DAT" )
```

TempDir（一時ディレクトリを取得する）

機能

一時ディレクトリを取得します。

形式

TempDir (DirNameBuff)

指定項目

DirNameBuff

一時ディレクトリを受け取る変数名を指定します。

説明

システムの一時ディレクトリを取得し，変数に格納します。コマンドが正常に実行された場合は真（True）を，エラーが発生した場合は偽（False）を，コマンドの実行結果として返します。

ディレクトリ名の後ろにはスラント（ / ）が付きます。

TempFile（一時ファイル名を作成する）

機能

一時ファイル名を作成します。

形式

TempFile (FileNameBuff ,〔Prefix〕〔, DirName〕)

指定項目

FileNameBuff

作成した一時ファイル名を受け取る変数名を指定します。

Prefix

プレフィックスを文字列，または値を格納した変数名で指定します。プレフィックスは先頭の 3 文字が有効となります。
この値は省略できます。省略した場合，"STX"が仮定されます。

DirName

一時ファイルを作成するディレクトリ名を文字列，または値を格納した変数名で指定します。
この値は省略できます。省略した場合，システムの一時ディレクトリを仮定します。

説明

一時ファイル名を作成します。コマンドが正常に実行された場合は真（True）を，エラーが発生した場合は偽（False）を，コマンドの実行結果として返します。このコマンドは，作成されたファイル名（拡張子は「TMP」）でファイルサイズ 0 バイトのファイルを実際に作成します。

例

```
Dim    bkupFileName ,outDir1
outDir1 = _TEMP_
TempFile ( bkupFileName ,"BUP" ,outDir1 )
Copy ( outDir1+"MODEM" ,bkupFileName )
```

SetFileTime（ファイルの日付と時刻を設定する）

機能

ファイルの日付と時刻を設定します。JP1/Script 07-51 以降では，ラージファイルが使用できます。

形式

SetFileTime (PathName , 〔Date〕 , 〔Time〕 〔, Type〕)

指定項目

PathName

設定するファイルのフルパスを文字列，または値を格納した変数名で指定します。

Date

設定する日付を yyyy/mm/dd の形式の文字列，または値を格納した変数名で指定します。yyyy に 0～69 の範囲の整数値を指定すると，2,000～2,069 に読み替えられます。必要がない場合は省略します。

Time

設定する時刻を hh:mm:ss の形式の文字列，または値を格納した変数名で指定します。必要がない場合は省略します。

Type

設定するファイル日付の種別を次の値で指定します。省略した場合，Update を仮定します。

値	意味
Update	更新日付

説明

指定したファイルに日付と時刻を設定します。指定できるファイルの日付は1970年1月1日　9時0分0秒～2038年1月19日　3時14分07秒の範囲内です。なお，この範囲はタイムゾーンに日本標準時間を設定している場合です。このため *Date* の yyyy に 39～69 を指定した場合，2039～2069 に読み換えられますので日付の範囲外のエラーになります。

コマンドが正常に実行された場合は真（True）を，エラーが発生した場合は偽（False）を，コマンドの実行結果として返します。また，Date と Time をともに省略した場合は，指定したファイルには何も設定しないで，常に真（True）を返します。指定したファイルがディレクトリの場合，偽（False）を返します。

例

```
Dim　file1
```

```
file1 = _SCF_+"User.TXT"
SetFileTime ( file1 ,Date ,Time ,Update )
```

GetFileTime（ファイルの日付と時刻を取得する）

機能

ファイルの日付と時刻を取得します。JP1/Script 07-51 以降では，ラージファイルが使用できます。

形式

GetFileTime (PathName ， 〔DateBuff〕 ， 〔TimeBuff〕 〔, Type〕)

指定項目

PathName

取得するファイルのフルパスを文字列，または値を格納した変数名で指定します。

DateBuff

日付を受け取る変数名を指定します。日付は yyyy/mm/dd の形式で返します。必要がない場合は省略します。

TimeBuff

時刻を受け取る変数名を指定します。時刻はhh:mm:ssの形式で返します。必要がない場合は省略します。

Type

取得するファイル日付の種別を次の値で指定します。省略した場合，Update を仮定します。

値	意味
Update	更新日付

説明

指定したファイルの日付と時刻を取得します。コマンドが正常に実行された場合は真（True）を，エラーが発生した場合は偽（False）を，コマンドの実行結果として返します。

例

```
Dim   file1 ,file2 ,dateBuff ,timeBuff
file1 = _SCF_+"User.TXT"
file2 = _SCF_+"UserBkup.TXT"
If   GetFileTime ( file1 ,dateBuff ,timeBuff ,Update ) = TRUE   Then
```

```
        SetFileTime ( file2 ,dateBuff ,timeBuff ,Update )
    End
```

GetFileSize（ファイルの容量を取得する）

機能

ファイルの容量を取得します。

形式

GetFileSize (PathName 〔, UnitofByte〕)

指定項目

PathName

取得するファイルのフルパスを文字列，または値を格納した変数名で指定します。

UnitofByte

取得する容量の単位を次の値で指定します。この値は省略できます。省略した場合，Byte を仮定します。

値	意味
Byte	バイト
KB	キロバイト
MB	メガバイト

UnitofByte に KB，または MB が指定した場合，切り上げた値を返します。例えば，UnitofByte に KB が指定され，容量が 1 キロバイトに満たない場合は，1 を返します。

説明

指定したファイルの容量を取得し，コマンドの実行結果として返します。指定したファイルが存在しない場合は，0 を返します。
ディレクトリに対して GetFileSize を発行した場合，ディレクトリの容量に 0 以上の値（AIX の場合は 2,048 バイト）を返します。
JP1/Script 07-51 以降では，ラージファイルが使用できます。ただし，ラージファイルでも指定された容量の単位に変換した結果が，JP1/Script で扱える数値の上限値 2,147,483,647 を超えると，エラーとなります。なお，2,251,799,812,636,672 バイト以上の容量のファイルに対しては，GetFileSize コマンドを使用することはできません。

例

```
` コピー元ファイルの容量を調べて,コピー先ディスクに書き込む空き容量があれば
```

```
` ファイルをコピーする。
Dim   path1 ,path2 ,fileSz ,freeSz
path1 = _SCF_+"Bkup.TXT"
path2 = "/WORK"
fileSz = GetFileSize ( path1 )
freeSz = GetDiskFreeSpace ( path2 ,KB )
If   ( freeSz * 1024 ) >= fileSz   Then
        Copy ( path1 ,path2 )
End If
```

SplitFile（ファイルを分割する）

機能

ファイルを指定したサイズで分割します。

形式

SplitFile (FilePath , SplitSize , 〔DirPath〕 , 〔Option〕 〔, SplitCnt〕)

指定項目

FilePath

分割するファイルのフルパスを文字列，または値を格納した変数名で指定します。

SplitSize

分割するサイズを数値，または値を格納した変数名で指定します。サイズはバイト単位で指定します。

DirPath

分割したファイルを格納するディレクトリ名を文字列，または値を格納した変数名で指定します。

この値は省略できます。省略した場合，*FilePath* で指定したファイルが存在するディレクトリを仮定します。

Option

オプションを次の値で指定します。

値	意味
Delete	ファイルを分割したあと，FilePath で指定したファイルを削除します。

SplitCnt

分割したファイル数を受け取る変数名を指定します。必要がない場合は省略します。

説明

指定したファイルを指定したサイズで分割し，分割したファイルを指定したディレクトリに格納します。コマンドが正常に実行された場合は真（True）を，エラーが発生した場合は偽（False）を，コマンドの実行結果として返します。

分割したファイルのファイル名は *FilePath* で指定したファイル名の末尾に拡張子.XXX（XXX は 001 からの数字で，999 を超えた場合 1,000 からの数字）を付けた値になります。

JP1/Script 07-51 以降では，ラージファイルが使用できます。ただし，分割するサイズに 2,147,483,648 バイト以上は，指定できません。

例

```
' ファイル"TESTFILE.TXT"を分割して一時ディレクトリ下に"TESTFILE.TXT.001"
' から splCnt 個の分割されたファイルを作成し,別のディレクトリに格納する。
' (参考）1.44MB の FD = 1423KB = 1423 * 1024B
Dim   file1 ,size1 ,splCnt ,filePath
file1 = "TESTFILE.TXT"
size1 = 1423 * 1024
SplitFile ( file1 ,size1 ,_TEMP_ , ,splCnt )

Dim   cnt1
For   cnt1 = 1 To splCnt
  ' 分割後ファイル名の作成
  filePath = _TEMP_+file1
  Select Case   Len ( cnt1 )
    Case 1
      filePath = filePath+".00"+cnt1
    Case 2
      filePath = filePath+".0"+cnt1
    Case Else
      filePath = filePath+"."+cnt1
  End Select

  ' 一時ディレクトリから別ディレクトリへコピー
 Copy ( filePath ,"/work/" ,Overwrite )
```

```
    If   splCnt - cnt1 <= 0   Then
      Message ( Target_Dispon ,"実行状況" ,"複写が終了しました。" )
      Exit For
    End
  Next
```

CatFiles（ファイルを統合する）

機能

ファイルを統合します。JP1/Script 07-51 以降では，ラージファイルが使用できます。

形式

CatFiles (PathName , 〔Option〕 , FilePath1 , 〔FilePath2〕〔, FilePath3 , ..., FilePath10 〕)

指定項目

PathName

統合したファイルを格納するディレクトリ，またはファイル名を文字列，または値を格納した変数名で指定します。指定したファイルがすでに存在している場合は上書きします。

Option

オプションを次の値で指定します。

値	意味
Delete	ファイルを統合後，FilePathn で指定したファイルを削除します。

FilePath1～10

統合するファイルのフルパスを文字列，または値を格納した変数名で指定します。ファイル名にワイルドカードを指定することもできます。

ファイル名は 10 個まで指定できます。指定した順序で統合します。

複数のファイルの，フルパスの文字列を一つずつ格納した一次元の配列変数を変数名で指定することもできます。この場合は，11 個以上のファイル名を指定できます。

ファイル名にワイルドカードを指定した場合は，該当するファイル名を昇順にソートして統合します。

説明

指定したファイルを統合し，統合したファイルを指定したディレクトリに格納します。コマンドが正常に実行された場合は真（True）を，エラーが発生した場合は偽（False）を，コマンドの実行結果として返します。

例

```
' ディレクトリ（/work）に格納されたファイル"/TESTFILE.TXT.*"を統合して,
' カレントディレクトリの"TESTFILE.TXT"を作成する。
Dim　dir1 ,file1
dir1 = _Scf_
file1 = "TESTFILE.TXT"
CatFiles ( dir1+file1 , , "/WORK/"+file1+"*" )
```

SetStandardFile または SetStdFile（標準入力，標準出力，および標準エラーファイルを設定する）

機能

Exec コマンドで呼び出すプロセスの標準入力，標準出力，および標準エラーファイルをオープンして設定します。JP1/Script 07-51 以降では，ラージファイルが使用できます。

形式

SetStandardFile (FilePath , Type 〔, Mode〕)

SetStdFile (FilePath ,Type 〔, Mode〕)

指定項目

FilePath

設定するファイルのフルパスを文字列，または値を格納した変数名で指定します。

Type

設定するファイルの種別を次の値で指定します。

値	意味
StdInput	標準入力
StdOutput	標準出力
StdError	標準エラー

Mode

Type で指定したファイルの種別が StdOutput，または StdError の場合，ファイル作成時の動作を指定します。ファイル作成時の動作を次の値で指定します。この値は省略できます。省略した場合，Append を仮定します。

値	意味
Create	新規モード
Append	追加モード

Type に StdInput を指定した場合は常に新規モードとなるため，Mode に指定した値は無効になります。

説明

このコマンドを実行したあとに Exec コマンドで呼び出すプロセスの標準入力，標準出力，および標準エラーファイルをオープンして設定します。コマンドが正常に実行された場合は真（True）を，エラーが発生した場合は偽（False）を，コマンドの実行結果として返します。

設定されたファイルは ResetStandardFile コマンドが実行されるまで有効になり，オープンされたままです。ResetStandardFile コマンドの詳細は「6.5 ファイル・ディレクトリ操作コマンド」の「ResetStandardFile または ResetStdFile」を参照してください。

例

```
SetStandardFile ( _SCF_+"EXECOUT.TXT" ,StdOutput ,Create )
SetStandardFile ( _SCF_+"EXECERR.TXT" ,StdError )
If   Exec ( _SCF_+"BACKUP" ,True )   Then
   Message( Target_DispOn ,"実行結果" ,"Exec コマンド 成功！" )
Else
   Message( Target_DispOn ,"実行結果" ,"Exec コマンド 失敗！" )
End
```

ResetStandardFile または ResetStdFile（標準入力，標準出力，標準エラーファイルを解除する）

機能

Exec コマンドで呼び出すプロセスの標準入力，標準出力，および標準エラーファイルをクローズして解除します。JP1/Script 07-51 以降では，ラージファイルが使用できます。

形式

ResetStandardFile (〔Type〕)

ResetStdFile (〔Type〕)

指定項目

Type

設定を解除するファイルの種別を，次の値で指定します。この値は省略できます。省略した場合，設定されているすべてのファイルを解除します。

値	意味
StdInput	標準入力
StdOutput	標準出力
StdError	標準エラー

説明

SetStandardFile コマンドで設定されたプロセスの標準入力，標準出力，および標準エラーファイルをクローズして解除します。コマンドが正常に実行された場合は真（True）を，エラーが発生した場合は偽（False）を，コマンドの実行結果として返します。

例

```
SetStandardFile ( _SCF_+"BatOut1.TXT" ,StdOutput )
SetStandardFile ( _SCF_+"BatErr.TXT" ,StdError )
Exec ( _SCF_+"Backup" ,True )

ResetStandardFile ( StdOutput )
SetStandardFile ( _SCF_+"BatOut2.TXT" ,StdOutput )
Exec ( _SCF_+"Build" ,True )
```

SplitPath（フルパスを解析する）

機能

フルパスを解析します。

形式

SplitPath (FullPath ,〔DrvNameBuff〕,〔DirNameBuff 〕,〔LblNameBuff〕〔, ExtNameBuff〕)

指定項目

FullPath

解析するフルパスを文字列，または値を格納した変数名で指定します。

DrvNameBuff

このシステムでは未支援のため，変数には何も設定されません。

DirNameBuff

ディレクトリ名を受け取る変数名を指定します。必要がない場合は省略します。

LblNameBuff

ファイルラベル名を受け取る変数名を指定します。必要がない場合は省略します。

ExtNameBuff

拡張子を受け取る変数名を指定します。必要がない場合は省略します。

説明

与えられたフルパスを，ディレクトリ名，ファイルラベル名，および拡張子に分解して各変数に格納します。コマンドが正常に実行された場合は真（True）を，エラーが発生した場合は偽（False）を，コマンドの実行結果として返します。

ディレクトリ名には前後にスラント（ / ）が付きます。拡張子にはピリオド（ . ）が付きます。

例

```
Dim    drv1 ,dir1 ,lbl1 ,ext1
SplitPath ( "/WORK/WORKFILE.TXT" ,drv1 ,dir1 ,lbl1 ,ext1 )
```

↓

drv1　空

dir1　/WORK/

lbl1　WORKFILE

ext1　.TXT

MakePath（フルパスを作成する）

機能

フルパスを作成します。

形式

MakePath (FullPathBuff , DrvName , DirName , LblName , ExtName)

指定項目

FullPathBuff

作成したフルパスを受け取る変数名を指定します。

DrvName

このシステムでは未支援のため，無視されます。

DirName

ディレクトリ名を文字列，または値を格納した変数名で指定します。

LblName

ファイルラベル名を文字列，または値を格納した変数名で指定します。

ExtName

拡張子を文字列，または値を格納した変数名で指定します。

説明

ディレクトリ名，ファイルラベル名，拡張子を結合してフルパスを作成し，変数に格納します。コマンドが正常に実行された場合は真（True）を，エラーが発生した場合は偽（False）を，コマンドの実行結果として返します。

ディレクトリ名の前後のスラント(/)，および拡張子のピリオド(.)は，省略しても自動的に付きます。

例

```
SplitPath    (OrginalFile   Drv   Dir   Lbl   Ext)
MakePath     (BackupFile   Drv   Dir   Lbl   ".BAK")
```

SetPath(実行ディレクトリのパスを設定する)

機能

カレントディレクトリのパスを設定します。

形式

SetPath (〔DirPath〕)

指定項目

DirPath

設定するディレクトリパスを文字列，または値を格納した変数名で指定します。
この値は省略できます。省略した場合，カレントディレクトリを仮定します。

説明

指定したディレクトリパスをカレントディレクトリのパスに設定します。_BIN_予約変数も指定したディレクトリパスに変更されます。

コマンドが正常に実行された場合は真(True)を，エラーが発生した場合は偽(False)を，コマンドの実行結果として返します。

指定したディレクトリパスが存在しない場合は，何もしないで，常に偽(False)を返します。

例

```
OutDir = %1
If   IsEmptyDir ( OutDir )   Then
     SetPath ( OutDir )        ' カレントディレクトリを設定する。
End
Exec ( OutDir+"ABC" ,TRUE )
SetPath                        ' カレントディレクトリをカレントに戻す。
```

GetPath（実行ディレクトリのパスを取得する）

機能

実行ディレクトリのパスを取得します。

形式

GetPath

このコマンドに引数はありません。

説明

現在の実行ディレクトリのパスを取得し，値をコマンドの実行結果として返します。

ディレクトリ名の末尾にはスラント（ / ）は付きません。ディレクトリの末尾にスラント（ / ）を付加する場合は，システム環境ファイル（jp1script.cf）の GetPath_Opt に 1 を設定してください。

例

```
If   GetPath<> _BIN_   Then
        SetPath
End
```

GetDiskFreeSpace（ディスク空き容量を取得する）

機能

ディスクの空き容量を取得します。

形式

GetDiskFreeSpace (DiskName 〔, UnitofByte〕)

指定項目

DiskName

取得するマウントしたディレクトリを文字列，または値を格納した変数名で指定します。

UnitofByte

取得する容量の単位を次の値で指定します。この値は省略できます。省略した場合，KB を仮定します。

値	意味
KB	キロバイト
MB	メガバイト

空き容量は切り捨てた値を返します。例えば，UnitofByte に MB が指定され，容量が 1 メガバイトに満たない場合は，0 を返します。

説明

指定したディレクトリの空き容量を取得し，コマンドの実行結果として返します。
空き容量が 1 キロバイト，または 1 メガバイトに満たない場合は，0 を返します。指定したディレクトリが存在しない場合も，0 を返します。

例

```
Dim    MBsize, GBsize
MBsize = GetDiskFreeSpace ( "/tmp" ,MB )
KBsize = MBsize / 1024
Message( Target_DispOn, "/tmp の空き容量は"+ GBsize +"ギガバイトです。" )
```

注意事項

存在しないパスを指定した場合は，エラーになります。

パスを指定した場合も，その指定したパスのマウントしたディレクトリのサイズが返ります。

Copy（ファイルをコピーする）

機能

ファイルをコピーします。JP1/Script 07-51 以降では，ラージファイルが使用できます。

形式

Copy (OldFileName , NewPathName , 〔Option1〕 , 〔Option2〕 , 〔Option3〕 , 〔Option4〕 , 〔Option5〕 , 〔ExceptFileName〕 〔, Option6〕 〔, Option7〕)

指定項目

OldFileName

コピー元（移動元）ファイル名を文字列，または値を格納した変数名で指定します。

指定したファイル名の最後の２文字が.Zの場合，COMPRESSコマンドで圧縮されたファイルとみなし，展開しながらコピーします。ただし，NewPathName でコピー先ファイル名の指定がないかぎり，ファイル名は変更しません。ファイル名にワイルドカードを指定することもできます。

NewPathName

コピー先（移動先）ディレクトリ名，またはファイル名を，文字列，または値を格納した変数名で指定します。

Option1～5

コピーオプションを次の値で指定します。オプションは五つまで複数指定できます。この値は省略できます。省略した場合は，Overwrite を仮定します。

値	意味
VersionUp	新しいファイルだけを上書きします。このオプションでは，ファイル日付だけを比較します。
Overwrite	すべて上書きします。
NoOverwrite	コピー先ディレクトリにすでに存在するファイルはコピーしません。
OverwriteOnly	コピー先ディレクトリに存在するファイルだけをコピーします。
Pile	既存のファイルは拡張子を変更して残し，コピーします。
Move	ファイルを移動します。コピーしません。 これ以外の Versionup，Overwrite，NoOverwrite とともに指定でき，移動先ディレクトリにすでにファイルが存在する場合はともに指定したオプションに従って移動します。
SubDirToo	下位のディレクトリまで，その構造を保ったままコピーします。 ファイルが存在しない下位のディレクトリはコピーしません。 これ以外のオプションとともに指定できます。
Trace	コピーしたファイル名を実行トレースファイルに出力します。これ以外のオプションと同時に指定できます。
ErrSkip	コピー元ファイルの排他エラーやアクセスエラーが発生しても無視してコピーを続行します。無視したファイルの数は_COPY_SKIP_CNT_予約変数に格納します。これ以外のオプションと同時に指定できます。

ExceptFileName

OldFileName で指定したファイル名の中でコピー（または移動）の対象にしないファイルがある場合，そのファイル名を文字列，または値を格納した変数名で指定します。

複数のファイル名を指定する場合は，セミコロン（;）で区切った文字列，または値を格納した変数名で指定します。

複数のファイル名の文字列を一つずつ格納した一次元の配列変数を変数名で指定することもできます。ファイル名にワイルドカードを指定することもできます。

Option6～7

コピーオプションを次の値で指定します。このオプションは省略できます。

値	意味
ErrSkip2	コピー先ファイルの排他エラーやアクセスエラーが発生しても無視してコピーを続行します。無視したファイルの数は_COPY_SKIP2_CNT_予約変数に格納します。これ以外のほかのオプションとともに指定できます。

説明

指定したコピーオプションでファイルをコピーします。コマンドが正常に実行された場合は真(True)を，エラーが発生した場合は偽(False)を，コマンドの実行結果として返します。
コマンドの実行結果が真(True)の場合，コピーの結果は_COPY_RTN_予約変数に格納されます。格納される値とその意味を次に示します。

値	意味
Overwrite	ファイルのバージョンをチェックしないでコピーした。
VersionUp	ファイルのバージョンをチェックしてコピーした。
Pile	既存ファイルの拡張子を変更して残し，コピーした。
Move	ファイルを移動した。
Skip	ファイルをコピーしなかった。

コピーしたファイルの数は_COPY_CNT_予約変数に格納されます。
コピーしなかったファイルの数は_COPY_SKIP_CNT_予約変数に格納されます。

例

```
Dim InDir ,OutDir
InDir  = _SCF_+"Inst/"
OutDir = _SCF_+"Inst_Backup/"
'_SCF_+Inst/CTL3D32 が_SCF_+Inst_Backup/CTL3D32 よりも新しい場合，コピーする
Copy (InDir+"CTL3D32" ,OutDir+"CTL3D32" ,VersionUp)
'_SCF_+Inst/SCRIPT.TXT.Z を展開しながら_SCF_+Inst_Backup/SCRIPT.TXT に上書きコピーする
Copy (InDir+"SCRIPT.TXT.Z" ,OutDir+" SCRIPT.TXT" ,Overwrite)
'_SCF_+Inst/SCR*.*を_SCF_+Inst_Backup/に上書きコピーする
Copy (InDir+"SCR*.*" ,OutDir)
```

6.6 メッセージ出力コマンド

メッセージ出力コマンドの詳細を説明します。

Message（ファイルまたはコンソール画面にテキストを出力する）

機能

ファイル，またはコンソール画面にメッセージテキストを出力します。
また，コンソール画面に表示しているメッセージテキストを消去します。

形式

Message (Target , 〔OutputName〕 , 〔Text〕 , 〔LineLength〕 , 〔MaxLines〕)

指定項目

Target

テキストの出力先を次の値で指定します。

値	意味
Target_File	ファイルに出力します。
Target_DispOn	コンソール画面に出力します。
Target_DispClear	コンソール画面に表示しているメッセージテキストを消去します。
Target_DispOff	
Target_SPAFile	実行中スクリプトの解析トレースファイルに出力します。
Target_SPXFile	実行中スクリプトの実行トレースファイルに出力します。

OutputName

Target に Target_File を指定した場合
出力先ファイルのフルパスを文字列，または値を格納した変数名で指定します。

Target に Target_DispOn，Target_DispClear，Target_DispOff，Target_SPAFile，Target_SPXFile を指定した場合
この値は無効になります。

Text

Target に Target_File，Target_DispOn，Target_SPAFile，Target_SPXFile を指定した場合
出力するメッセージテキストを文字列，または値を格納した変数名で指定します。
Target に Target_DispOn 以外を指定した場合は，メッセージテキストに含まれる文字列，"¥r"，"¥n"，"¥t"，"¥¥"は，それぞれ対応するコントロールコードとして処理されます。コントロールコードとして処理しない場合については，「4.1.6 コーディング規則」を参照してください。

Target に Target_DispOn，Target_DispClear，Target_DispOff を指定した場合

この値は無効になります。

LineLength

Target に Target_File を指定した場合
出力するメッセージテキストの 1 行の長さを数値，または値を格納した変数名で指定します。メッセージテキストに含まれる"¥r"，"¥n"を除いた値を指定します。
この値は *OutputName* で指定したファイル名に対して，初めて Message コマンドを実行する場合に指定します。Message コマンドを 2 回目以降に実行する場合に *LineLength* の値が変更されたとき，1 行の長さを変更して *OutputName* で指定したファイルを新しく作成します。
この値は省略できます。省略した場合，150 を仮定します。

Target に Target_DispOn，Target_DispClear，Target_DispOff を指定した場合
この値は無効になります。

Target に Target_SPAFile，Target_SPXFile を指定した場合
この値は無効になり，実行環境ファイルで定義されたトレース情報の最大列数を仮定します。

MaxLines

Target に Target_File を指定した場合
出力するメッセージテキストの最大行数を数値，または値を格納した変数名で指定します。
この値は *LineLength* と同様に，*OutputName* で指定したファイル名に対して，初めて Message コマンドを実行する場合に指定します。Message コマンドを 2 回目以降に実行する場合に MaxLines の値が変更されたとき，最大行数を変更して OutputName で指定したファイルを新しく作成します。
この値は省略できます。省略した場合，1,024 を仮定します。

Target に Target_DispOn，Target_DispClear，Target_DispOff を指定した場合
この値は無効になります。

Target に Target_SPAFile，Target_SPXFile を指定した場合
この値は無効になり，実行環境ファイルで定義されたトレース情報の最大行数を仮定します。

説明

指定したファイル，またはコンソール画面に，指定したメッセージテキストを出力します。また，出力したコンソール画面の内容を消去します。
コマンドが正常に実行された場合は真（True）を，エラーが発生した場合は偽（False）を，コマンドの実行結果として返します。

注意事項

複数起動の指定をしたスクリプト中に，Target_SPAFile，または Target_SPXFile を指定した Message コマンドがあり，そのスクリプトファイルを同時に複数実行した場合，同一の解析トレースファイル，または実行トレースファイルにメッセージを出力します。このため，ファイルに排他制御がかかり，実行性能が低下することがありますので注意が必要です。また，Target_File で出力するファイル名が固定の場合も同様です。

Target_File で作成したファイルを他のアプリケーションでテキスト出力しないでください。その後

の Message コマンドの出力結果が保証されません。

Message コマンドで Target_File を指定して，ユニークなファイル名称を持ったユーザトレースファイルを大量に作成する場合，トレース管理ファイルの増大によって，スクリプトの実行性能に影響を与える場合があります。また，以下の例のようにスクリプト実行が，メモリ不足で異常終了したり，コマンド実行がメモリ不足でエラーになる場合があります。

スクリプト実行が終了コード「20」で異常終了する。

Copy コマンドで「メモリ不足が発生しました」となる。

このような場合は，Message コマンドから TextOpen/TextWrite/TextClose コマンドによってトレースファイルを作成してください。

例

```
' スクリプト実行の履歴を"Loging.TXT"に書き込む。
Message ( Target_File ,_BIN_+"Loging.TXT" ,"実行開始" ,30 ,100 )
   :
Message ( Target_File ,_BIN_+"Loging.TXT" ,"実行終了" ,30 ,100 )

' スクリプト実行の履歴状況をコンソールに表示する。
Message ( Target_DispOn ,"実行状況" ,"開始しました。")
Message ( Target_DispOn ,"実行状況" ,"終了しました。")
Sleep ( 3000 )
Message ( Target_DispOff ,"実行状況" )
```

InputLine（コマンドライン上からテキストを入力する）

機能

コマンドラインにメッセージを出力して，コマンドライン上からテキスト入力できるようにし，入力された値を変数へ格納します。

形式

InputLine（[Text], InputBuff[,DefaultStr]）

指定項目

Text

メッセージとして表示する文字列を文字例，または値を格納した変数名で指定します。この値は省略できます。
実行時には指定したメッセージの先頭に"KBSC4003-I"が付けられます。この値を指定した場合は"KBSC4003-I"だけ表示します。

InputBuff

入力された文字列を格納する変数名を指定します。何も入力されなかった場合は，*DefaultStr* で指定した値を変数へ格納します。*DefaultStr* が指定されていなければ長さ 0 の文字列（""）を変数へ格納します。

DefaultStr

コマンドライン上で文字列が何も入力されなかった場合に格納する文字列，または値を格納した変数名で指定します。この値は省略できます。

説明

指定されたパラメタ Text でメッセージを表示し入力待ちになります。Enter キーが押されるとコマンドラインの内容を InputBuff で指定した変数へ返します。何も入力しないで Enter キーが押された場合は，DefaultStr で指定した値を変数へ格納します。DefaultStr が指定されていなければ長さ 0 の文字列（""）を変数に格納します。

コマンドが正常に実行された場合は真（True）を，エラーが発生した場合は偽（False）をコマンドの実行結果として返します。

Text，および DefaultStr に指定できる文字列の長さは半角文字で 1,024 文字以内です。それを

超えるデータは無視されます。また，InputBuff に格納できる文字列の長さは半角文字で 80 文字以内です。それ以上文字を入力した場合の動作は保障しません。

InputBuff を省略，または文字列を指定した場合は解析エラーとなります。

この機能は標準入出力ファイルを使用します。SetStandardFile，または起動時に標準入力ファイルを指定した場合は，ファイルの先頭 1 行を読み込み，InputBuff で指定した変数に格納されます。再度 InputLine コマンドが実行された場合の読み込み位置は次行になります。

例

```
'コマンドラインに"backup を実行しますか？（する＝"y"）"を表示し，入力待ちとなる。
'"y"を入力（または省略）し Enter キーを押すと"backup"を実行する。
'"y"以外の文字列を入力した場合は"backup"を実行しない。
Dim cmdans
InputLine("backup を実行しますか？（する＝"y"）" , cmdans , "y")
If(cmdans="y")Then
  Exec("backup" , True)
End
```

6.7 演算処理コマンド

演算処理コマンドの詳細を説明します。

+演算子（加算）（二つの式の和を求める）

機能

二つの数値の和を求めます。

形式

Result = expression1 + expression2

指定項目

Result

二つの数値の和を受け取る変数名を指定します。

Expression1

任意の式を指定します。

Expression2

任意の式を指定します。

説明

式 *Expression1* と *Expression2* を加算し，その和を演算結果 *Result* として返します。式の形式と + 演算子で行われる演算を次に示します。

式の形式	行われる演算
両方の式が文字列	文字列連結
両方の式が数値	加算
両方の式が数値だけから成る文字列	加算
一方の式が数値，他方が文字列	文字列連結
一方の式が文字列，他方が数値だけから成る文字列	文字列連結
一方の式が数値，他方が数値だけから成る文字列	加算

両方の式が Empty 値のとき，演算結果 Result は数値 0 になります。ただし，一方の式だけが Empty 値のときは，他方の式がそのまま演算結果 Result として返されます。

例

```
' 変数 result1 には"ABCDEF"が格納される。
result1 = "ABC" + "DEF"

' 変数 result2 には 12 が格納される。
result2 = 7 + 5

' 変数 result3 には"10min."が格納される。
result3 = 10+ "mim."
```

+=演算子（加算）（変数と式の和を変数に代入する）

機能

変数の値と式の値を加算し，その和を変数に代入します。

形式

Result += Expression

指定項目

Result

結果を受け取る変数名を指定します。

Expression

任意の式を指定します。

説明

変数 *Result* の値と式 *Expression* の値を加算し，その和を演算結果として変数 *Result* に代入します。式の形式と+=演算子で行われる演算を次に示します。

式の形式	行われる演算
両方の値が文字列	文字列連結
両方の値が数値	加算
両方の値が数値だけから成る文字列	加算
一方の値が数値，他方が文字列	文字列連結
一方の値が文字列，他方が数値だけから成る文字列	文字列連結
一方の値が数値，他方が数値だけから成る文字列	加算

両方の値が Empty 値のときは，変数 *Result* には数値 0 が代入されます。変数 *Result* が Empty 値，または未定義のときは，式 *Expression* の値がそのまま変数 *Result* に代入されます。式 *Expression* の値が Empty 値のときは，変数 *Result* の値は変わりません。

例

```
' 変数 result1 には"ABCDEF"が格納される。
result1 = "ABC"
result1 += "DEF"

' 変数 result2 には 11 が格納される。
```

```
result2 = 10
result2 += 1

' 変数 result3 には 1 が格納される。
Dim result3
result3 += 1
```

-演算子（減算・マイナス符号）（二つの数値の差を求める）

機能

二つの数値の差を求めます。または，数式の符号を反転した値を指定します。

形式 1

```
Result = Number1 - Number2
```

形式 2

```
-Number
```

指定項目

Result

二つの数値の差を受け取る変数名を指定します。

Number

任意の数式を指定します。

Number1

任意の数式を指定します。

Number2

任意の数式を指定します。

説明

形式 1 では，数式 Number1 から Number2 を減算し，その差を演算結果 Result として返します。

形式 2 では，- 演算子は単項マイナス符号演算子として，式の符号を反転した値を指定するた

めに使用します。

Empty 値を持つ式は，0 として扱われます。

例

```
' 変数 result1 には 2 が格納される。
result1 = 7 - 5
```

-=演算子（減算）（変数と式の差を変数に代入する）

機能

変数の値から式の値を減算し，その差を変数に代入します。

形式

```
Result -= Number
```

指定項目

Result

結果を受け取る変数名を指定します。

Number

任意の数式を指定します。

説明

変数 Result の値から数式 Number の値を減算し，その差を演算結果として変数 Result に代入します。

両方の値が Empty 値のときは，変数 Result には数値 0 が代入されます。変数 Result が Empty 値，または未定義のときは，数式の符号を反転した値-Number が変数 Result に代入されます。数式 Number の値が Empty 値のときは，変数 Result の値は変わりません。

例

```
' 変数 result1 には-1 が格納される。
Dim result1
result1-= 1
```

' 変数 result2 には 9 が格納される。

result2 = 10

result2 -= 1

Mod 演算子（剰余演算）（二つの数値の剰余を求める）

機能

二つの数値の除算を行い，その剰余を返します。

形式

Result = Number1 Mod Number2

指定項目

Result

二つの数値の剰余を受け取る変数名を指定します。

Number1

任意の数式を指定します。

Number2

任意の数式を指定します。

説明

数式 Number1 を Number2 で除算し，その余りを演算結果 Result として返します。

Empty 値を持つ式は，0 として扱われます。

例

' 変数 result1 には 5 が格納される。

Mod=演算子（剰余演算）（変数と式の剰余を変数に代入する）

機能

変数の値を式の値で除算し，その剰余を変数に代入します。

形式

Result Mod= Number

指定項目

Result

結果を受け取る変数名を指定します。

Number

任意の数式を指定します。

説明

変数 Result の値を数式 Number の値で除算し，その余りを演算結果として変数 Result に代入します。

Empty 値は，0 として扱われます。変数 Result が Empty 値，または未定義のときは，0 が変数 Result に代入されます。数式 Number の値が Empty 値のときは，0 で除算を行うことになるため実行エラーになります。

例

```
' 変数 result1 には 19 が格納される。
result1 = 19
result1 Mod = 7
result1 = 19 Mod 7
```

*演算子（乗算）（二つの数値の積を求める）

機能

二つの数値の積を求めます。

形式

Result = Number1 * Number2

指定項目

Result

二つの数値の積を受け取る変数名を指定します。

Number1

任意の数式を指定します。

Number2

任意の数式を指定します。

説明

数式 Number1 に Number2 を乗算し，その積を演算結果 Result として返します。

Empty 値を持つ式は，0 として扱われます。

例

' 変数 result1 には 35 が格納される。

*=演算子（乗算）（変数と式の積を変数に代入する）

機能

変数の値に式の値を乗算し，その積を変数に代入します。

形式

Result *= Number

指定項目

Result

結果を受け取る変数名を指定します。

Number

任意の数式を指定します。

説明

変数 Result の値に数式 Number の値を乗算し，その積を演算結果として変数 Result に代入します。

Empty 値は，0 として扱われます。変数 Result が Empty 値，または未定義の場合，または数式 Number の値が Empty 値の場合は，0 が変数 Result に代入されます。

例

```
' 変数 result1 には 35 が格納される。
result1 = 7
result1 *= 5
result1 = 7 * 5
```

/演算子（除算）（二つの数値の商を求める）

機能

二つの数値の商を計算し，結果を整数で返します。

形式

Result = Number1 / Number2

指定項目

Result

二つの数値の商を受け取る変数名を指定します。

Number1

任意の数式を指定します。

Number2

任意の数式を指定します。

説明

数式 *Number1* を *Number2* で除算し，その商を演算結果 *Result* として整数で返します。

Empty 値を持つ式は，0 として扱われます。

例

' 変数 result1 と変数 result2 には 2 が格納される。

result1 = 14 / 7

result2 = 19 / 7

/=演算子（除算）（変数と式の商を変数に代入する）

機能

変数の値を式の値で除算し，その商を整数で変数に代入します。

形式

Result /= Number

指定項目

Result

結果を受け取る変数名を指定します。

Number

任意の数式を指定します。

説明

変数 Result の値を数式 Number の値で除算し，その商を演算結果として整数で変数 Result に代入します。

Empty 値は，0 として扱われます。変数 Result が Empty 値，または未定義のときは，0 が変数 Result に代入されます。数式 Number の値が Empty 値のときは，0 で除算を行うことになるため実行エラーになります。

例

```
' 変数 result1 と変数 result2 には 2 が格納される。
Dim result1, result2
result1 = 14
result2 = 19
result1 /= 7
result2 /= 7
```

¥演算子（整数除算）（二つの数値の商を求める）

機能

二つの数値の商を計算し，結果を整数で返します。

形式

Result = Number1 ¥ Number2

指定項目

Result

二つの数値の商を受け取る変数名を指定します。

Number1

任意の数式を指定します。

Number2

任意の数式を指定します。

説明

数式 Number1 を Number2 で除算し，その商を演算結果 Result として整数で返します。

Empty 値を持つ式は，0 として扱われます。

例

```
' 変数 result1 と変数 result2 には 2 が格納される。
result1 = 14 ¥ 7
result2 = 19 ¥ 7
```

¥=演算子（整数除算）（変数と式の商を変数に代入する）

機能

変数の値を式の値で除算し，その商を整数で変数に代入します。

形式

Result ¥= Number

指定項目

Result

結果を受け取る変数名を指定します。

Number

任意の数式を指定します。

説明

変数 Result の値を数式 Number の値で除算し，その商を演算結果として整数で変数 Result に代入します。

Empty 値は，0 として扱われます。変数 Result が Empty 値，または未定義のときは，0 が変数 Result に代入されます。数式 Number の値が Empty 値のときは，0 で除算を行うことになるため実行エラーになります。

例

```
' 変数 result1 と変数 result2 には 2 が格納される。
result1 = 14
result2 = 19
result1 ¥= 7
result2 ¥= 7
```

^演算子（べき乗）（二つの数値のべき乗を求める）

機能

二つの数値のべき乗を計算し，結果を整数で返します。

形式

Result = Number1 ^ Number2

指定項目

Result

二つの数値のべき乗を受け取る変数名を指定します。

Number1

任意の数式を指定します。

Number2

任意の数式を指定します。

説明

数式 Number1 と Number2 のべき乗を計算し，その演算結果を Result に整数で返します。

Empty 値を持つ式は，0 として扱われます。

例

```
' 変数 result1 には 8 が格納される。
result1 = 2 ^ 3
```

^=演算子（べき乗）（変数と式のべき乗を変数に代入する）

機能

変数の値と式の値のべき乗を計算し，結果を整数で変数に代入します。

形式

Result ^= Number

指定項目

Result

結果を受け取る変数名を指定します。

Number

任意の数式を指定します。

説明

変数 Result の値と数式 Number の値のべき乗を計算し，その演算結果を整数で変数 Result に代入します。

Empty 値は，0 として扱われます。変数 Result が Empty 値，または未定義のときは，0 が変数 Result に代入されます。数式 Number の値が Empty 値のときは，0 でべき乗を計算することになるため常に 1 になります。

例

```
' 変数 result1 には 8 が格納される。
result1 = 2
result1 ^= 3
```

比較演算子（=, <>, <, <=, >, >=）（二つの式を比較する）

機能

二つの式を比較します。

形式

Result = Expression1 comparisonoperator Expression2

指定項目

Result

結果を受け取る変数名を指定します。

Expression1

任意の式を指定します。

Expression2

任意の式を指定します。

comparisonoperator

任意の比較演算子を指定します。

説明

二つの式を比較します。

各比較演算子の演算結果 *Result* が，真（True），偽（False）となる条件を，次に示します。

=（等しい）

真となる条件 *Expression1* = *Expression2*

偽となる条件 *Expression1* <> *Expression2*

<>（等しくない）

真となる条件 *Expression1* <> *Expression2*

偽となる条件 *Expression1* = *Expression2*

<（より小さい）

真となる条件 *Expression1* < *Expression2*

偽となる条件 *Expression1* >= *Expression2*

<=（以下）

真となる条件 *Expression1* <= *Expression2*

偽となる条件 *Expression1* > *Expression2*

>（より大きい）

真となる条件 *Expression1* > *Expression2*

偽となる条件 *Expression1* <= *Expression2*

>=（以上）

真となる条件 *Expression1* >= *Expression2*

偽となる条件 *Expression1* < *Expression2*

保持しているデータ型の条件と行われる演算との対応を次に示します。

条件	行われる演算
両方の式が数値データ型	数値比較
両方の式が文字列型	文字列比較
一方の式が数値データ型，他方の式が文字列型	文字列比較
一方の式が Empty 値，他方の式が数値データ型	Empty 値を 0 とみなして数値比較をします。ただし，等しいかどうかを判定する場合は，0 とみなしません。
一方の式が Empty 値，他方の式が文字列型	Empty 値を長さ 0 の文字列（""）とみなして文字列比較をします。
両方の式が Empty 値	二つの式は等しいという結果になります

論理積(And)(二つの式の論理積を求める)

機能

二つの式の論理積を求めます。

形式

Result = Expression1 And Expression2

指定項目

Result

結果を受け取る変数名を指定します。

Expression1

真(True), または偽(False)を評価する任意の式を指定します。

Expression2

真(True), または偽(False)を評価する任意の式を指定します。

説明

二つの式の論理積を求めます。

演算結果 *Result* の値は次のようになります。

Expression1	Expression2	Result
True	True	True
True	False	False
False	True	False
False	False	False
文字列	True	True
文字列	False	False
文字列	文字列	True
Empty 値	True	False
Empty 値	False	False
Empty 値	Empty 値	False

例

' 変数 Number1 の値が 1 以上 10 以下の場合,メッセージを表示する。

Number1 = Day ( )

```
If 1 <= Number1 And Number1 <= 10 Then
        Message( Target_DispOn, "実行結果", "OK" )
End If
```

論理和(Or)(二つの式の論理和を求める)

機能

二つの式の論理和を求めます。

形式

Result = Expression1 Or Expression2

指定項目

Result

結果を受け取る変数名を指定します。

Expression1

真(True), または偽(False)を評価する任意の式を指定します。

Expression2

真(True), または偽(False)を評価する任意の式を指定します。

説明

二つの式の論理和を求めます。

演算結果 *Result* の値は次のようになります。

Expression1	Expression2	Result
True	True	True
True	False	True
False	True	True
False	False	False
文字列	True	True
文字列	False	True
文字列	文字列	True
Empty 値	True	True
Empty 値	False	False
Empty 値	Empty 値	False

例

```
' 変数 Number1 の値が 7 以下，または 20 以上の場合,メッセージを表示する。
Number1 = Hour ( )
If   Number1 <= 7   Or   20 <= Number1   Then
      Message( Target_DispOn, "実行結果", "OK" )
End If
```

論理否定(Not)(式の論理否定を求める)

機能

式の論理否定を求めます。

形式

```
Result = Not Expression
```

指定項目

Result

結果を受け取る変数名を指定します。

Expression

真(True)，または偽(False)を評価する任意の式を指定します。

説明

式の論理否定を求めます。

演算結果 *Result* の値は次のようになります。

Expression	Result
True	False
False	True
文字列	False
Empty 値	True

例

```
' 変数 Number1 の値が 10 より小さい場合，メッセージを表示する。
Number1 = Day ( )
If   Not 10 < Number1 Then
      Message( Target_DispOn, "実行結果", "OK" )
```

```
End If
```

6.8 チェック処理コマンド

チェック処理コマンドの詳細を説明します。

IsEmpty（変数が Empty 値かどうかを調べる）

機能

変数が Empty 値かどうかを調べて，結果を真（True）か偽（False）で返します。

形式

IsEmpty (VarName)

指定項目

VarName

調べる変数名を指定します。

説明

指定した変数が Empty 値かどうかを調べて，結果をコマンドの実行結果として返します。Empty 値の場合は真（True）を，それ以外の場合は偽（False）を返します。

また，指定した変数が定義されていない場合は，常に偽（False）を返します。

例

```
' _ALLRIGHT_予約変数が Empty 値かどうかを調べる。
If　IsEmpty ( _ALLRIGHT_ )　Then
　Message( Target_DispOn, "実行結果","スーパーユーザの ID でログオンし直して_
ください。" )
End If
```

IsDefine または IsDef（変数が定義されているかどうかを調べる）

機能

変数が定義されているかどうかを調べて，結果を真（True）か偽（False）で返します。

形式

IsDefine (VarName)

IsDef (VarName)

指定項目

VarName

調べる変数名を指定します。

説明

指定した変数が定義されているかどうかを調べて，結果をコマンドの実行結果として返します。
定義されている場合は真（True）を，それ以外の場合は偽（False）を返します。

例

```
' 位置変数%1 が指定したかどうかで処理を分岐する。
If   IsDefine ( %1 )   Then
   Exec ( "CallProc01.SPT" ,True ,%1 )
Else
   Exec ( "CallProc02.SPT" ,True )
End If
```

IsNumeric（数値として評価できるかを調べる）

機能

値が数値として評価できるかどうかを調べて，結果を真（True）か偽（False）で返します。

形式

IsNumeric (CheckValue)

指定項目

CheckValue

調べる値，または値を格納した変数名を指定します。

説明

指定した値が数値として評価できるかどうかを調べて，結果をコマンドの実行結果として返します。数値として評価できる場合は真（True）を，それ以外の場合は偽（False）を返します。

IsEmptyDir（ディレクトリの中身が空かどうかを調べる）

機能

ディレクトリの中身が空かどうかを調べて，結果を真（True）か偽（False）で返します。

形式

IsEmptyDir (DirName)

指定項目

DirName

チェックするディレクトリ名を文字列，または値を格納した変数名で指定します。

説明

指定したディレクトリの中にファイル，またはディレクトリが存在するかどうかをチェックして，結果をコマンドの実行結果として返します。中身が空の場合は真（True）を，それ以外の場合は偽（False）を返します。

また，指定したディレクトリが存在しない場合は，常に偽（False）を返します。

例

```
' "BKUP"ディレクトリの中身が空かどうかを調べる。
outDir = _SCF_+"BKUP¥"
If   IsExistDir ( OutDir ) = True   Then
   If   IsEmptyDir ( OutDir ) = True   Then
      Message( Target_DispOn, "実行結果", "Empty !" )
   Else
      Message( Target_DispOn, "実行結果", "Not Empty !" )
   End
Else
   Message( Target_DispOn, "実行結果", "Not Exist !" )
End
```

IsExistDir（ディレクトリが存在するかどうかを調べる）

機能

ディレクトリが存在するかどうかを調べて，結果を真（True）か偽（False）で返します。

形式

IsExistDir (DirName)

指定項目

DirName

チェックするディレクトリ名を文字列，または値を格納した変数名で指定します。

説明

指定したディレクトリが存在するかどうかをチェックし，結果をコマンドの実行結果として返します。存在する場合は真（True）を，それ以外の場合は偽（False）を返します。指定したディレクトリに読み取り権限がない場合は偽（False）を返します。

例

```
' カレントディレクトリ下に"SCRIPT"ディレクトリが存在しない場合は作成する。
path1 = _BIN_+"SCRIPT"
If   IsExistDir ( path1 ) = FALSE   Then
   MakeDir ( path1 )
End
```

IsExistFile（ファイルが存在するかどうかを調べる）

機能

ファイルが存在するかどうかをチェックして，結果を真（True）か偽（False）で返します。

形式

IsExistFile (FilePath)

指定項目

FilePath

チェックするファイルのフルパスを文字列，または値を格納した変数名で指定します。

説明

指定したファイルが存在するかどうかをチェックして，結果をコマンドの実行結果として返します。
存在する場合は真（True）を，それ以外の場合は偽（False）を返します。

例

```
Dim    IsExist
IsExist = IsExistFile ( _BIN_+"SCRIPT/Loging.TXT")

If    IsExist = True    Then
      Copy ( _BIN_+"SCRIPT/Loging.TXT",_TEMP_+"ScpLog.TXT" )
      If    _COPY_RTN_ <> Skip    Then
      DeleteFile ( _BIN_+"SCRIPT/Loging.TXT" )
      End
End
```

IsWriteableDir（ディレクトリが書き込みできるかどうかを調べる）

機能

ディレクトリが書き込みできるかどうかをチェックして，結果を真（True）か偽（False）で返します。

形式

IsWriteableDir (DirName)

指定項目

DirName

チェックするディレクトリ名を文字列，または値を格納した変数名で指定します。

説明

指定したディレクトリに書き込みできるかどうかをチェックして，結果をコマンドの実行結果として返します。書き込みできる場合は真（True）を，それ以外の場合は偽（False）を返します。

また，指定したディレクトリが存在しない場合は，常に偽（False）を返します。

例

```
' "bkup"ディレクトリに書き込みできるかどうかを調べる。
outDir = _SCF_+"bkup/"
If   IsWriteableDir ( OutDir ) = True   Then
   writeFlag ="OK"
Else
   writeFlag = "NG"
End
```

IsNew（ファイル日付の新旧を比較する）

機能

二つのファイルのファイル日付の新旧を比較し，結果を真（True）か偽（False）で返します。

形式

IsNew (PathName1 , PathName2 〔, Option〕)

指定項目

PathName1

比較する一方のファイルのフルパスを文字列，または値を格納した変数名で指定します。

PathName2

比較するもう一方のファイルのフルパスを文字列，または値を格納した変数名で指定します。

Option

このオプションには次の値を指定します。省略した場合は，FileTime を仮定します。

値	意味
FileTime	ファイル日付だけで比較します。

説明

二つのファイルの日付の新旧を比較し，PathName1 で指定したファイルが PathName2 で指定したファイルよりも新しい場合は真（True）を，古い，または等しい場合は偽（False）を，コマンドの実行結果として返します。
エラーが発生した場合は，長さ 0 の文字列（""）を返します。

例

```
'_SCF_+/TEST.SPT ファイルと/WORK/TEST.SPT ファイルの更新日付をチェックして，
'_SCF_+/TEST.SPT ファイルが新しい場合は，/WORK/TEST.SPT ファイルをコピーする。
Dim file1 ,file2
file1 = _SCF_+"TEST.SPT"
file2 = "/WORK/TEST.SPT"
If IsNew(file1 ,file2) = False Then
' 古いファイルに戻す
Copy(file2 ,file1)
```

```
End
```

CheckDirName（ディレクトリ名の末尾がスラント（ / ）かどうかを調べる）

機能

ディレクトリ名の末尾がスラント（ / ）であるかどうかを調べます。

形式

CheckDirName (DirNameBuff 〔, Option〕)

指定項目

DirNameBuff

チェックするディレクトリ名を格納している変数名を指定します。

Option

このオプションには次の値を指定します。

値	意味
Remove	末尾のスラント（ / ）を取り去ります。

説明

オプションが省略された場合，指定した変数に格納されているディレクトリパスの末尾の文字を調べ，スラント（ / ）以外の文字の場合は，末尾にスラント（ / ）が付きます。オプションにRemove が指定されている場合は，末尾のスラント（ / ）を取り去ります。

コマンドが正常に実行された場合は真（True）を，エラーが発生した場合は偽（False）を，コマンドの実行結果として返します。指定した変数に格納されているディレクトリのパスがスラント（/）1文字の場合，*Option* の指定にかかわらず実行結果は，スラント（/）1文字になります。

例

```
' ディレクトリパスが( / )で終わることを保証してからフルパス名を作成
If CheckDirName ( OutDir ) = False Then
    Exit
End
If CheckDirName ( InDir ) = False Then
    Exit
End
Copy ( InDir+"file.dat" ,OutDir+"file.dat" ,Overwrite )
```

6.9 外部プログラム呼び出しコマンド

外部プログラム呼び出しコマンドの詳細を説明します。

Exec（実行ファイルを呼び出す）

機能

複数パラメタ指定で実行ファイル（実行可能ファイル，JP1/Script ファイル，シェルスクリプトファイル）を呼び出します。

形式

Exec (FileName , Flag 〔, Param1 , Param2 , …Param31 〕)

指定項目

FileName

呼び出す実行ファイル名を文字列，または値を格納した変数名で指定します。

実行ファイルに指定できるファイルの種類は次のとおりです。

実行可能ファイル

JP1/Script ファイル（.SPT）

シェルスクリプトファイル

Flag

真（True）を指定します。*FileName* で指定した実行ファイルの終了を待ちます。

Param1～31

FileName で指定したファイルの実行に必要なパラメタがある場合，パラメタを文字列，数値，または値を格納した変数名で指定します。

複数のパラメタを指定できます。

複数のパラメタを一つずつ格納した 1 次元の配列変数を，変数名でも指定できます。

パラメタの記述規則の詳細は，「4.2.3 コマンドラインの記述規則」を参照してください。

説明

指定した実行ファイルを実行します。呼び出したアプリケーションが終了するまで待機します。

アプリケーションが正常終了した場合は真（True）を，それ以外の場合は偽（False）を，コマンドの実行結果として返します。

コマンドの実行結果が真（True）の場合は，実行ファイルの終了コードを_EXEC_RTN_予約変数

に符号付きの数値で格納します。コマンドの実行結果が偽（False）の場合は，_EXEC_RTN_予約変数には何も格納されません。

補足

Exec コマンドは，呼び出す実行ファイルの種類が異なると，同等のエラーであってもコマンドの実行結果が異なる場合があります。例えば，存在しない実行可能ファイルを呼び出した場合は偽（False）を，存在しない JP1/Script ファイルを呼び出した場合は真（True）をコマンドの実行結果として返します。

Exec コマンドが正常終了したかどうかを判定するためには，コマンドの実行結果だけでなく，_EXEC_RTN_予約変数に格納された実行ファイルの終了コードの判定も必要です。例 1 を参照してください。

Exec コマンドでシェルスクリプトファイルを呼び出す場合は，スクリプトファイルの拡張子を（.sh）に変更するか，呼び出す実行ファイル名にシェルプログラムを指定しパラメタ1にシェルスクリプトファイル名を指定してください。例 2，および例 3 を参照してください。指定したスクリプトファイルが存在しない場合は，_EXEC_RTN_予約変数には 127 が設定されます。

Exec コマンドで実行可能ファイルを呼び出した場合，基本的には実行可能ファイルが設定した終了コードが返ります。しかし，実行可能ファイルが途中で異常終了などした場合，次の終了コードが返ることがあります。

値	意味
4	不正な命令（illegal instruction（not reset when caught））
6	プロセス中断（about process）
9	強制終了（kill (cannot be caught or ignored)）
11	セグメンテーション違反（segmentation violation）
12	システムコールに対する無効な引数（不正システムコール）
15	ソフトウェア終了（kill による強制終了）（software termination signal）

例 1

```
' スクリプトファイル"ABC.SPT"を呼び出す。
rtn = Exec ( _SCF_+"ABC.SPT", True )
rtnCode = _RTN_

' 実行結果を判定する。
If   rtn = True   Then
   If   _EXEC_RTN_ = 0   Then
          Message( Target_DispOn, "実行結果", "正常終了しました。")
   Else
```

```
        Message( Target_DispOn, "実行結果", "異常終了しました。終了コード＝" +
_EXEC_RTN_)
        Exit   (_EXEC_RTN_)
  End
Else
    Message( Target_DispOn, "実行結果", "異常終了しました。エラーコード＝" +
rtnCode)
    Exit   (rtnCode)
End
```

例 2

```
' シェルファイル"ABC.sh"を呼び出す(シェルファイルの拡張子を".sh"にして実行する方法)。
Rtn1 = Exec(_BIN_+"ABC.sh", True)
:
```

例 3

```
' シェルファイル"ABC"を呼び出す(シェルプログラムを起動して"ABC"を引き渡す方法)。
Rtn1 = Exec("/usr/bin/sh", True, "ABC")
:
```

6.10 コメントコマンド

コメントコマンドの詳細を説明します。

Rem または '(プログラム内にコメントを記述する)

機能

プログラム内にコメントを記述する場合に指定します。

形式

Rem Comment

' Comment

指定項目

Comment

プログラムをわかりやすくするための説明文(コメント)を記述します。Rem と *Comment* の間には,半角文字で 1 文字以上のスペースを入力する必要があります。

説明

Rem の代わりに引用符(')を使えます。同じ行のほかのステートメント, またはコマンドのあとに引用符(')や Rem を記述できます。

例

```
Rem　ファイルにメッセージを出力する。
Message ( Target_File ,_SCF_+"履歴.TXT" ,"実行開始" )

' コンソール画面にメッセージを出力する。
Message ( Target_DispOn ,"履歴" ,"実行開始" )
```

6.11 その他のコマンド

その他のコマンドの詳細を説明します。

Sleep（スクリプトの実行を中断する）

機能

指定した時間中，スクリプトの実行を中断します。

形式

Sleep (Time)

指定項目

Time

停止する時間を数値，または値を格納した変数名で指定します。
時間はミリ秒単位で指定します。

説明

指定した時間中，スクリプトの実行が中断されます。

例

```
' デバッグ時は 3 秒間スリープする。
If　Debug = True　Then
　Sleep ( 3000 )
End
```

Beep（スピーカからビープ音を鳴らす）

機能

指定した時間中，スピーカからビープ音を鳴らします。

形式

Beep

説明

スピーカからビープ音を単発で鳴らします。ビープ音は非同期で鳴るため，連続して実行すると以下のような現象が起きる場合があります。

スクリプトの実行が終了してもビープ音が鳴ることがあります。

指定した回数のビープ音が鳴らないことがあります。

このような場合は，Sleep コマンドで間隔を空けて，コマンドを実行してください。

例

```
' エラー時はビープ音を鳴らす。
If_RTN_ <> 0 Then
    Beep
```

Exit（スクリプトを終了する）

機能

スクリプトの実行を終了します。

形式

Exit (〔Code〕)

指定項目

Code

終了コードを数値，または値を格納した変数名で指定します。指定できる終了コードの数値は 0～255 です。

この値を省略した場合は，0 が仮定されます。

説明

スクリプトの実行を終了します。

なお，JP1/Script の終了コードの詳細は，「4.1.7 終了コード」を参照してください。

例

```
' 終了コード 0 でスクリプトを終了する
Message( Target_DispOn, "実行結果", "スクリプトの実行を終了します。" )
Exit ( 0 )
```

GetErrorMessage（エラーメッセージを取得する）

機能

指定したエラー詳細コードのエラーメッセージを取得します。

形式

GetErrorMessage (〔Code〕)

指定項目

Code

エラー詳細コードを数値，または値を格納した変数名で指定します。

この値は省略できます。省略した場合，現在の_RTN_予約変数に格納されている値を仮定します。

説明

指定したエラー詳細コードのエラーメッセージを取得し，メッセージを実行結果として返します。

また，エラーメッセージの文字数が半角文字で 1,024 文字を超える場合，1,024 文字で切り捨てられます。

例

```
If  Exec ( "ABC" ,True ,_BIN_+"Loging.TXT" )  Then
  Exit ( _EXEC_RTN_ )
Else
```

```
    Dim    ErrMsg
    ErrMsg = GetErrorMessage ( _RTN_ )
    Message ( Target_File ,_BIN_+"ErrLog.TXT" ,ErrMsg )
    Exit ( 1 )
End
```

7. 障害発生時の対処方法

この章では，JP1/Script で発生する障害の対処方法について説明します。

7.1 スクリプト実行時の障害の対処方法

スクリプト実行時に障害が発生した場合，次のファイルから原因を調査できます。
解析トレースファイル，および実行トレースファイルから原因を調査する
syslog ファイルから原因を調査する
システム情報ファイルから原因を調査する

7.1.1 解析トレースファイルおよび実行トレースファイルから原因を調査する

(1) 解析トレースファイル

解析トレースファイルは，スクリプトエンジンプログラムが，コマンドの解析結果を出力するファイルです。このファイルから障害の原因を調査します。

解析トレースファイルの詳細は，「付録 A.1 トレースファイルの出力形式」を参照してください。

(2) 実行トレースファイル

実行トレースファイルは，スクリプト実行プログラムが，コマンドの実行結果を出力するファイルです。このファイルから障害の原因を調査します。実行トレースファイルの詳細は，「付録 A.1 トレースファイルの出力形式」を参照してください。

7.1.2 syslog ファイルから原因を調査する

スクリプトの実行時，その状況に応じたメッセージを **syslog ファイル**へ出力します。この syslog ファイルから障害の原因を調査します。出力される**ログ ID** と **syslog メッセージ**の意味を，表 7-1 に示します。
また，メッセージにはメッセージ ID（KBSC3xxx-X（xxx:メッセージ番号））を付け，環境変数 LANG の指定にかかわらず，メッセージは英文で出力されます。

表 7-1　ログ ID と syslog メッセージの意味

ログ **ID**	syslog **メッセージの意味**
1	スクリプト実行開始時の情報ログ
2	スクリプト実行終了時の情報ログ
16	複数スクリプト起動時のエラーログ
19	文法エラーによるスクリプト終了時の情報ログ
98	解析トレース，および実行トレース未出力時のエラーログ※
99	スクリプト実行開始時のエラーログ

注※

システム環境ファイル（jp1script.cf）の ErrorEventLog に 1 を指定した場合だけ出力されます。

これらのログ ID の中で，エラーログ以外のログ ID を，syslog ファイルに出力しないことができます。出力抑止を指定する場合は，「4.2.2(4) -SPT:NOSYSLOG（または-spt:nosyslog）」を参照してください。

各ログ ID の説明を次に示します。出力形式中の斜体部分は，ログごとに内容が変わります。なお，各メッセージの先頭には sptxe[*プロセス ID*]：メッセージ ID が付きます。

(1)　ログ ID：1

出力時のレベルは LOG_INFO（通知メッセージ）を設定します。

出力形式

"sptxe[プロセス ID]:メッセージ ID<ログ ID>:Script execution has started.（スクリプトファイル名）"

(2)　ログ ID：2

出力時のレベルは LOG_INFO（通知メッセージ）を設定します。

出力形式

" sptxe[プロセス ID]:メッセージ ID<ログ ID>:Script execution has finished.（スクリプトファイル名）"

(3)　ログ ID：16

出力時のレベルは LOG_ERR（エラー）を設定します。

出力形式

" sptxe[プロセス ID]:メッセージ ID<ログ ID>:The specified script file has already been started.（スクリプトファイル名）"

(4)　ログ ID：19

出力時のレベルは LOG_INFO（通知メッセージ）を設定します。

出力形式

" sptxe[プロセス ID]:メッセージ ID<ログ ID>:Script execution has finished. Syntax error occurred.（スクリプトファイル名）"

(5) ログ ID：98

出力時のレベルは LOG_ERR（エラー）を設定します。

出力形式

"sptxe[プロセス ID]:メッセージ ID<ログ ID>: [ユーザ名]：スクリプト実行時のユーザ名　[スクリプトファイル名]：スクリプトファイル名　[エラー内容]：エラーの行位置　行目;この部分には発生したエラーのエラーメッセージが出力されます。"

(6) ログ ID：99

出力時のレベルは LOG_ERR（エラー）を設定します。

出力形式

"sptxe[プロセス ID]:メッセージ ID<ログ ID>: [ユーザ名]：発生したエラーのエラーメッセージ"

注意

syslog ファイルへの出力に失敗した場合，メッセージは出力されません。

syslog ファイルへの出力例を図 7-1 に示します。

図 7-1　syslog ファイルへの出力例

```
Oct 22 10:22:21 ep8000 sptxe[24716]: KBSC3002-I<1>Script execution has finished.(/home/
kawano/VarCon/U-JE-01-128.SPT)
Oct 22 10:22:35 ep8000 sptxe[25818]: KBSC3001-I<1>Script execution has started.(/home/
kawano/VarCon/U-JE-01-130.SPT)
Oct 22 10:22:35 ep8000 sptxe[25818]: KBSC3002-I<1>Script execution has finished.(/home/
kawano/VarCon/U-JE-01-130.SPT)
```

7.1.3 システム情報ファイルから原因を調査する

システム情報ファイルは，スクリプトを実行したシステムの実行環境情報としてマシン情報などを出力するファイルです。
システム情報の詳細は，「付録 A.2 システム情報ファイルの出力形式」を参照してください。

7.2 バックアップとリカバリー

JP1/Script のファイルや動作環境情報のバックアップとリカバリーについて説明します。
バックアップ対象ファイルを表 7-2 に示します。必要に応じてバックアップし，元の場所へリカバリーします。

表 7-2 バックアップ対象ファイル

項番	ファイル名	ファイル名または拡張子	ファイルの所在
1	スクリプトファイル	.SPT	ユーザ任意のディレクトリに作成されます。 スクリプトファイル以外はスクリプトの定義や実行後に作成されるファイルなので，ファイルが存在しない場合もあります。
2	実行環境ファイル	.SPV	
3	解析トレースファイル	.SPA	
4	実行トレースファイル	.SPX	
5	ユーザトレースファイル	.TXT など	
6	トレース管理ファイル	SPTLOGDB.SPB	/opt/jp1script/data に作成されます。 スクリプトの定義やの実行後に作成されるファイルなので，ファイルが存在しない場合もあります。
7	グローバル変数ファイル	SPTGV.SPG	
8	実行環境構文ファイル	.SPU	ユーザ任意のフォルダに作成されます。実行環境ファイルコンバータを使用しないかぎり，存在しません。
9	システム情報ファイル	SPTENV01 ～ 50.LOG	/opt/jp1script/log/sptenv に作成されます。実行プログラム（sptxe）の実行後に作成されるファイルなので，ファイルが存在しない場合もあります。

8. メッセージ

この章では，JP1/Script で表示されるメッセージについて説明します。

8.1 メッセージの出力形式

メッセージはメッセージ ID と，それに続くメッセージテキストで構成されます。
JP1/Script で表示されるメッセージの形式を示します。

KBSCnnnnn-Z メッセージテキスト

メッセージ ID は，次の内容で構成されています。

K

システム識別子を表します。

BSC

JP1/Script のメッセージであることを表します。

nnnnn

メッセージの通し番号を表します。

z

メッセージの種類を表します。メッセージの種類を次に示します。

E

エラーメッセージを表します。処理は中断されます※。

W

警告メッセージを表します。メッセージ表示後，処理は続行されます。

I

通知メッセージを表します。ユーザに情報を知らせます。

Q

確認メッセージを表します。ユーザに情報を確認させます。

注※

メッセージ ID KBSC9001~9999 で種別が E の場合，処理が続行される場合もあります。

8.2 メッセージの記述形式

メッセージ ID

英語メッセージテキスト

日本語メッセージテキスト

メッセージの説明文

（S）

JP1/Script の処置を示します。

（O）

メッセージが表示されたときに，ユーザがとる処置を示します。

なお，メッセージによっては，メッセージテキストの先頭に次の内容が表示される場合があります。

<n>:

実行時のイベント ID です。Syslog 中のメッセージにだけ付加されます。なお，n はメッセージごとに固定の数字になります。

(nnnn):

システムのエラーコードです。このエラーコードの後ろに，エラーに対応するメッセージ（OS から取得されたもの）が表示されます。

8.3 メッセージの出力先

メッセージ ID　KBSC1001～KBSC1999：解析トレースファイル
メッセージ ID　KBSC2001～KBSC2999：実行トレースファイル
メッセージ ID　KBSC3001～KBSC3999：syslog
メッセージ ID　KBSC4001～KBSC4999：標準出力装置，解析／実行トレースファイル
メッセージ ID　KBSC5001～KBSC5777：標準出力装置
メッセージ ID　KBSC9001～KBSC9999：実行トレースファイル，syslog