Micro Focus Net Express[R]

キャラクタユーザインターフェイス

第 2 版
2001 年 6 月

このマニュアルでは、キャラクタユーザインターフェイスの構築に使用する Net Express ツールを紹介します。ここでは、Adis モジュール、Panels、およびオンラインヘルプについて説明し、文字画面の設定と入力方法を紹介します。また、多くの使用例とサンプルプログラムを紹介しています。

第 1 章 : はじめに

Micro Focus では、文字環境とグラフィカル環境の両方にユーザインターフェイスを作成する多くの方法を提供しています。Net Express ヘルプでは、グラフィカルユーザインターフェイスを作成する方法について詳細に説明しています。

このオンラインマニュアルでは、キャラクタユーザインターフェイスの作成に Net Express で使用可能なツールを紹介します。 説明する内容は、次のとおりです。

- Adis
- Keybcf
- Panels
- 文字画面ルーチン
- ウィンドウ構文
- オンラインヘルプ

これらのツールの機能によって、アプリケーションに高度でわかりやすいユーザインターフェイスを作成できます。 ただし、これらは下位互換性のためのみに提供されています。新しいアプリケーションを作成するときには、これらの機能を使用しないことをお奨めします。

- 拡張 ACCEPT および DISPLAY 構文 - 拡張 ACCEPT/DISPLAY 構文によって、画面位置と画面属性を指定できます。単一フィールドまたは複数フィールドの ACCEPT 操作を行うことができます。複数フィールドの ACCEPT 操作の場合には、FILLER は次のフィールドへスキップする文字数を記述します。DISPLAY 操作では、FILLER は定数間の空白文字数を定義します。 FILLER として定義されている全領域は、ACCEPT または DISPLAY 動作の影響は受けません。
- ANSI ACCEPT/DISPLAY 構文 - ANSI ACCEPT 構文を使用すると、データ項目を入力したり、曜日、日付、時間をデータ項目に受け取ったりできます。 ANSI DISPLAY 構文は定数やデータ項目の内容の出力を実現します。
- Panels は、呼び出しインターフェイスを使用した多くのテキストウィンドウサポートを提供します。COBOL プログラムから任意の数のウィンドウを作成し、操作することができ、同時に 255 個までの表示が可能です。たとえば、スクロール、文字や属性のブロック更新、パネルの移動、表示領域のサイズ変更などの多くのパネル操作機能が提供されています。 Adis モジュール (拡張 ACCEPT/DISPLAY 構文をサポート) からの出力は、呼び出しインターフェイスを使用してパネルへ直接出力できます。
- 画面節は、1 つ以上の画面定義を含むデータ部にある 1 つの節です。画面定義には、フィールド、グループ、テキストと属性を含めることができます。フィールドには、編集した絵文字列を入れることができます。また、NO-ECHO、JUSTIFIED RIGHT、および BLANK WHEN ZERO などの関数を入れることもできます。 画面定義は、手続き部で取り込まれ表示されます。
- 下位レベルの COBOL システムライブラリルーチンの下位レベルインターフェイスは、COBOL システムライブラリルーチンによって提供されます。これらのルーチンを使用すると、COBOL プログラムから下位レベル機能にアクセスできます。
- ウィンドウ構文 - COBOL システムは、端末の画面上に線やボックスを描画したり、実端末上の仮想端末ウィンドウを作成する構文をサポートします。

1.1 キャラクタインターフェイスでの拡張 ACCEPT/DISPLAY 構文 (Adis) の使用方法

ACCEPT/DISPLAY 拡張構文によって、画面位置と画面属性を指定できます。単一フィールドまたは複数フィールドの ACCEPT 操作を行うことができます。複数フィールドの ACCEPT 操作の場合には、FILLER は次のフィールドへスキップする文字数を記述します。DISPLAY 操作では、FILLER は定数間の空白文字数を定義します。 FILLER として定義されている全領域は、ACCEPT または DISPLAY 動作の影響は受けません。

拡張 ACCEPT/DISPLAY と画面節 ACCEPT/ DISPLAY 操作では、Adis または ACCEPT/DISPLAY 拡張構文と呼ばれるランタイムサポートモジュールを使用します。 Adis は、構成ユーティリティの Adiscf を使用してアプリケーションの条件に合うように構成できます。 Adis の呼び出しは、COBOL アプリケーションからも行え、実行時に構成できます。たとえば、ファンクションキーを使用可能にします。

利点

- ACCEPT/DISPLAY 操作の位置を指定できる
- 1 つ以上のフィールドを取り込み / 表示できる
- ACCEPT/DISPLAY 操作に属性を指定できる
- 入力中の数字項目のフォーマット
- RM 互換 (RM 指令を使用)、MS 2.2 互換 (MS"2" 指令を使用)、X/Open 互換

欠点

- 一部の COBOL 方言に移植できない
- Adis のロードに大量のメモリを消費することがある
- 複数のフィールドの無駄な空白文字を取り込むと FILLER 項目が使用される

1.2 キャラクタインターフェイスでの ANSI ACCEPT/DISPLAY の使用方法

ANSI ACCEPT 構文を使用すると、データ項目を入力したり、曜日、日付、時間をデータ項目に受け取ったりできます。 ANSI DISPLAY 構文は定数やデータ項目の内容の出力を実現します。

利点

- すべての COBOL 方言に移植できる
- ACCEPT/DISPLAY モジュール、Adis のオーバーヘッドがない

欠点

- 位置指定情報を指定できない

- 属性が指定できない
- ACCEPT 操作でのフィールドの型が英数字として扱われる (文字は、数字として定義されていてもフィールドに入力できる)。
- 1 つの ACCEPT 文では単一のフィールドしか入力できない

備考

基本画面出力とキーボード入力のみ

1.3 キャラクタインターフェイスでの Panels の使用方法

Panels は、呼び出しインターフェイスを使用した多くのテキストウィンドウサポートを提供します。COBOL プログラムから任意の数のウィンドウを作成し、操作することができ、同時に 255 個までの表示が可能です。 パネルの表示領域は、画面の実際のサイズまでです。 各パネルには、最大 65,536 文字まで表示できます。たとえば、スクロール、文字や属性のブロック更新、パネルの移動、表示領域のサイズ変更などの多くのパネル操作機能が提供されています。 Adis モジュール (拡張 ACCEPT/DISPLAY 構文をサポート) からの出力は、呼び出しインターフェイスを使用してパネルへ直接出力できます。

利点

- テキストウィンドウ機能の提供
- パネル内のテキストや属性をスクロールできる
- パネルを画面上で移動できる
- パネルの表示領域を変更できる
- ACCEPT/DISPLAY 構文と互換性がある

欠点

- 大きなパネルが多数作成されると大量のメモリを消費する
- Panels モジュールでディスクやメモリリソースが消費される
- ACCEPT/DISPLAY 操作は、直接の画面操作よりも多少遅くなる

1.4 キャラクタインターフェイスでの画面節の使用方法

画面節は、1 つ以上の画面定義を含むデータ部にある 1 つの節です。画面定義には、フィールド、グループ、テキストと属性を含めることができます。フィールドには、編集した絵文字列を入れることができます。また、NO-ECHO、JUSTIFIED RIGHT、および BLANK WHEN ZERO などの関数を入れることもできます。 画面定義は、手続き部で取り込まれ表示されます。

利点

- 画面の指定領域にのみ書き込んで、残りの領域に影響を与えない
- 画面の位置とそのテキストおよびデータを指定できる
- 1 つ以上のフィールドと集団の取り込み / 表示ができる

- ACCEPT/DISPLAY 操作に属性を指定できる
- フィールド入力の取り込み順序を指定できる
- 特定画面に関連する情報がすべて 1 箇所にあって、構文が理解しやすいため、画面のレイアウト表示が読みやすい
- 特定の画面定義された部分のみが保存される (メモリを占有する FILLER 項目を使用できる拡張 ACCEPT/DISPLAY 方式とは対照的)
- Data General (DG) 互換 (DG 命令を使用)、MS 2.2 互換 (MS"2" 命令を使用)、X/Open 互換

欠点

- 一部の COBOL 方言に移植できない
- 画面に多数のテキストとデータが含まれる場合は、複数フィールド ACCEPT よりも多少遅くなる
- ACCEPT/DISPLAY モジュールのメモリオーバーヘッド

備考

データ部に単一の画面定義を入れて、アプリケーションを X/Open 互換にすると便利

1.5 キャラクタインターフェイスでの COBOL システムライブラリルーチンの使用方法

下位レベルインターフェイスは COBOL システムライブラリルーチンによって提供されます。これらのルーチンを使用すると、COBOL プログラムから下位レベル機能にアクセスできます。次の例では、テキストと属性を画面に配置する 1 つの方法を示しています。その他の多くの呼び出しは、画面とキーボード機能にアクセスするために使用できます。

利点

- 高速 - オペレーティングシステムレベルのインターフェイス
- Adis モジュールのオーバーヘッドがない

欠点

- 完全な画面を構築するために多くの呼び出しが必要になることがある
- パラメータが複雑になることがある。たとえば、オフセットの計算が必要になることがある。

備考

マシンとオペレーティングシステムの機能を活用したい場合や画面処理でのメモリオーバーヘッドを最小限に抑えたい場合には便利です。画面が複雑になるとこれらのルーチンに対してより多くの呼び出しが必要になることに注意してください。

1.6 キャラクタインターフェイスでのウィンドウ構文の使用方法

COBOL システムは、端末の画面上に線やボックスを描画したり、実端末上の仮想端末ウィンドウを作成したりする構文をサポートします。 すべての ACCEPT/DISPLAY 文は、現在のウィンドウ (ACCEPT 書き方 1、2 や 3、DISPLAY 書き方 1、DISPLAY WINDOW/LINE/BOX 文を除く) 内で動作します。 この構文では、基礎となる表示を保持したり、回復したりできます。

利点

- COBOL にテキストウィンドウを追加する
- 追加のソフトウェアを必要としない
- 画面上の描画を容易にする

欠点

- 一部の COBOL 方言に移植できない
- 同一の実行単位の Panels への呼び出しには使用できない

備考

COBOL アプリケーションにテキストウィンドウを簡単に追加できるため便利

第 2 章 : 拡張 ACCEPT/DISPLAY 構文

拡張 ACCEPT/DISPLAY 構文 (Adis) は、画面節と拡張 ACCEPT/DISPLAY 構文を提供するランタイムサポートモジュールです。Adis は、構成ユーティリティ Adiscf を使用してアプリケーションの条件に合うように構成できます。Adis の呼び出しは、COBOL アプリケーションからも行え、実行時に構成できます。たとえば、ファンクションキーを使用可能にします。詳細については、『*Adiscf を使用した Adis の構成*』の章を参照してください。

拡張 ACCEPT/DISPLAY 構文は、標準の ANSI ACCEPT 構文以上の機能を提供しています。

次の表で、標準の ANSI ACCEPT および DISPLAY 構文以上の Adis の機能を紹介します。

構文タイプ	機能
ANSI ACCEPT	データ項目入力
	データ項目への曜日、日付、時間の受け取り
ANSI DISPLAY	定数出力
	データ項目の内容の出力
Adis	画面位置と画面属性の指定
	データの取り込みと表示
	- 個別のフィールドの基本データ項目
	- 複数フィールドの集団項目 [1]
	- 画面節の項目
	キーボードのキー構成

[1] 複数フィールドの集団項目。複数フィールド ACCEPT 操作の場合には、FILLER は次のフィールドへスキップする文字数を記述します。DISPLAY 操作では、FILLER は定数間の空白文字数を定義します。FILLER として定義されている全領域は、ACCEPT または DISPLAY 動作の影響は受けません。

これらについては、後で詳しく説明します。

キーボード上のキーを設定し、ACCEPT 文実行中に使用できます。

- Adis キーの種類の識別
- Adis ファンクションキーの使用
- Adis キーの機能へのマッピング

Adis は 4 つのモジュールで構成されています。

モジュール	説明
Adis	主要なサポートモジュール
adisinit	初期化モジュール (拡張 ACCEPT/DISPLAY 構文が最初にロードされたときのみに使用)
adiskey	キーボード処理モジュール
adisdyna	拡張 ACCEPT/DISPLAY 構文動的属性モジュール

これらの関係を次図に示します。

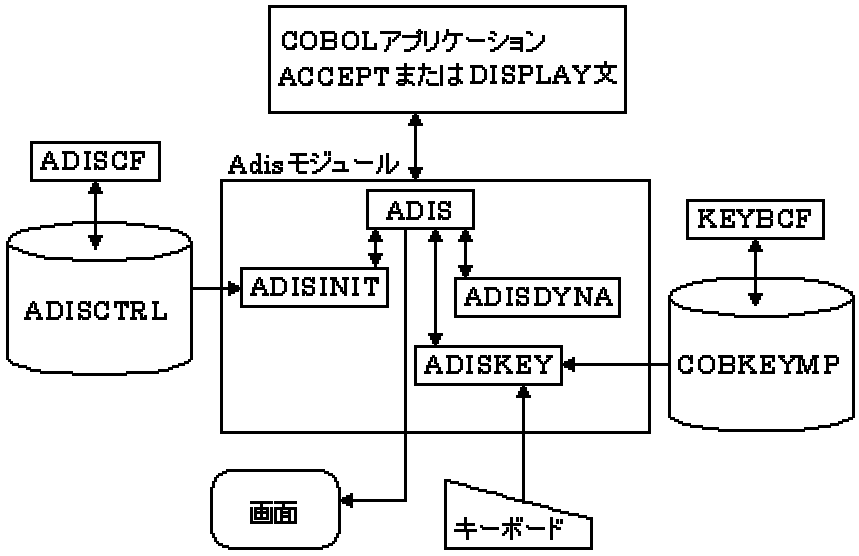


図 2-1：Adis モジュール

2.1 単一フィールドから基本データ項目を取り込む

ACCEPT 文は英数字フィールドと数字フィールドに対して別々に実行されます。

2.1.1 単一フィールドの取り込み

実行時に、ACCEPT および DISPLAY で基本項目を単一フィールドから出し入れして使用するには、次の構文を使用します。

```
display data-item at xxyy ...
accept data-item at xxyy ...
```

パラメータは、次のとおりです。

data-item	プログラムのデータ部で指定された基本項目
xxyy	DISPLAY や ACCEPT フィールドが始まる画面上の位置 (*xx*=line,*yy*=column) を指定します。フィールドの長さは *data-item* の長さと型、および ACCEPT または DISPLAY 文の SIZE 句によって決定されます。

2.1.2 英数字フィールド

ここで使用される「英数字フィールド」という用語は、英数字、英字および英数字編集フィールドすべてを意味します。ACCEPT 文の目的として、英数字編集フィールドは、同じ長さの英数字フィールドとして処理されます。スラッシュ文字 (/) またはゼロのような挿入記号は無視され、X として処理されます。そのため、PIC XX0XX0XX と定義されているフィールドは、PIC X(8) と指定されたものとして処理されます。

フィールドが英字フィールドと定義されている場合は、「A」～「Z」と「a」～「z」の文字と空白文字のみを、フィールドに入力できます。

カーソルは最初、フィールドの先頭位置にあります。

データをフィールドに入力すると、カーソルは次の文字位置へ移動します。矢印キーを使用して、フィールド内のデータに沿ってカーソルを後退させることもできます。また、バックスペースや削除といった編集機能も提供されています。

2.1.3 数字フィールドと数字編集フィールド

数字フィールドと数字編集フィールドへのデータエントリの形式は、次のどれかになります。

- 固定形式
- 自由形式
- RM 形式

Adiscf を使用して希望する形式を選択します。データエントリのデフォルト形式は固定形式です。自由形式を試行して、どちらかの方法を選択してください。特に、単純な数字フィールドへの取り込みの場合に自由形式を試行します。

2.1.3.1 固定形式のデータエントリ

固定形式モードはフォーマットモードとしても知られています。「What you see is what you get」という表現が本形式を表すのに適しています。各キーを押すたびに、PICTURE 文字列に従うようにフィールドは自動的にその形式が整えられます。そのため、拡張 ACCEPT/DISPLAY 構文は、データ部に格納されているのと同じ状態で、そのフィールドを画面に表示し、そのサイズはデータ項目のサイズと同じになります。

DISPLAY 文と同じように、暗黙符号または小数点の付いていない USAGE DISPLAY として定義されているフィールドを取り込む場合のみが有効となります。

データエントリ中には、数字、正符号 (+)、負符号 (-) および小数点 (.) 以外の文字はすべて拒否されます。編集フィールド内への挿入文字は、カーソルを前後に移動したときに、自動的にスキップします。+ または - を押すと、カーソルがフィールド内のどの位置にあっても、符号は自動的に修正されます。

ゼロは、小数点の前の先行記号、または、小数点に続く終了記号として使用される場合のみに、挿入文字として受け付けられます。たとえば、PIC 0.9999 と PIC 999.90 は有効ですが、PIC 0.0009 と PIC 999.000 は無効です。

ゼロ抑制をしていない単純な数字フィールドや数字編集フィールドのデータエントリは、挿入モードが数字フィールドや数字編集フィールドではサポートされていないことを除けば、英数字フィールドの場合のデータエントリと同じです。単純な数字フィールドで小数点文字を入力すると、数字は右詰めされます。たとえば、初期値がゼロである PIC 9(5) として定義されたフィールドは、次のように取り込まれます (下線の付いた文字はカーソル位置を表します)。

初期表示	00000
1 を押す。	10000
2 を押す。	12000
4 を押す。	12400
Back Space キーを押す。	12000
「.」を押す。	00012

ゼロ抑制したフィールドへのデータエントリの処理は異なります。カーソルは、当初、ゼロ抑制されていない最初の文字に位置しています。小数点より前の桁がすべてが抑制されている場合には、カーソルは小数点の上に位置しています。

カーソルが小数点の左側 (つまり、フィールドの整数部) に位置している間は、数字が入力されるたびに小数点に達するまで、カーソルは右に移動します。整数部がすべて埋まるまで、数字は小数点の直前まで挿入されます。次に、カーソルは自動的に小数点第 1 位の数字に移動し、小数位の数字が入力されるたびに先に進みます。

整数部がまだすべて埋まっていないときに、小数点の後の数字を入力したい場合は、小数点 (.) を入力する必要があります。

たとえば、数字編集データフィールドが PIC ZZZ99.99 と定義され、初期値ゼロが入っているとすると、そのフィールドは ACCEPT 操作中に次のように表示されます。

初期表示	00.00
1 を押す。	10.00
2 を押す。	12.00

3 を押す。	123.00
4 を押す。	1234.00
Back Space キーを押す。	123.00
5 を押す。	1235.00
6 を押す。	12356.00
7 を押す。	12356.70
8 を押す。	12356.78

同じフィールドに 123.45 を入力したい場合は、フィールドは次のように表示されます。

初期表示	00.00
1 を押す。	10.00
2 を押す。	12.00
3 を押す。	123.00
「.」を押す。	123.00
4 を押す。	123.40
5 を押す。	123.45

固定形式モードは、最大 32 文字までの長さの数字編集データ項目についてのみ適用されます。このデータフィールドが 32 文字よりも長くなると、次に説明する自由形式モードで自動的に処理されます。ACCEPT 文に SIZE 句が指定されていれば、フィールドも自由形式として自動的に処理されます。

2.1.3.2 自由形式のデータエントリ

自由形式モードは、数字フィールドまたは数字編集フィールド、または、その両方に対して選択できます。これは Adiscf を使用します。

自由形式フィールドへのデータエントリ中には、フィールドは適切な長さの英数字フィールドとして処理されます。 PICTURE 文字列に合うように、そのフィールドの形式を再度整えるのは、ユーザがそのフィールドを抜けるときのみです。数字、符号文字および小数点以外の文字はすべて廃棄されます。

フィールドは、メモリ上で占有するバイト数と同数の文字を画面上で占有します。ただし、暗黙符号や小数点に対して追加文字が割り付けられている場合を除きます。したがって PIC S99V99 と定義されたデータ項目は、固定形式モードでは 4 文字を占有するのに対し、自由形式モードでは画面上で 6 文字を占有します。

プログラムは、DE-EDIT"1" を使用してコンパイルしてください。これは、自由形式では、画面項目の PICTURE 句に準拠しないデータが受け入れられるからです。DE-EDIT"2" を使用し、USING または TO データフィールドに転記した場合、これらのデータはゼロと解釈されます。

データエントリは英数字フィールドの場合と同じです。

拡張 ACCEPT 文が使用されている場合には、SPACE-FILL、ZERO-FILL、LEFT-JUSTIFY、RIGHT-JUSTIFY および TRAILING-SIGN 句は、自由形式の非編集フィールドにしか適用できません (固定形式フィールドの場合は無視されます)。

ACCEPT 文の 書き方 5 の詳細については、ヘルプを参照してください。

2.1.3.3 RM 形式のデータエントリ

RM 形式データエントリは、Adiscf の該当する Accept/Display オプションを使用して選択します。 RM 形式データエントリは、RM/COBOL V2.0 での数字および数字編集 ACCEPT 操作をエミュレートするために提供されています。このモードが選択されると、他の Accept/Display オプションの数字および数字編集データエントリはすべて無視されます。 データエントリは英数字フィールドの場合と同じです。 ACCEPT および DISPLAY 文はすべて、MODE IS BLOCK 句が指定されたものとして動作します。

この形式の数字入力は、COBOL システムにおける画面処理構文用に使用するものではなく、RM/COBOL 用に記述されたプログラムにのみ適用できるものです。

詳細については、『*MODE IS BLOCK 句*』を参照してください。

2.2 単一フィールドでの基本データ項目の表示

データ項目が画面に表示されるときは、メモリ上で占有するバイト数と同じ数の文字を画面上で占有します。 唯一の例外は、EUC などの DBCS 符号化方式で、記憶空間と表示サイズが異なる文字を含む場合です。

2.2.1 表示されるデータ形式 - 例

データの説明	画面上のサイズ	備考
X(5)	5 文字	
N(5)	10 文字	各 PIC N 文字は、メモリ上で 2 バイトを占有します。
G(5)	10 文字	各 PIC G 文字は、メモリ上で 2 バイトを占有します。
9(5)	5 文字	
99.99	5 文字	
Z(4)9	5 文字	
99V99	4 文字	V は暗黙の小数点のため、画面に表示されません。
S9(4)	4 文字	符号は暗黙のため、画面に表示されません。
S9(4) SIGN LEADING		

SEPARATE	5 文字	
9(4) COMP	2 文字	COMP フィールドはバイナリで格納され、9(4) COMP フィールドはメモリ上で 2 バイトを占有します。

この例から、暗黙符号や小数点を含まない USAGE DISPLAY として定義されたフィールドを表示する場合のみが有効になります。フィールドが表示されるときは、必要な場所でデータ項目から画面にバイト単位でコピーされます。

注:ANSI DISPLAY 操作は、表示前に USAGE DISPLAY でない数字項目を USAGE DISPLAY に変換します。

2.2.2 単一フィールドの表示

単一フィールドから ACCEPT および DISPLAY で基本項目を出し入れするには、次の構文を使用します。

```
display data-item at xxyy ...
accept data-item at xxyy ...
```

パラメータは、次のとおりです。

data-item	プログラムのデータ部で指定された基本項目
xxyy	DISPLAY や ACCEPT フィールドが始まる画面上の位置 (*xx*= 行、*yy*= カラム) を指定します。フィールドの長さは *data-item* の長さと型、および ACCEPT または DISPLAY 文の SIZE 句によって決定されます。

2.2.3 表示データの制御シーケンス

表示されているデータに、制御シーケンスを埋め込むことはできません。制御コード (つまり、32 未満の ASCII コード) をデータに強制的に埋め込むと、そのコードに対応する文字が表示されます。たとえば、ASCII コード 7 の表示は、警報を鳴らすかわりにダイヤモンド文字の表示という結果になります (環境によっては、そのような文字は空白文字で置き換えられます)。すべてのハイライトとカーソル制御は、提供の構文を使用して行う必要があります。

2.2.4 ハイライト表示テキストの表示

DISPLAY 文に CRT-UNDER 指定を使用すると、通常表示のテキストから表示が目立つ属性を付けて、データ項目が表示されます。ユーザの環境によって、CRT-UNDER 指定を使用すると、データ項目が下線付き、ハイライト、着色、または反転して表示されます。Adiscf を用いて表示方法を変更できます。

2.3 集団項目の取り込みと表示

集団項目を表示または取り込んでいるときに、その集団内の各基本項目は別々のフィールドとして処理されます。基本項目が FILLER と定義されている場合には、その基本項目は単に適切なサイズの位置決め項目として使用されます (つまり、FILLER フィールドへの取り込み、または FILLER フィールドの表示はできません) 。

2.3.1 集団項目への取り込み

集団項目への取り込み (ACCEPT) の際には、各フィールドは、後述する単一フィールド ACCEPT 文の場合のように取り込まれます。

CURSOR IS 句を使用して、カーソルを明示的に位置付けしないと、カーソルは、当初、最初のフィールドの先頭に位置します。

フィールドの末尾に達すると、カーソルは通常、自動的に次のフィールドの先頭へ進みます。この動作は、Adiscf を用いて自動スキップをオフにすると防げます。矢印キーと次フィールドへ移動および前フィールドへ移動用に設定されたキー (通常、**Tab** キーと **Back Tab** キー) は、カーソルのフィールド間の移動に使用します。

2.3.2 集団項目の表示

次の集団項目があるとします。

```
 01 display-item.
     03 display-item-1                pic x(20).
     03 filler                        pic x(35).
     03 display-item-2                pic 9(10).
     03 filler                        pic x(105).
     03 display-item-3                pic z(4)9.
```

80 文字幅の画面上で、次の文を実行します。

```
display display-item at 0101
display-item-1 is displayed at row 1, column 1,
display-item-2 is displayed at row 1, column 56,
display-item-3 is displayed at row 3, column 11
```

その他の画面領域はすべて DISPLAY 文による影響は受けません。FILLER 項目により画面にデータが表示されることはありません。その結果、各 FILLER で定義された画面上の位置にすでに存在するデータは、DISPLAY 文によって変更されることはありません。

次のデータ項目があるとします。

```
 01 data-item.
```

```
    03 data-char                        pic x occurs 2000.
```

次の文を実行します。

```
display data-item at 0101
```

それぞれが PIC X と定義された 2000 フィールドの表示として処理され、必要とする表示とは異なって表示されます。これを避けるには、データ項目を PIC X(2000) と再定義して表示するか、または、後述の MODE IS BLOCK 句を使用します。

2.3.2.1 MODE IS BLOCK 句

集団項目の ACCEPT または DISPLAY に MODE IS BLOCK 句を追加すると、その集団項目は、その集団項目の全長をもった基本項目であるかのように処理されます。

注:プログラムを IBM-MS、MS"1"、MS"2" または RM 指令を使用してコンパイルすると、コンパイラも MODE IS BLOCK が指定されているかのように集団項目のすべての DISPLAY および ACCEPT 操作を処理します。

2.4 画面節項目の取り込みと表示

画面節項目を含んだ ACCEPT および DISPLAY 操作は、他の ACCEPT または DISPLAY 操作と同様に処理されますが、1 つのみ重要な相違があります。次のコードがあるとします。

```
working-storage section.
01 item-a              pic 9(5).
01 item-b              pic 9(10).
01 item-c              pic x(10).
...
screen section.
01 demo-screen.
    03 blank screen.
    03 line 1 column 1 pic z(4)9 from item-a.
    03 line 3 column 1 pic 9(10) to item-b.
    03 line 5 column 1 pic x(10) using item-c.
```

このコードをコンパイルすると、コンパイラは各画面節のレベル-01 項目 (レコード) ごとに作業領域を設定します。そのため、この例では、3 つのフィールド用データを保持するのに十分な大きさの作業領域が、demo-screen に設定されます。

次の文を実行します。

```
display demo-screen
```

item-a と item-c の内容が、作業場所節から作業領域に転記され、これら 2 つのフィールド用作業領域のデータが画面に表示されます。item-b は、ACCEPT のみのフィールドであるので、表示されません。

次の文を実行します。

```
accept demo-screen
```

item-b と item-c の作業領域にデータが取り込まれ、これら 2 つのフィールドのデータは作業領域から作業場所節に転記されます。 item-a は DISPLAY のみのフィールドであるので、データは item-a には取り込まれません。

データ部から作業領域への転記は、DISPLAY 文を実行したときのみに発生し、作業領域からデータ部への転記は ACCEPT 文の実行中にのみ発生することに注意してください。 そのため、DISPLAY 文をはさまない 2 つの ACCEPT 文が連続して実行されると、2 番目の ACCEPT の実行開始時におけるフィールドの初期の内容は、前の ACCEPT 実行中に作業領域に入れられたフィールドの内容であり、データ部項目の現在の内容ではありません。

これの意味するもう 1 つのことは、データ部と画面節のどちらでも、フィールドは数字編集フィールドとして定義してはならないということです。 数字編集フィールドとして定義すると、データ部項目を作業領域に転記したり、また、その逆の転記をするために、コンパイラは数字編集フィールドを数字編集フィールドに転記することになります。 このような転記の結果は未定義で、予期しない結果になります。 データ部項目と画面節項目が両方とも同じ PICTURE 文字列であっても同じことが言えます。

次のプログラムがあります。

```
working-storage section.
 01 ws-item     pic zz9.99 value 1.23
 screen section.
 01 demo-screen.
     03 pic zz9.99 using ws-item.
 procedure division.
     display demo-screen.
```

DISPLAY 処理の結果は未定義となります。 データ部で画面節の ACCEPT または DISPLAY 文の送り出し項目または取り込み側項目として使用される項目は、常に非編集フィールドである必要があります。

2.4.1 ACCEPT 文でのカーソルの位置付け

CURSOR IS 句で、ACCEPT 操作開始時のカーソル位置をフィールド内のどこにするかを指定でき、ACCEPT 操作の終了時にカーソル位置を返します。 プログラムで CURSOR IS 句を

指定しない場合は、各 ACCEPT 操作ごとにカーソル位置は最初に第一番目のフィールドの先頭になります。

構文

CURSOR IS 句は、特殊名段落で次のように定義されます。

```
special-names.
    cursor is cursor-position.
```

cursor-position は作業場所節で次のように定義されるフィールドです。

```
01 cursor-position.
    03 cursor-row       pic 99.
    03 cursor-column    pic 99.
```

または

```
01 cursor-position.
    03 cursor-row       pic 999.
    03 cursor-column    pic 999.
```

パラメータは、次のとおりです。

cursor-row	カーソルが位置する行。有効な数値は 1 と画面上の行数の間の数です。
cursor-column	カーソルが位置するカラム。有効な数値は 1 と画面上のカラム数の間の数です。

操作

ACCEPT 文が実行されるときは必ず、拡張 ACCEPT/DISPLAY 構文は *cursor-position* で指定された位置に初期カーソルを位置させようとします。もし指定された位置が無効である場合 (つまり、*cursor-row* または *cursor-column* に有効な値が入っていない) には、カーソルは画面上の最初のフィールドの先頭に位置します。

cursor-position に指定されている値が有効な場合には、拡張 ACCEPT/DISPLAY 構文はフィールドすべてを検索して、要求されたカーソル位置がないか調べます。もしあれば、カーソルは要求されたところに位置付けされます。なければ、最初のフィールドの先頭にカーソルは位置付けされます。そのため、カーソル位置を最初のフィールドの先頭に位置付けする場合は、*cursor-row* と *cursor-column* の両方を 1 に設定してください。

定義された位置が、数字編集フィールドの抑制文字または挿入文字の場合には、カーソルはその右側にある最初の使用可能な文字に移動します。その先にデータ項目がない場合は、カーソルは画面上の最初のデータ項目に戻ります。

ACCEPT を終了するときには、*cursor-position* の値が ACCEPT 開始時有効であった場合に、終了キーが押されると、カーソル位置は *cursor-position* に戻ります。 ただし、現在カーソル位置とは同じにならないということに注意してください。拡張 ACCEPT/DISPLAY 構文は ACCEPT 操作の終了時に通常カーソルをフィールドの末尾に移動し、相対的に位置付けされた ACCEPT 文が画面上の正しい位置から開始します。

ACCEPT 操作の開始時に *cursor-position* の値が無効であった場合には、ACCEPT 操作終了時の *cursor-position* の内容は変化しません。

この機能の使用例をあげると、メニュー方式の操作で、オペレータに要求される操作は、必要な選択に対応する画面上の位置にカーソルを移動することのみです。 オペレータの選択は、*cursor-position* の戻り値で決定できます。

2.5 大画面上でのデータの取り込みと表示

25 行よりも大きな画面は、COBOL システムで自動的に検出されます。

ANSI ACCEPT/DISPLAY 文、拡張 ACCEPT/DISLAY 構文、画面節および COBOL システムライブラリルーチンを使用しているプログラムは、大画面上でも正しく実行されなければなりません。 アプリケーションが、実行されている画面よりも大きな画面用に開発されている場合には、複数フィールドの ACCEPT/DISPLAY からはみ出す行は失われます。 さらに、ACCEPT/DISPLAY 文の AT 句で指定された位置が画面の外である場合は、画面は 1 行分上にスクロールします。

2.5.1 CONTROL 句

CONTROL 句は、画面節項目に関連した属性を実行時に定義できるようにします。
CONTROL 句を ACCEPT および DISPLAY 文を処理する画面の 書き方 2 の WITH 指定内、および Adis の ACCEPT および DISPLAY 文内で指定します。

2.6 Adis によるキーボード操作

ここでは、次の内容について説明します。

- キーボードのキーの種類
- CRT STATUS 句
- 強制終了
- ユーザファンクションキー
- Adis キー
- ユーザキーおよび Adis キーリストの定義
- データキー処理
- Adis 互換 GET SINGLE CHARACTER 呼び出し
- ACCEPT/DISPLAY 文の CONTROL 句
- 画面節のサイズ

2.6.1 Adis キーの種類

Adis キーの種類には、ファンクションキー、データキー、シフトキー、およびロックキーなどがあります。

Adis キー				
ファンクションキー		データキー	シフトキー	ロックキー
Adis ファンクションキー	ユーザファンクションキー			
矢印キー	F1…F12	ASCII コード 32-255	Alt	Caps Lock
Tab	Esc		Ctrl	Ins
Back Space			Shift	Num Lock
Enter				Scroll Lock
Del				

2.6.1.1 ファンクションキー

最も一般的なファンクションキーの定義は、タイプライターのキーボードにないキーです。これには、キーボード上に明記されたファンクションキー (通常 **F1**、**F2** ...**F12** キー)、 **Esc** キー、矢印キー、**Tab**、および **Backspace** キーも含まれます。 **Enter** キーもファンクションキーとして扱われますが、後で説明するように、特別な意味があります。

これらのファンクションキーは、2 つに大別されます。

- Adis ファンクションキー。これらは、ACCEPT 文の実行中に Adis により使用されます。Adis ファンクションキーには、矢印キー、**Tab**、**Back Space**、**Del** および **Enter** キーが含まれます。

  通常、これらのキーは ACCEPT 操作中に定義されたものとして処理します。← キーは、カーソルを左に移動し、**Back Space** キーは 1 つ前の文字を削除するなどです。**Enter** キーを除けば、これらのキーは通常 ACCEPT 操作を終了することはありません。ただし、必要に応じて、これらのキーで ACCEPT 操作を終了させることができます。

- ユーザファンクションキー。ユーザファンクションキーと呼ばれる理由は、アプリケーションを記述するときに、これらのキーをどのような目的で使用するかを、プログラマが決定するからです。 これらのキーに割り当てられた定義済みの動作はありません。ユーザファンクションキーには、通常 **F1**、**F2**...**F12** キー、**Esc** キー、およびキーボード上の他の特殊キーが含まれます。

2.6.1.2 データキー

データキーは拡張 ASCII 文字集合に含まれる文字、つまり、ASCII コード 32 ～ 255 に対応する文字を生成するキーです。ACCEPT 処理中にデータキーを押すと、その文字をフィールドに入力します。ただし、キーを完全に使用できなくしたり、キーを押すことで ACCEPT 処理を終了させたりする (ファンクションキーの動作と同様) ことができます。

ASCII コード 0 ～ 31 はデータキーとも制御キーとも見なされます。多くの場合は、制御キーとして処理され、データキーとしての処理は無効とされています。

2.6.1.3 シフトキー

シフトキーとは、押す、および放すの動作を別々のイベントとして見なすキーです。シフトキーの拡張 ACCEPT/DISPLAY 構文の例としては、**Alt**、**Ctrl** および **Shift** キーがあります。シフトキーの詳細については、『*シフトキーの処理*』を参照してください。

2.6.1.4 ロックキー

ロックキーとは、押したときに状態が切り換わるキーです。ロックキーには、**Caps Lock**、**Ins**、**Num Lock**、および **Scroll Lock** の各キーがあります。ロックキーの詳細については、『*ロックキーの処理*』を参照してください。

2.6.2 ファンクションキー処理

ここでは、COBOL システムで、ファンクションキーを使用する方法について説明します。すべての環境で動作する移植可能な方法についても説明します。x"B0" ルーチンを使用する別の方法もあります。このルーチンは、マシンに依存しないのでここでは説明しません。このルーチンを使用している場合、どのような構成で問題が発生するかの詳細については、『*x"B0" COBOL システムライブラリルーチンとの競合*』を参照してください。

2.6.3 押されたファンクションキーの特定

アプリケーションにファンクションキーを使用するときは、どのキーが押されたのかを正確に特定できるようにする必要があります。これを実現するには、次に示したように CRT STATUS 句をプログラムの特殊名段落に記述する必要があります。

2.6.3.1 CRT STATUS 句

```
special-names.
    crt status is key-status.
```

パラメータは、次のとおりです。

key-status　　次の定義をもつプログラムの作業場所節の 3 バイトデータ項目。

```
01 key-status.
    03 key-type      pic x.
    03 key-code-1    pic 9(2) comp-x.
```

```
03 key-code-2   pic 9(2) comp-x.
```

ACCEPT 文が実行されると、どのように ACCPET が終了したかを示す *key-status* が設定されます。*key-status* の個々のフィールドの正確な使用方法は、後述します。ただし、一般に次のように使用されます。

key-type
どのように ACCEPT が終了したかが示されます。戻り値は次のようになります。

「0」 - ACCEPT の正常な終了
「1」 - ユーザファンクションキーによる終了
「2」 - Adis キーによる終了
「3」 - 8 ビットデータキーによる終了
「4」 - 16 ビットデータキーによる終了
「5」 - シフトキーによる終了
「6」 - ロックキーによる終了
「9」 - エラー

これらの値については、ここの後半で詳しく解説します。

key-code-1
ACCEPT 操作を終了させたキーの番号を示します。この番号の正確に意味は、*key-type* で返された値によって変わります。

key-code-2
key-type と *key-code-1* が 0 の場合には、*key-code-2* は ACCEPT 操作を終了させたキーに対するそのままのキーボードコードを含んでいます。単一キー以外のキーストロークのシーケンスが 1 つの機能を実行するように構成されている場合では、最初のキーストロークのコードのみが返されます。

key-type が 4 の場合は、*key-code-2* には ACCPET 操作を終了させた文字の 2 番目のバイトが含まれています。

それ以外の場合は、*key-code-2* の内容は未定義です。

CRT STATUS 句の詳細については、『***言語リファレンス***』を参照してください。

2.6.4 ACCEPT 操作の正常終了

ACCEPT の正常終了では、**key-type** には値 0 が、**key-code-1** には次の値が返されます。

48　(0 を表す ASCII コード)。ACCEPT 操作が **Enter** キーを押して終了された
1　画面の最後のフィールドへの自動スキップで ACCEPT 操作が終了された

例

```
accept data-item at 0101
if key-type = "0"
        if key-code-1 = 48
          display "Enter キーで終了されました"
        else
          display "最後のフィールドへの自動スキップで終了されました"
        end-if
end-if.
```

2.6.5 標準のユーザファンクションキー

最大 128 までのユーザファンクションキーがあります。COBOL システムに標準で提供されるキーは、オペレーティングシステムによって異なりますが、一部のキーは標準です。

キーストローク	ユーザファンクションキー番号
Esc	0
F1	1
F2	2
F3	3
F4	4
F5	5
F6	6
F7	7
F8	8
F9	9
F10	10

キーボードによっては一部のキーが存在しない場合もありますが、存在するキーについては上のように構成してください。さらに、次のユーザファンクションキーを定義します。

キーストローク	ユーザファンクションキー番号
Shift+F1 - Shift+F10	11 - 20
Ctrl+F1 - Ctrl+F10	21 - 30
Alt+F1 - Alt+F10	31 - 40
Alt+1 - Alt+9	41 - 49
Alt+0	50
Alt+-	51

Alt+=	52
PgUp	53
PgDn	54
Ctrl+PgUp	55
Ctrl+PgDn	56
Alt+A - Alt+Z	65 - 90
F11	91
F12	92
Shift+F11	93
Shift+F12	94
Ctrl+F11	95
Ctrl+F12	96
Alt+F11	97
Alt+F12	98

2.6.5.1 ユーザファンクションキーの有効化と無効化

デフォルトでは、Adis ファンクションキーは有効ですが、無効にしたり、ファンクションキーとして使用したりできます。

ユーザファンクションキーを使用できるようにするには、それらのキーを有効にする必要があります。ユーザキーが有効になると、そのキーを押すことで ACCEPT 操作が終了します。そのキーが無効であれば、そのキーは拒否され、警報が鳴ります。

詳細については、『*Adis キーの有効化と無効化*』と『*ユーザファンクションキーの有効化と無効化*』を参照してください。

次の呼び出しを使用してユーザファンクションキーを有効にしたり、無効にしたりします。

```
call x"AF" using    set-bit-pairs
                    user-key-control
```

set-bit-pairs および *user-key-control* は、プログラムの作業場所節で次のように定義されます。

```
01 set-bit-pairs          pic 9(2) comp-x value 1.
01 user-key-control.
   03 user-key-setting    pic 9(2) comp-x.
   03 filler              pic x value "1".
   03 first-user-key      pic 9(2) comp-x.
   03 number-of-keys      pic 9(2) comp-x.
```

パラメータは、次のとおりです。

user-key-setting	キーを無効にする場合は 0 を、有効にする場合は 1 を設定します。
first-user-key	有効または無効にする最初のキー番号です。
number-of-keys	有効または無効にする次のキー番号です。

ファンクションキーは、別の x"AF" 呼び出しによって明示的に変更されるか、または、アプリケーションが終了されるまで、有効または無効に設定されます。ファンクションキーを有効または無効に設定する呼び出しは累積的に大きくなります。たとえば、**F1** ファンクション キーを有効にする x"AF" を呼び出し、**F10** を有効にする 2 番目の呼び出しをすると、両方のキーが有効になります。

例

次のコードでは、**Esc** キー、ファンクションキーの **F1** と **F10** を有効にしますが、その他のユーザファンクションキーは無効になります。

```
* Esc キーを有効にします。
     move 1 to user-key-setting
     move 0 to first-user-key
     move 1 to number-of-keys
     call x"AF" using set-bit-pairs
                      user-key-control
* キー 1 から始まる 126 キーを無効にします。
     move 0 to user-key-setting
     move 1 to first-user-key
     move 126 to number-of-keys
     call x"AF" using set-bit-pairs
                      user-key-control.
* F1 キーと F10 キーを有効にします。
     move 1 to user-key-setting
* F1 キーを有効にします。
     move 1 to first-user-key
     move 1 to number-of-keys
     call x"AF" using set-bit-pairs
                      user-key-control
* F10 キーを有効にします。
     move 10 to first-user-key
     call x"AF" using set-bit-pairs
                      user-key-control
```

2.6.5.2 ユーザファンクションキーの検出

ACCPET 操作中にユーザファンクションキーを押します。キーが有効になっている場合は、ACCEPT 操作が終了して *key-status* のフィールドが次のように設定されます。

データ項目	設定内容
key-type	"1"
key-code-1	押されたユーザキー番号
key-code-2	未定義

2.6.5.3 COBOL の別の方言からのキーの変換

ほとんどの場合は、1 つのユーザファンクションキーのリストを使用するのみで十分です。ただし、COBOL の別の方言からの変換を行う場合は、あるプログラムは標準のユーザキーのリストから返される値を期待しており、また、あるプログラムは COBOL の別の方言でファンクションキーにより返される値を期待するという状況が発生することがあります。このような状況に応えるために、互換キーリストが提供されています。このリストの動作は通常のキーリストとまったく同じです。どのプログラムも、標準のユーザキーのリストまたは互換キーリストのいずれも使用できますが、両方使用することはできません。

キー変換の方法についての詳細は、『*Adis キーでの互換キーリストの使用方法*』を参照してください。

2.6.5.3.1 Adis キーでに互換キーリストの使用方法

システムの全プログラムで互換キーリストを使用する場合は、Adiscf でその使用を選択できます。プログラムごとに別のリストを使用する場合は、次の呼び出しをプログラムに挿入して必要なリストを選択します。

```
call x"AF" using   set-bit-pairs
                   key-list-selection
```

set-bit-pairs と *key-list-selection* は、プログラムの作業場所節で次のように定義されています。

```
01 set-bit-pairs       pic 9(2) comp-x value 1.
01 key-list-selection.
   03 key-list-number pic 9(2) comp-x.
   03 filler          pic x value "1".
   03 filler          pic 9(2) comp-x value 87.
   03 filler          pic 9(2) comp-x value 1.
```

key-list-number が、次の値のどちらかに設定されます。

1 標準のユーザファンクションキーリストを選

択

2 互換キーリストを選択

2.6.5.4 ユーザファンクションキーと妥当性検査用の句

通常、FULL または REQUIRED のような妥当性検査用の句が ACCEPT 文で指定されると、その句の条件が満たされないと、フィールドから出られません。 たとえば、次の文を実行します。

```
accept data-item with required
```

フィールドに何かを入力しない限り、ACCEPT 操作を終了できません。

ただし、有効にされたユーザファンクションキーが ACCEPT 操作中に押されると、例外と見なされ、妥当性検査用の句の条件が満たされていないときでも ACCEPT 操作が終了されます。妥当性検査用の句の条件を満たしていないときに、Adiscf を使用してファンクションキーで ACCEPT 操作を終了させたくない場合は、ACCEPT/DISPLAY オプション 9 を設定できます。詳細については、『*Adiscf を使用した Adis の構成*』の章にある『*構成可能な ACCEPT/DISPLAY オプション*』を参照してください。

2.6.6 Adis キーの使用方法

機能を実行するキーとそれらの機能の区別をしておく必要があります。それは、キーとそれらのキーが実行する機能との間に、実際にはソフトウェア的なマッピングが存在するからです。これは、任意の拡張 ACCEPT/DISPLAY 構文のキーが実行する機能を変更できることを意味しています。

2.6.6.1 標準 Adis 機能へのキーのマッピング

28 の拡張 ACCEPT/DISPLAY 構文キー (0 から 27 で番号が付けられている) があります。各キーに機能がマップされているので、キーが押されると、マップされた機能が実行されます。機能については、『*標準 Adis キーの機能*』で詳しく説明します。

また、拡張 ACCEPT/DISPLAY 構文キーにも名前が指定されていることを注意する必要があります。 ただし、この名前は、各キーを区別するために使用するのみなので、キーが実際に実行する機能については説明する必要はありません。 次のリストでは、キー名前とその名前を取得するために必要なキーストロークを示します。 環境によって構成が異なることがあります。詳細については、『*リリースノート*』を参照してください。

キー番号	機能	標準キー / キーストローク
0	ACCEPT の終了	なし
1	プログラムの終了	Ctrl+K
2	キャリッジリターン	Enter

3	カーソルを左に移動	← (左カーソル)
4	カーソルを右に移動	→ (右カーソル)
5	カーソルを上に移動	↑ (上カーソル)
6	カーソルを下に移動	↓ (下カーソル)
7	ホーム	Home
8	Tab	Tab
9	Back Tab	Back Tab
10	End	End
11	次フィールド	なし
12	前フィールド	なし
13	大小文字の変換	Ctrl+F
14	文字の消去	Back Space
15	文字の再入力	Ctrl+Y
16	文字の挿入	Ctrl+O
17	文字の削除	Del
18	文字の復元	Ctrl+R
19	フィールドの末尾までクリア	Ctrl+Z
20	フィールドのクリア	Ctrl+X
21	画面の末尾までクリア	Ctrl+End
22	画面をクリアします。	Ctrl+Home
23	挿入モード設定	Ins
24	置換モード設定	なし
25	元に戻す	Ctrl+A
26	フィールドの開始	なし
27	マウス位置へ移動	なし

注：Carriage Return (CR) キーは、ここでは **Enter** キーと呼びます。キーボードの種類によっては、CR キーと **Enter** キーがあることがあります。この場合は、拡張 ACCEPT/DISPLAY 構文の Carriage Return キーを CR として、拡張 ACCEPT/DISPLAY 構文キー「ACCEPT の終了」を **Enter** として設定してください。

2.6.6.1.1 標準 Adis キーの機能

次は、Adis キーによって実行される機能のすべてのリストです。機能 0 から 27 までは単純な機能です。機能 55 から 62 までは複雑な機能で、状態に応じてさまざまな処理を実行できます。たとえば、RM 互換に提供される機能には、UPDATE 句が ACCEPT 文で指定されているかどうかに応じて、処理が異なります。

0 - ACCEPT の終了	この機能は、ACCEPT 操作を終了させます。CRT STATUS フィールド (pic 9 display として定義) の最初のバイトは文字「0」(ASCII 48) に設定され、2 番目のバイト (PIC 9(2) COMP-X として定義) は 0 に設定されます。
1 - プログラムの終了	有効になっている場合は、この機能によって中止確認のメッセージが画面に出力され、ユーザに文字の入力を求めます。ユーザが「Y」または「y」以外を入力すると、メッセージが空白になり、通常の ACCPET 操作の処理が継続します。ユーザが「Y」または「y」を入力したり、メッセージが設定されていなかったりすると、STOP RUN が実行されたかのようにプログラムが終了します。
2 - Carriage Return	カーソルが、画面の次の行の第 1 カラムまたはその後のフィールド内で、最初の文字位置となる位置に移動します。該当するフィールドがない場合、処理は行われません。
3 - カーソルを左に移動	フィールド内の前の文字にカーソルを移動します。現在の文字がフィールド内で最初の文字である場合は、カーソルが前のフィールドの最終文字に移動します。現在の文字が画面で最初のフィールドの先頭文字である場合は、エラーがユーザに通知されます。
4 - カーソルを右に移動	フィールド内の次の文字にカーソルを移動します。現在の文字がフィールドの最終文字である場合は、カーソルが次のフィールドの先頭文字に移動します。現在の文字は画面で最後のフィールドの最終文字である場合は、エラーがユーザに通知されます。
5 - カーソルを上に移動	カーソルが次の非保護文字位置、つまり、現在位置の真上の行にあるフィールド内の位置に移動します。
6 - カーソルを下に移動	カーソルを現在位置の真下の行にある、次の非保護文字位置に移動します。 カーソルを上に移動 / カーソルを下に移動 (機能 5 および 6) の場合は、フィールド内の位置ではあるが、挿入文字または抑制された数字を含むことによって保護されている位置が見つかると、カーソル移動機能は CURSOR IS 句で定義された項目と同じ規則に従います。このような場合は、カーソルがフィールド内の最初の非保護文字位置に移動します。このような位置がすべて保護されている場合は、カーソルはフィールド内の最後の非保護文字位置に移動します。
7 - 画面の開始点へ移動	現在の画面の最初の非保護文字位置にカーソルを移動します。
8 - 次のタブ位置へ移動	カーソルを次のカラムタブストップ位置に移動します。カーソル

	は、現在のフィールド (または複数行フィールドの行) の末尾の後の、最初の文字位置よりも後ろには移動できません。
9 - 前のタブ位置へ移動	カーソルを前のカラムタブストップ位置に移動します。 カーソルは、現在のフィールド (または複数行フィールドの行) の先頭の前の、最後の文字位置よりも前へは移動できません。
10 - End	カーソルを次の順に非保護文字位置へ移動させます。 1 複数行の英数字フィールド内における現在行の最後の文字位置 2 現在のフィールドの最後の文字位置 3 現在の画面で最終フィールドの最初の文字位置
11 - 次のフィールドへ移動	画面上の次のフィールドの最初の非保護文字位置にカーソルを移動します。 カーソルがすでに最後のフィールドにあり、このキーに対して ACCPET の自動スキップが有効になっていない場合は、カーソルがフィールドの最後の文字位置に移動し、要求は不成功になったと見なされます。
12 - 前のフィールドへ移動	現在のフィールドの最初の非保護文字にカーソルを移動します。 カーソルがすでに先頭位置にある場合は、前のフィールドの先頭文字位置に移動します。 カーソルが最初のフィールドの先頭文字位置にすでにある場合は、要求が不成功になったと見なされます。
13 - 現在の文字の大小文字変換	現在のカーソル位置にある文字を取得し、アルファベットの場合は大文字 / 小文字を変換し、そのキーが入力されたかのように処理します。 数字や漢字のフィールドでは機能しません。 アルファベット以外の文字で大文字 / 小文字を変換する処理は、単純にその文字を再度入力する処理として扱われます。 この機能を使用しても、大文字保持機構 (文字を小文字にした場合に、単純に大文字に戻す機能) を無効にできません。
14 - Back Space 文字	論理的に現在の文字位置の前にある非保護文字位置にカーソルを移動し、そこにあった文字を再入力バッファにプッシュし、復元バッファにからポップした文字で置き換えます。 挿入モードのときは、フィールド内に残っている文字の位置を 1 つのみ移動して、末尾を埋めるようにオーバーフローバッファから文字をポップすることによって、削除を行います。 どちらの場合も、関連するバッファが空だったり、次の文字がフィールドで有効でなかったりしたときは、空白文字またはゼロ (フィールドの型による) が使用されます。 数字フィールドの小数点位置の左にあるときは処理が多少異なりますが、論理的な結果は同じです。 文字を入力した結果と逆になります。 論理的な要件から生じる異常な影響の 1 つは、文字が最後の位置に入力されたときにカーソルがフィールドから出せないとこ

	ろでは、文字を置き換えるまでカーソルを移動できないことです。
15 - 文字の再入力	復元バッファから文字をポップし、それがあたかもキーボードから受け取られたかのように処理を続行します。バッファが空の場合、または、文字が現在フィールドでは不正となる場合は、エラーが通知されます。
16 - 単一文字の挿入	空白文字またはゼロ文字 (フィールドの種類による) を、フィールドに沿って文字列を移動させてその文字用の空間を作成して、現在カーソル位置に挿入します。数字フィールドの小数点の左側の場合を除いて、有意文字は末尾から押し出されることがあります (失われる文字はオーバフローバッファに押し込まれ、エラーがユーザに通知されます)。数字フィールドの小数点の左側のときは、有効数字が失われる場合、挿入は成功しません。または、カーソルが左端の桁位置にある場合、挿入は成功しません (それは、このような状態での数字挿入処理は、現在の桁の前に挿入することを意味しているからです)。
17 - 文字の削除	現在カーソルの位置にある文字を復元バッファにプッシュし、フィールド内の残りの文字列を左に 1 文字位置のみ移動し、末尾を埋めるために 1 文字オーバフローバッファからポップします。オーバフローバッファが空の場合は、空白文字またはゼロ (フィールドの種類による) が使用されます。 数字フィールドの小数点の左側に対する動作とは多少異なりますが、論理的な効果は通常同じです。つまり、文字を挿入または復元する効果を逆にする論理的な効果です。数字フィールドの場合には、文字は廃棄され、復元バッファにはプッシュされません。
18 - 文字の復元	この機能の影響と制限は、挿入される文字が復元バッファからポップされることを除けば、文字挿入 (上記 16) の場合と同じです。バッファが空であるか、または、ポップされた文字が現在フィールドで有効でない場合は、エラーがユーザに通知されます。 この機能は数字フィールドでは利用できません。
19 - フィールドの末尾までクリア	現在フィールドの現在カーソル位置およびその右側にある文字が復元バッファにプッシュされ、空白文字またはゼロ (フィールドの種類による) で置き換えられます。処理は左から右に行われ、後続非有意空白文字およびゼロを含みます。カーソルは移動しません。複数行の英数字フィールドは、この処理のために、行境界で分割されているものとして扱われます。
20 - フィールドのクリア	最初の文字位置でフィールドの末尾までをクリアする場合のように、現在フィールド (複数行の英数字フィールドの場合は、現在行) のすべての内容が復元バッファにプッシュされます。フィールドへは、空白文字 (英数字) またはゼロ (数字) が転記されます。カーソルは、そのフィールドの種類と形式に対して定義されている初期位置に配置されます。
21 - 画面の末尾までクリア	フィールド末尾のクリアに対して定義された動作が現在フィール

	ド (または行) 上で実行され、以降のフィールドには空白文字またはゼロが適切に転記されます。
22 - 画面のクリア	画面上のすべてのフィールドは、空白文字やゼロを適切に転記し、カーソルはそのホーム位置 (最初のフィールドの最初の非保護文字) に移動します。復元や再入力バッファは、この機能でクリアされます。
23 - 挿入モード設定	現在の編集モードを挿入に設定し、関連する構成済みのインジケータを表示またはクリアします。フィールド外のインジケータは、この機能によりクリアされます。
24 - 置換モード設定	現在の編集モードを置換に設定し、関連する構成済みのインジケータを表示またはクリアします。フィールド外のインジケータは、この機能によりクリアされます。 挿入モードと置換モードは、英数字フィールドのみ適用されます。モードフラグは、数字フィールドでは抑制され、他の英数字フィールドに移動すると回復します。 置換モードは、上書きモードとも呼ばれます。
25 - 元に戻す	現在フィールド (または行) を、カーソルが最後にそのフィールドへ移動したときの状態に回復します。現在のフィールド外の隠れた MOVE 操作 (画面クリア、または、画面末尾までクリア) はフィールドタブや自動スキップのような明白な操作と同様に、編集を元に戻すことはできません。
26 - フィールドの開始	カーソルを次の順に非保護文字位置へ移動させます。 1 複数行の英数字フィールドの現在行の最初の文字位置 2 現在フィールドの最初の文字位置 3 現在の画面の最初の文字位置

2.6.6.1.2 Adis の特殊なマッピング機能

55 - RM フィールドのクリア	この機能は、RM 互換を提供しています。ACCEPT 操作中に、UPDATE 句が指定されると、画面の開始点へ移動 (機能 7) が実行されます。指定されていないときは、フィールドのクリア (機能 20) が実行されます。
56 - RM Back Space	この機能は、RM 互換を提供しています。ACCEPT 操作中に、UPDATE 句が指定されると、カーソルを左に移動 (機能 3) が実行されます。指定されていないときは、Back Space 文字 (機能 14) が実行されます。
57 - RM Tab	この機能は、RM 互換を提供しています。ACCEPT 操作中に、UPDATE 句が指定されると、ACCEPT の終了 (機能 0) が実行されます。指定されていないときは、処理は行われません。
58 - 挿入トグル	現在のモードが挿入モードの場合は、置換モード設定 (機能 24) を実行します。それ以外は、挿入モード設定 (機能 23) が実行されます。

59 - 置換トグル	現在のモードが置換モードの場合は、挿入モード設定 (機能 23) を実行します。それ以外は、置換モード設定 (機能 24) を実行します。
60 - 前方 Tab	複数フィールド ACCEPT の場合は、次のフィールドへ移動 (機能 11) が実行されます。それ以外は、次のタブ位置へ移動 (機能 8) が実行されます。
61 - 後方 Tab	複数フィールド ACCEPT の場合は、前のフィールドへ移動 (機能 12) が実行されます。それ以外は、前のタブ位置へ移動 (機能 9) が実行されます。
62 - 復元	現在のフィールドが数字や復元バッファが空の場合は、単一文字の挿入 (機能 16) が実行されます。それ以外は、文字の復元 (機能 18) が実行されます。
255 - 未定義のマッピング	この値は、特定の機能にどのキーも設定されていないときに使用します。

2.6.6.1.3 Adis キーの機能へのマッピング

一般に、拡張 ACCEPT/DISPLAY 構文キーは同じ名前の機能へマップされます。そのため、<←> キーはカーソルを左に移動し、**Back Space** キーは一つ前の文字を消去します。 ただし、デフォルトで異なる機能へマップされるキーもあります。

キー	機能
Carriage Return	ACCEPT 操作の終了。機能 0
Tab	次のフィールドへ移動します。機能 11
Back Tab	前のフィールドへ移動します。機能 12
挿入モード設定	挿入トグル。機能 58

そのため、キーボードで **Enter** キーを押すと、ACCEPT が終了します。これは、**Enter** キーが Carriage Return キーにマップされているため、Carriage Return キーは ACCEPT を終了します。

注: 機能はマップされるので、番号による呼び出しルーチン x"AF" を使用して、Adis キー 8 と 9 をそれぞれ有効、無効にするには、キー 11 と 12 を参照する必要があります。

この段階では、キーをマッピングするという概念は、不必要な混乱を招くかも知れません。この概念は、COBOL の他の方言をエミュレートするときに大変有用になってきます。 たとえば、Microsoft COBOL V2.2 では、**Enter** キーを押すと、ACCEPT を終了するのではなく、次のフィールドへ移動します。これをエミュレートするには、単にキー 2 (Carriage return) のマッピングを 0 (ACCEPT の終了) から 11 (次のフィールドへ移動) へ変更することで容易に行えます。

Adiscf の機能マッピング画面で、Microsoft COBOL V2.2 互換設定を見れば、エミュレートが行われていることが分かります。

RM/COBOL V2.0 互換設定の場合は、RM/COBOL でのキーの動作をエミュレートするために、多くの変更がデフォルトのマッピングに含まれています。

キーが 255 という値にマップされると、ACCEPT 操作中にそのキーは何の機能も実行しません。

2.6.6.2 特殊な Adis 機能へのキーのマッピング

拡張 ACCEPT/DISPLAY 構文キーマッピングのすべての標準機能は、コンテキストにかかわらず、常に同じ機能を実行します。たとえば、次のフィールドへ移動機能では、常に次のフィールドへの移動を試行します。ただし、コンテキストに応じて異なる動作をする機能もあります。これらの機能については、『*Adis の特殊なマッピング機能*』で説明しています。

たとえば、挿入モード設定キー (キー番号 23) は、通常、機能 58 (挿入トグル) へマップされます。これは、**Ins** キーを繰り返し押すと、挿入モードと置換モードへ交互に切り替わることを意味しています。

2.6.6.3 プログラムからの Adis キーマッピングの変更

次の呼び出しを使用します。

```
call x"AF" using        set-map-byte
                        adis-key-mapping
```

set-map-byte と *adis-key-mapping* は、プログラムの作業場所節で次のように定義されます。

```
01 set-map-byte        pic 9(2) comp-x value 3.
01 adis-key-mapping.
   03 adis-map-byte    pic 9(2) comp-x.
   03 adis-key-number pic 9(2) comp-x.
```

パラメータは、次のとおりです。

adis-map-byte　マップするキーの機能番号を設定します。

adis-key-number　変更するキー番号を設定します。

例

次のコードでは、**Back Space** キー (キー番号 14) の動作をカーソルを左に移動する (機能 3) に変更し、**Tab** キー (キー番号 8) で **Tab** 機能 (機能 8) を実行するように変更します。

```
* Back Space キーのマッピングを変更
```

```
      move 14 to adis-key-number
      move 3 to adis-map-byte
      call x"AF" using set-map-byte
                       adis-key-mapping
* Tab キーのマッピングを変更
      move 8 to adis-key-number
      move 8 to adis-map-byte
      call x"AF" using set-map-byte
                       adis-key-mapping
```

2.6.6.4 x"B0" COBOL システムライブラリルーチンとの競合

x"B0" COBOL システムライブラリルーチンは、ファンクションキーを定義するもう 1 つの方法です。UNIX や他の環境ではサポートされていないことがあります。ただし、x"B0" ルーチンではなく、ここで定義されているファンクションキーの検出方法を使用することをお奨めします。

一般に、x"B0" ルーチンを使用して、ACCEPT 操作を終了させるファンクションキーを定義します。ただし、1 つのみ制限があります。x"B0" を使用する場合は、Carriage Return キーが ACCEPT の終了機能にマップされている必要があります。これがこの製品では標準です。Carriage Return キーのマッピングを変更する場合や、マッピングを変更する設定を使用する場合 (Microsoft COBOL V2.2 または IBM COBOL 1.0 互換) は、x"B0" 呼び出しによって設定されたキーは、ACCEPT 操作を終了しません。そのかわりに、Carriage Return キーがマップされている機能を実行します。

2.6.6.5 Adis キーの有効化と無効化

次の呼び出しを使用します。

```
 call x"AF" using  set-bit-pairs
                   adis-key-control
```

set-bit-pairs と a*adis-key-control* は、プログラムの作業場所節で次のように定義されます。

```
01 set-bit-pairs          pic 9(2) comp-x value 1.
01 adis-key-control.
   03 adis-key-setting    pic 9(2) comp-x.
   03 filler              pic x value "2".
   03 first-adis-key      pic 9(2) comp-x.
   03 number-of-adis-keys pic 9(2) comp-x.
```

パラメータは、次のとおりです。

adis-key-setting　次のように、影響を受けるキーの動作を定義します。

0　キーは無効です。ACCEPT 操作中にキーが押されると、拒否されます。

1 キーはファンクションキーとして動作します。ACCEPT 操作中にキーが押されると、ACCEPT 操作が終了します。

2 ACCEPT 操作中に、キーは標準の処理を行います。これは、デフォルト設定です。

3 カーソルが現在のフィールドから出ない限り、キーは通常の処理を続けます。カーソルが現在のフィールドから出ると、ファンクションキーのような動作をします。

first-adis-key 影響を受ける最初のキー番号です。

number-of-adis-keys 影響を受ける次のキー番号です。

2.6.6.6 Adis ファンクションキーの検出

Adis キーをファンクションキーとして動作するように設定している場合は、ACCEPT 操作を終了して、*key-status* が次の値で返されます。

データ項目	設定内容
key-type	"2"
key-code-1	押された拡張 ACCEPT/DISPLAY 構文キー番号。このキー番号は、キーマップされた機能番号ではありません。
key-code-2	未定義

例

次のプログラムは、**Tab** と **Back Tab** キーをファンクションキーとして動作するように設定します。また、← と → キーを、カーソルがフィールドから出る場合に、ファンクションキーとして動作するように設定します。

```
* Tab (キー 8) と Back Tab (キー 9) キーを次のファンクションキーとして
* 動作するように設定します。
     move 1 to adis-key-setting
     move 8 to first-adis-key
     move 2 to number-of-adis-keys
     call x"AF" using set-bit-pairs
                      adis-key-control
* カーソルがフィールドから出る場合にのみ、
* ← キー (キー 3) と → キー (キー 4) がファンクションキー
* として動作するように設定します。
     move 3 to adis-key-setting
     move 3 to first-adis-key
     move 2 to number-of-adis-keys
```

```
      call x"AF" using set-bit-pairs
                       adis-key-control
      accept data-item at 0101

      if key-type = "2"
        evaluate key-code-1
         when 3
            display "cursor-left caused the cursor to " &
-               "フィールドから出ました"
         when 4
            display "cursor right caused the cursor to " &
-               "フィールドから出ました"
         when 8
            display "Tab キーが押されました"
         when 9
            display "Back Tab キーが押されました"
        end-evaluate
      end-if.
```

2.6.7 ユーザキーリストと Adis キーリストの両方でのキーの定義

一般に、キーはユーザキーリストまたは拡張 ACCEPT/DISPLAY 構文キーリストのどちらか一方で定義し、両方では定義しません。ただし、同じキーを両方のリストで定義してもかまいません。

両方のリストで定義されているキーが ACCEPT 操作中に押されると、次の一連の処理が実行されます。

1. キーはユーザキーリストで定義されているか？
2. 定義されていないときは、5 へ。
3. ユーザキーが有効になっているか？
4. 有効になっているときは、*key-type* に「1」を、 *key-code-1* にキー番号を返します。
5. キーは拡張 ACCEPT/DISPLAY 構文キーリストで定義されているか？
6. 定義されていないときは、キーが未定義であることをユーザに通知します。
7. 有効、無効、またはファンクションキーとして動作の設定状態に応じて、拡張 ACCEPT/DISPLAY 構文キーの動作を実行します。

2.6.8 データキーの処理

データキーは、拡張 ASCII 文字集合の 256 キーです。通常、ACCEPT 操作中にこれらのキーの内の 1 つを押すと、文字が単にフィールドに入力されます。この処理の例外は、0 から 31 の範囲の ASCII コードをもつキー、つまり制御キーです。これらは通常は無効になっています。

2.6.8.1 データキーの制御

キーボードの大部分のキーと同じように、データキーを無効にしたり、ファンクションキーのように動作させたりして、ACCEPT 操作を終了できます。これを行うには、次の呼び出しを使用します。

```
call x"AF" using     set-bit-pairs
                     data-key-control
```

set-bit-pairs と *data-key-control* は、プログラムの作業場所節で次のように定義されます。

```
 01 set-bit-pairs              pic 9(2) comp-x value 1.
 01 data-key-control.
    03 data-key-setting        pic 9(2) comp-x.
    03 filler                  pic x value "3".
    03 first-data-key          pic x.
    03 number-of-data-keys     pic 9(2) comp-x.
```

data-key-control のフィールドは、次のように設定してください。

data-key-setting 次のように、影響を受けるキーの動作を定義します。

- 0 キーは無効です。ACCEPT 操作中にキーが押されると、ベルが鳴ってキーが拒否されます。
- 1 キーはファンクションキーとして動作します。ACCEPT 操作を終了します。
- 2 文字は単にフィールドに入力されます。これは、デフォルト設定です。

first-data-key 影響を受ける最初の文字。

number-of-data-keys 影響を受ける文字数。

2.6.8.2 ファンクションキーとして動作するよう設定されたデータキーの検出

データキーがファンクションキーとして動作するように設定されている場合は、キーが押されると ACCEPT 操作が終了して、*key-status* が次のように設定されます。

データ項目	設定内容
key-type	"3"
key-code-1	押されたキーの ASCII コード
key-code-2	未定義

例

```
* 「A」から「Z」までの文字で、
* ACCEPT 操作を終了するように設定します。
     move 1 to data-key-setting
     move "A" to first-data-key
     move 26 to number-of-data-keys
     call x"AF" using set-bit-pairs
                      data-key-control
     accept data-item at 0101
     if key-type = "3"
            evaluate key-code-1
            when 65
            display "A が押されました"
            when 66
            display "B が押されました"
            when 90
            display "Z が押されました"
        end-evaluate
     end-if.
```

2.6.9 シフトキーの処理

Adis は、アプリケーションでキーボードのシフトキー機能を利用するルーチンを提供しています。ここでは、これらのルーチンを説明します。

UNIX:
ほとんどの UNIX 端末では、他の有効なキーとともに押されない限り、**Alt** キーや **Ctrl** キーが押されたことを検出できません。そのため、移植するアプリケーションにこれらのキーを使用しないでください。そのかわりに、キーシーケンス /a と /c が **Alt** と **Ctrl** キーをシミュレートするために使用されます。

2.6.9.1 有効なシフトキーの判断

このルーチンでは、一意なイベントとして検出できるシフトキーを見つけます。

次の呼び出しで、プログラムで使用可能なシフトキーを判断します。

```
call x"AF" using     adis-function
                     adis-parameter
```

adis-function と *adis-parameter* は、プログラムの作業場所節で次のように定義されています。

```
 01 adis-function         pic 9(2) comp-x.
 01 adis-parameter        pic 9(4) comp-x.
```

パラメータは、次のとおりです。

adis-function 44 です。

adis-parameter プログラムで使用可能なシフトキーを返します。*adis-parameter* の 16 ビットは、次のようなシフトキーを示します。ビット 0 が LSB です。

ビット	対応するキー
4 - 15	予約済み
3	**Alt**
2	**Ctrl**
1	**Left Shift**
0	**Right Shift**

特定のビット値 1 は、対応するキーが一意に検出できることを示します。

2.6.9.2 シフトキーの現在の状態を検出

このルーチンは、どのシフトキーが現在押されているかを判断します。

次の呼び出しで、現在どのシフトキーが押されているかを判断します。

```
call x"AF" using    adis-function
                    adis-parameter
```

adis-function と *adis-parameter* は、プログラムの作業場所節で次のように定義されています。

```
 01 adis-function       pic 9(2) comp-x.
 01 adis-parameter      pic 9(4) comp-x.
```

パラメータは、次のとおりです。

adis-function 46 です。

adis-parameter 現在どのシフトキーが押されているかを返します。*adis-parameter* の 16 ビットは、次のようなシフトキーを示します。ビット 0 が LSB です。

ビット	対応するキー
4 - 15	予約済み
3	Alt
2	Ctrl

1	**Left Shift**
0	Right Shift

特定のビット値 1 は、対応するキーが現在押されていることを示します。

2.6.9.3 ACCEPT を終了するシフトキーの有効化と無効化

デフォルトでは、ACCEPT 操作中はすべてのシフトキーが無効になっています。x"AF" 呼び出しでキーを取得します。 このルーチンでは、動的にシフトキーを有効または無効にできます。

次の呼び出しで、シフトキーを有効または無効にします。

```
call x"AF" using    adis-function
                    adis-parameter
```

adis-function と *adis-parameter* は、プログラムの作業場所節で次のように定義されています。

```
  01 adis-function              pic 9(2) comp-x.
  01 adis-parameter.
     03 shift-key-setting       pic 9(2) comp-x.
     03 filler                  pic x value "4".
     03 first-shift-key         pic 9(2) comp-x.
     03 number-of-shift-keys    pic 9(2) comp-x.
```

パラメータは、次のとおりです。

adis-function
1 です。

shift-key-setting
次のように、影響の受けるキーの動作を定義します。

0 キーは無効です。キーが押されても無視されます。

1 キーは有効です。

first-shift-key
影響を受ける最初のキー番号です。有効にするイベントには、次のように番号が付けられています。

0 - **Alt** を押す。
1 - **Alt** を放す。
2 - **Ctrl** を押す。
3 - **Ctrl** を放す。
4 - **Left Shift** を押す。
5 - **Left Shift** を放す。
6 - **Right Shift** を押す。
7 - **Right Shift** を放す。

number-of-shift-keys　影響を受ける次のキー番号です。

例

次のプログラムは、ACCEPT 操作を終了して、ACCEPT 操作が **Ctrl** によって終了したかどうかを確認できるように、**Ctrl** を有効にします。

```
* Ctrl キーを有効にします。
      move 1 to shift-key-setting
      move 2 to first-shift-key
      move 1 to number-of-shift-keys
      move 1 to adis-function
      call x"AF" using adis-function
                       adis-parameter
      accept data-item at 0101
      if key-type = "5"
          evaluate key-code-1
              when 2
                  display "Ctrl キーが押されました"
              when other
                  display "他のシフトキーが押されました"
          end-evaluate
      end-if.
```

2.6.10 ロックキーの処理

Adis は、アプリケーションでキーボードのロックキー機能を利用するルーチンを提供しています。ここでは、これらのルーチンを説明します。

2.6.10.1 有効なロックキーの判断

このルーチンでは、一意なイベントとして検出できるロックキーを見つけます。

次の呼び出しで、プログラムで使用可能なロックキーを判断します。

```
call x"AF" using    adis-function
                    adis-parameter
```

adis-function と *adis-parameter* は、プログラムの作業場所節で次のように定義されています。

```
 01 adis-function       pic 9(2) comp-x.
 01 adis-parameter      pic 9(4) comp-x.
```

パラメータは、次のとおりです。

adis-function 45 です。

adis-parameter プログラムで使用可能なロックキーを返します。*adis-parameter* の 16 ビットは、次のようなロックキーを示します。ビット 0 が LSB です。

ビット	対応するキー
4 - 15	予約済み
3	**Ins Lock**
2	**Caps Lock**
1	**Num Lock**
0	**Scroll Lock**

特定のビット値 1 は、対応するキーが一意に検出できることを示します。

2.6.10.2 ロックキーの現在の状態を検出

このルーチンは、どのロックキーが現在アクティブであるかを判断します。 たとえば、**Scroll Lock** キーは、スクロールロックがオンのときアクティブです。

次の呼び出しで、現在どのロックキーがアクティブであるかを判断します。

```
call x"AF" using    adis-function
                    adis-parameter
```

adis-function と *adis-parameter* は、プログラムの作業場所節で次のように定義されています。

```
 01 adis-function        pic 9(2) comp-x.
 01 adis-parameter       pic 9(4) comp-x.
```

パラメータは、次のとおりです。

adis-function 47 です。

adis-parameter 現在どのロックキーがアクティブであるかを返します。 *adis-parameter* の 16 ビットは、次のようなロックキーを示します。ビット 0 が LSB です。

ビット	対応するキー
4 - 15	予約済み
3	**Ins Lock**
2	**Caps Lock**

1 **Num Lock**

0 **Scroll Lock**

特定のビット値 1 は、対応するキーが現在アクティブであることを示します。

2.6.10.3 ACCEPT を終了するロックキーの有効化と無効化

デフォルトでは、ACCEPT 操作中はすべてのロックキーが無効になっています。x"AF" 呼び出しでキーを取得します。 このルーチンでは、動的にロックキーを有効または無効にできます。

次の呼び出しで、ロックキーを有効または無効にします。

```
call x"AF" using    adis-function
                    adis-parameter
```

adis-function と *adis-parameter* は、プログラムの作業場所節で次のように定義されています。

```
 01 adis-function              pic 9(2) comp-x.
 01 adis-parameter.
    03 lock-key-setting        pic 9(2) comp-x.
    03 filler                  pic x value "5".
    03 first-lock-key          pic 9(2) comp-x.
    03 number-of-lock-keys     pic 9(2) comp-x.
```

パラメータは、次のとおりです。

adis-function 1 です。

lock-key-setting 次のように、影響の受けるキーの動作を定義します。

0 キーは無効です。キーが押されても無視されます。

1 キーは有効です。

first-lock-key 影響を受ける最初のキー番号です。有効にするキーには、次のように番号が付けられています。

0 - Ins Lock

1 - Caps Lock

2 - Num Lock

3 - Scroll Lock

number-of-lock-keys 影響を受ける次のキー番号です。

2.6.10.4 取り込んだ文字の大文字への変更

次の呼び出しを使用します。

```
call x"af" using adis-function
                 adis-parameter
```

adis-function と *adis-parameter* は、プログラムの作業場所節で次のように定義されています。

```
01  adis-function        pic x comp-x value 1.
01  adis-parameter
   03  bitpair-setting  pic x comp-x value 1.
   03  bitpair-section  pic x value "2".
   03  bitpair-index    pic x comp-x value 85.
   03  bitpair-count    pic x comp-x value 1.
```

キーボードから入力された大文字と小文字を取り込めるように Adis を復元するには、bitpair-setting をゼロに設定します。

第 3 章 : Adiscf を使用した Adis の構成

拡張 ACCEPT/DISPLAY 構文 (Adis) の機能の 1 つは、ACCEPT および DISPLAY 文の動作を構成できるという点です。 Adiscf ユーティリティが作成した **adisctrl** ファイルを通じて、Adiscf ユーティリティで完全構成も可能です。

adisctrl は、拡張 ACCEPT/DISPLAY 構文の構成データベースです。 Adisctrl は最大 16 種類の構成を保持でき、いずれも使用可能です。 **adisctrl** ファイル先頭のエントリにより、拡張 ACCEPT/DISPLAY 構文がどの構成を使用するか決定されます。

adisctrl ファイルには、拡張 ACCEPT/DISPLAY 構文が必要とする、マシンに依存しない情報がすべて格納されています。 これには、次のような情報が含まれています。

- 拡張 ACCEPT/DISPLAY 構文が実行される方法。
- 拡張 ACCEPT/DISPLAY 構文がエラーに応じて出力するメッセージ。
- キーの有効/無効。

adisctrl に保持されているどの構成も、構成ユーティリティ Adiscf を用いて変更できます。 このユーティリティはメニュー階層方式で設計されています。 これらのメニューは、いつでも画面の下部に表示でき、使用可能なオプションを一覧表示します。

COBOL システムとともに提供される **adisctrl** には、COBOL の異なる方言をエミュレートするために設定されているいくつかの構成が含まれています。 ほとんどの場合は、これらの構成から Adiscf で使用する構成を選択する以外に必要な作業はありません (Adiscf のメインメニューで選択オプションを使用)。

COBOL システムとともに提供される構成には、次のものが含まれます。

- デフォルト構成

  adisctrl ファイルが見つからなかった場合には、実行時に使用される構成は拡張 ACCEPT/DISPLAY 構文に組み込まれている構成とまったく同じです。 ACCEPT および DISPLAY 文実行中の標準モード処理を提供します。

- RM 互換

  元のプログラムが RM/COBOL V2.0 で記述されている場合に、この構成を選択する必要があります。この構成は RM COBOL の ACCEPT および DISPLAY 文の動作に非常に近いエミュレートをします。

- DG ICOBOL 互換

  Data General の ICOBOL での ACCEPT および DISPLAY 文の動作をエミュレートするために提供されています。

- **IBM V1.0 互換**

  IBM COBOL V1.0 をエミュレートするために提供されています。正確なエミュレーションとは言えず、特に数字および数字編集フィールドへの ACCEPT 動作で相違があります。

- **Microsoft V2 互換**

  Microsoft COBOL V2.2 をエミュレートするために提供されています。正確なエミュレーションとは言えず、特に数字および数字編集フィールドへの ACCEPT 動作で相違があります。

ユーザ独自の構成を作成する場合は、提供された構成を直接変更しないことをお奨めします。かわりに、変更したい構成を別の名前で保存して、それに変更を加えるようにしてください。

カスタム adisctrl ファイルを作成する場合は、プログラム (カレント ディレクトリ、または COBDIR で指定したディレクトリにある) にアクセスできるようにする必要があります。 実行時環境変数アクセスできない場合、省略値が使用されます。

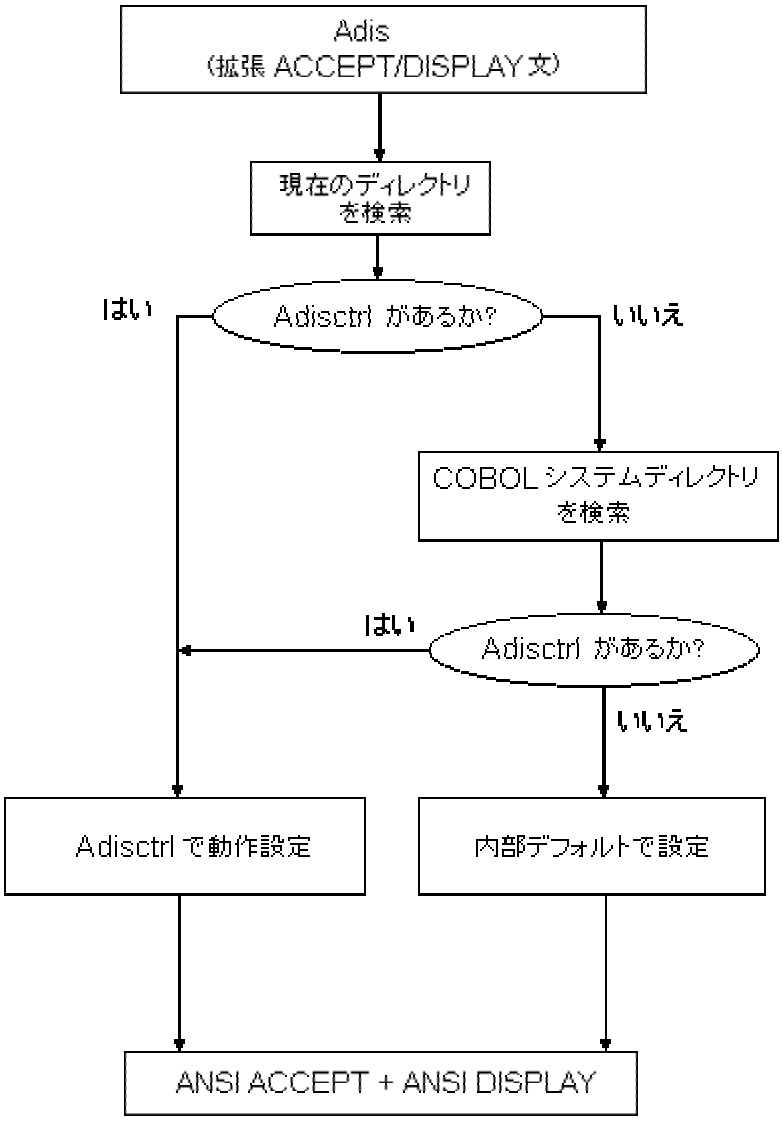
拡張 ACCEPT/DISPLAY 構文には、adisctrl ファイルが存在しない場合に使用されるデフォルト設定が含まれています。 このデフォルトオプションを使用して問題なく動作するアプリケーションは、adisctrl ファイルを含めずに出荷できます。

3.1 Adiscf の起動方法

Adiscf を起動するには、次のコマンド行を入力します。

```
adiscf
```

次の図は、実行時に使用される検索機構を示したもので、どの構成を Adis で使用するかを決定します。



アプリケーションがすべて同じカスタム構成を使用しない限り、adisctrl ファイルを使用しないで作業してください。可能であれば、必要な場合にのみ、現在のディレクトリに adisctrl ファイルを作成するようにしてください。

次のようなメッセージが表示された場合は、Adiscf が以前の製品で使用した adisctrl ファイルを見つけたことを意味します。

```
Old style adisctrl found. Converting to new format.
```

この adisctrl ファイルは、形式が異なります。ただし、Adiscf は古い形式を読み、変換できます。ファイルが新しい形式にいったん変換されると、以前のバージョンの COBOL システムに提供された拡張 ACCEPT/DISPLAY 構文は、それらのファイルを読み込むことはできません。

インストール中に、デフォルトの **adisctrl** ファイルが COBOL システムディレクトリへコピーされることに注意してください。 デフォルトの設定を変更したい場合は、**adisctrl** ファイルを現在のディレクトリへ移動してから変更し、COBOL システムディレクトリへは戻さないようにしてください。

設定を行うと、**adisctrl** ファイルが変更され、拡張 ACCEPT/DISPLAY 構文の内部デフォルトは変更されません。

3.2 メニュー

Adiscf のメニューは、よく使用する Adiscf 機能にすばやくアクセスできるように設計されています。

F1=Help を押すと、どの Adiscf メニューからでもヘルプメニューにアクセスできます。 ヘルプを呼び出したオプションについてのヘルプ画面が呼び出されます。

ここで説明するメニューは、メニュー名のアルファベット順に並んでいます。 メニュー名は、情報行の左側に表示されます。

情報行は現在のメニューに関する情報を表示し、メインメニュー上では現在ロードされている構成に関する情報を表示します。 各メニューには、Adiscf ユーティリティを使用するときと同じような情報行が含まれています。

3.2.1 Adiscf メインメニュー

Adiscf を起動すると、 図 3-1 のようなメインメニューと情報行が表示されます。

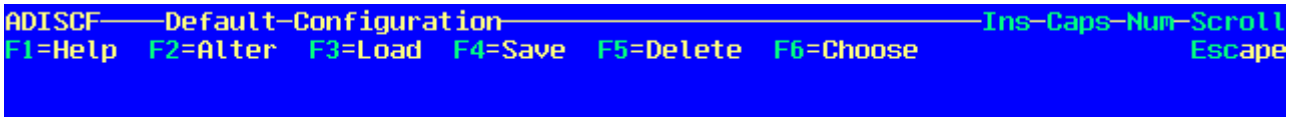


図 3-1：Adiscf メインメニュー

Adiscf メインメニューから多くのサブメニューにアクセスし、次の動作を実行できます。

- 構成を変更する。
- 構成をロードする。
- 構成を保存する。
- 構成を削除する。
- プログラム実行時に使用する Adis の構成を選択する。

3.2.1.1 Adis キーコントロールメニュー

構成変更メニューから **F8=Key Control** を押すと、Adis キーコントロールメニューが表示されます。このメニューを、図 3-2 に示します。

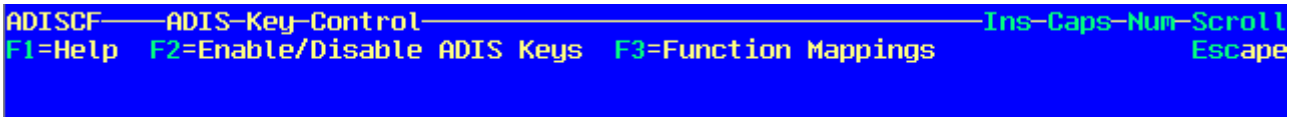


図 3-2：Adis キーコントロールメニュー

このメニューは、次の動作を実行できます。

- 特定のキーによって実行される機能を変更する。
- あるキー動作を Adis のファンクションキーにする。
- あるキーを完全に無効にする。

ここでは、Adis ファンクションキーとは、編集機能を実行する ACCEPT 操作中に使用されるキーです。たとえば、カーソルの移動などです。

3.2.1.2 Accept/Display オプションメニューの変更

構成変更メニューから **F3=Accept/Display Options** を押すと、Accept/Display オプション変更メニューが表示されます。図 3-3 には、Accept/Display オプションメニューが示されています。

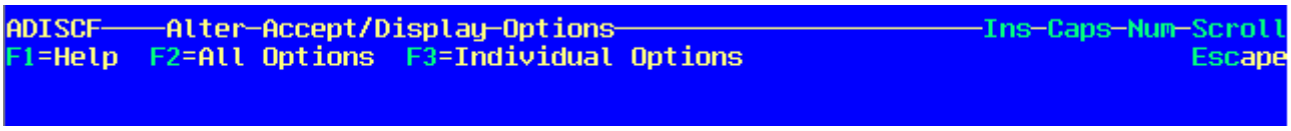


図 3-3：Accept/Display オプション変更メニュー

このメニューを使用すると、ACCEPT 操作中に、カーソルの移動方法、または、フィールドの表示方法を指定できます。使用可能なオプションをすべて変更するか、または、個別に変更するかを選択できます。

3.2.1.3 すべてのメッセージ変更メニュー

メッセージ変更メニューから **F2=All Messages** または **A** を押すと、すべてのメッセージ変更メニューが表示されます。

このメニューを使用すると、ある条件 (カーソルがフィールドの末尾にあるときなど) が生じたときに、Adis が表示するメッセージを変更できます。

3.2.1.4 すべてのオプション変更メニュー

オプション変更メニューから **F2=All Options** または **A** を押すと、すべてのオプション変更メニューが表示されます。

このメニューを使用すると、ACCEPT および DISPLAY 操作の動作を決定するオプションのセット (ACCEPT 操作中に使用できるデータキーの範囲など) を変更できます。

3.2.1.5 構成変更メニュー

Adiscf メインメニューから **F2=Alter** を押して、変更メニューを選択すると、設定のすべて、または、一部を変更できます。図 3-4 には、設定変更メニューが示されています。

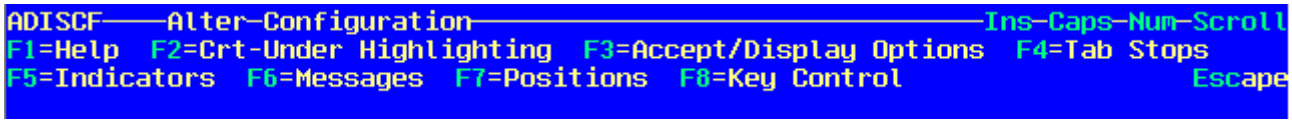


図 3-4：構成変更メニュー

3.2.1.6 Crt-Under ハイライト表示変更メニュー

構成変更メニューから **F2=Crt-Under Highlighting** を押すと、Crt-Under ハイライト表示変更メニューが表示されます。図 3-5 に、Crt-Under ハイライト表示変更メニューが示されています。

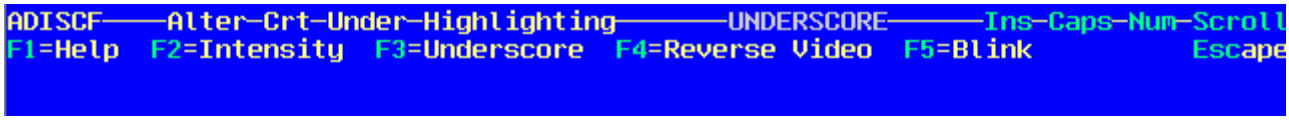


図 3-5：Crt-Under ハイライト表示変更メニュー

このメニューでは、DISPLAY ... UPON CRT-UNDER 文、DISPLAY ... WITH UNDERLINE 文、または UNDERLINE 句がプログラムの画面節で使用されたときに使用するハイライトのタイプを変更できます。

3.2.1.7 機能マッピング変更メニュー

Adis キーコントロールメニューから **F3=Function Mappings** または **M** を押すと、機能マッピング変更メニューが表示されます。

このメニューを使用すると、Adis キーをマップして、それらがそのキーに本来関連していたのとは異なる機能を実行できます。

3.2.1.8 インジケータ変更メニュー

構成変更メニューから **F5=Indicators** を押すと、インジケータ変更メニューが表示されます。このメニューを、図 3-6 に示します。

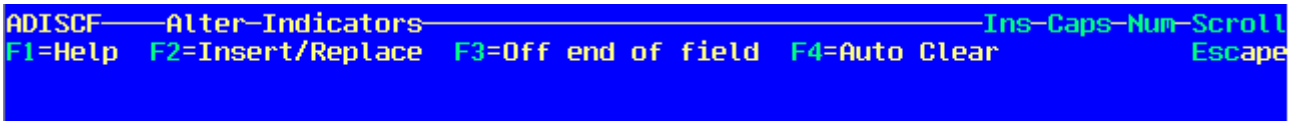


図 3-6：インジケータ変更メニュー

このメニューを使用すると、ACCEPT 操作中に Adis によって表示されるテキストを変更し、さまざまな状態を示すことができます。各メッセージのテキストの長さは、最大 32 文字です。ユーザ独自のメッセージを入力後に、**Enter** キーを押して変更してください。

3.2.1.9 個別メッセージ変更メニュー

メッセージ変更メニューから **F3=Individual Messages** または I を押すと、個別メッセージ変更メニューが表示されます。

このメニューは、現在定義されているメッセージのリストが各メッセージの次に数字を付けた状態で表示します。使用可能なメッセージの次画面を見るには **F2=Next Page** を押し、元のメッセージ画面に戻るには、もう一度 **F2** キーを押します。

3.2.1.10 個別オプション変更メニュー

Accept/Display オプション変更メニューから **F3=Individual Options** または I を押すと、個別オプション変更メニューが表示されます。

このメニューを使用すると、ACCEPT および DISPLAY 操作の使用を管理するオプションの一つを変更できます (ACCEPT 操作中に有効なデータキーの範囲など)。

3.2.1.11 メッセージ / インジケータ位置変更メニュー

構成変更メニューから **F7=Positions** を押すと、メッセージ / インジケータ位置変更メニューが表示されます。

このメニューを使用すると、ACCEPT 操作中に Adis によって通常表示されるメッセージおよびインジケータが表示されるかどうかを管理できます。それらを表示する選択をすると、このメニューを使用して表示される画面上の位置を変更できます。

3.2.1.12 メッセージ変更メニュー

構成変更メニューから **F6=Messages** を押すと、メッセージ変更メニューが表示されます。このメニューを、図 3-7 に示します。

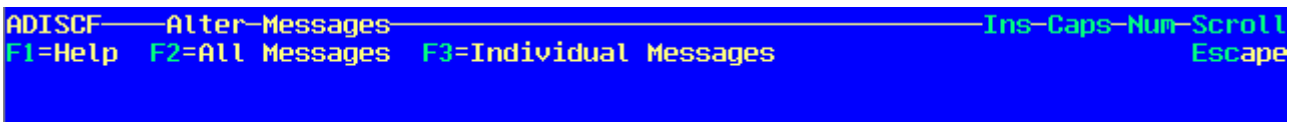


図 3-7：メッセージ変更メニュー

このメニューを使用すると、ACCEPT 操作中に Adis によって表示されるメッセージのテキストを変更し、さまざまなエラー状態を示すことができます。使用可能なオプションをすべて変更するか、または、個別に変更するかを選択できます。

3.2.1.13 タブストップ変更メニュー

構成変更メニューから **F4=Tab Stops** を押すと、タブストップ変更メニューが表示されます。

このメニューを使用すると、実行時に使用されるタブストップを 96 個まで設定できます。タブストップ画面では、画面の上部にルーラーが表示され、現在のタブストップの位置が T という文字で示されます。ルーラーに沿ってカーソルを移動するには、矢印キーの ← と → キーを使用します。

3.2.1.14 構成選択メニュー

Adiscf メインメニューから **F6=Choose** を押すと、構成選択メニューが表示されます。

このメニューを使用すると、プログラムを実行するときに Adis によって使用される構成を選択できます。

3.2.1.5 構成削除メニュー

Adiscf メインメニューから **F5=Delete** または **D** を押すと、構成削除メニューが表示されます。

このメニューを使用すると、**adisctrl** ファイルから既存の構成を削除できます。

3.2.1.16 Adis キーの有効化 / 無効化メニュー

Adis キーコントロールメニューから **F2=Enable/Disable Adis Keys** または **D** を押すと、Adis キーの有効化 / 無効化メニューが表示されます。

このメニューを使用すると、Adis キーによって実行される機能のうち、どれを有効にするかを管理できます。

3.2.1.17 構成ロードメニュー

Adiscf メインメニューから **F3=Load** を押すと、構成ロードメニューが表示されます。

このメニューを使用すると、既存の構成をメモリにロードできます。特定の構成に変更を加える前に、必ずメモリに構成をロードする必要があります。

3.2.1.18 保存メニュー

Adiscf メインメニューから **F4=Save** を押すと、保存メニューが表示されます。このメニューを、図 3-8 に示します。

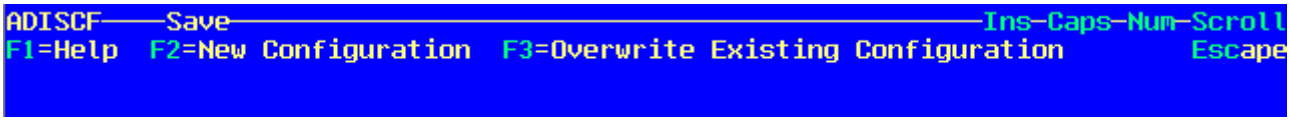


図 3-8：保存メニュー

変更メニューを使用して構成を変更した後に、このメニューを使用して変更を保存できます。新しいバージョンの構成を使用したい場合は、構成を保存する必要があります。構成に変更を加えた後に保存しないで Adiscf を終了しようとすると、保存しないで終了してよいか確認を求められます。

3.3 Adiscf 機能

Adiscf 機能は、メニューからアクセスできます。次の表では、使用可能な機能をアルファベット順に示し、詳細な説明とアクセスするために押すキーを示してあります。

新しいタブストップの追加	**F2、F4、F2**	タブストップを現在のカーソル位置へ設定します。新しいタブストップの位置には、T という文字が表示されます。ACCEPT 操作中は、**Tab** キーが「次のタブストップに移動」機能にマップされている場合は、次のタブストップのみに移動します。
ACCEPT/DISPLAY オプションの変更	**F2、F3**	ACCEPT/DISPLAY オプション変更メニューを表示します。
すべての ACCEPT/DISPLAY オプションの変更	**F2、F3、F2**	各オプション内容とその現在の値を、画面ごとに 1 オプションで順に表示します。 **Enter** キーを押して、ある画面から次の画面に移動するか、または、**Esc** キーを押して Accept/Display オプション変更メニューに戻ります。オプションの値を変更するには、対応するプロンプトで必要な選択番号を入力してください。変更するオプションの数は自由です。
すべてのメッセージの変更	**F2、F6、F2**	各メッセージの現在のテキストとそのメッセージが出力される条件が、画面ごとに 1 メッセージで順に表示されます。次の画面に移動するには、**Enter** キーを押してください。 メッセージを変更するには、対応する画面のプロンプトで新しいメッセージのテキストを入力し、**Enter** キーを押して新しいメッセージを保存して、メッセージ変更メニューに戻ります。変更するメニューの数は自由です。**Esc** キーを押すとメッセージ変更メニューに戻り、現在のメッセージへの変更

		は廃棄されます。
自動クリアインジケータの変更	**F2、F5、F4**	自動クリア用に現在のテキストを表示し、自動クリアインジケータをクリアします。 自動クリアインジケータは、あるフィールドに入力された最初の文字がその文字の入力前にそのフィールドをクリアするような場合に、ACCEPT 操作中に表示されます。 自動クリアインジケータは、ACCEPT 操作の最後に表示され、自動クリアインジケータが現在表示されている場合は、それを削除します。 2 つのフィールドの間を移動するには、矢印キー (↑ と ↓ キー) を使用します。 変更内容を保存するには **Enter** キーを押し、インジケータを変更しないまま自動クリアインジケータ変更メニューを出る場合は **Esc** キーを押します。
構成の変更	**F2**	構成変更メニューを表示します。
CRT-UNDER ハイライト表示の変更	**F2、F2**	CRT-UNDER ハイライト表示変更メニューを表示します。
インジケータの変更	**F2、F5**	インジケータ変更メニューを表示します。
個別の ACCEPT/DISPLAY オプションの変更	**F2、F3、F3**	変更可能な ACCEPT/DISPLAY オプション画面を表示します。 現在選択されているオプションは、ハイライトされます。 次のオプションページを見るには、**F2=Next Page** を押し、元のオプションのページに戻るには、もう一度 **F2** キーを押します。 変更したいオプションの番号を入力するか、または、矢印キーを使用して対応する行にカーソルを置きます。 **Enter** キーを押して、そのオプションを選択します。 そのオプションの説明と現在の値が表示されます。 オプションの値を変更するには、対応するプロンプトで必要な選択番号を入力し、**Enter** キーを押します。 個別オプション変更メニューに戻るには、**Esc** キーを押します。
個別メッセージの変更	**F2、F6、F3**	変更可能なメッセージの画面を表示します。 現在選択されているメッセージはハイライトされます。 次のオプションページを見るには、**F2=Next Page** を押し、元のメッセージのページに戻るには、もう一度 **F2** キーを押します。 変更したいメッセージの番号を入力するか、または、矢印キー (↑ と ↓ キー) を使用して関連行にカーソルを移動します。 **Enter** キーを押して、そのメッセージを選択します。 現在そのエラー状態について構成されているメッセージが表示されます。 画面上のプロンプトで新しいメッセージを入力します。 **Enter** キーを押すと入力したメッセージが保存さ

		れてメッセージのリストに戻り、**Esc** キーを押すと変更内容が廃棄されてメッセージのリストに戻ります。
挿入 / 置換インジケータの変更	**F2、F5、F2**	現在のテキストの挿入、置換を表示し、挿入 / 置換インジケータをクリアします。 挿入インジケータは、ACCEPT 操作中に挿入モードがアクティブになっているときに表示されます。 置換インジケータは、ACCEPT 操作中に置換モードがアクティブになっているときに表示されます。 挿入 / 置換インジケータのクリアは ACCEPT 操作の最後に表示され、他の 2 つのインジケータが現在表示されている場合に、それを削除します。 3 つのフィールドの間を移動するには、矢印キー (↑ と ↓ キー) を使用します。 変更内容を保存するには **Enter** キーを押し、インジケータを変更しないで挿入 / 置換インジケータ変更メニューを出るには **Esc** キーを押します。
キーコントロールの変更	**F2、F8**	拡張 ACCEPT/DISPLAY 構文のキーコントロールメニューを表示します。
メッセージの変更	**F2、F6**	メッセージ変更メニューを表示します。
フィールド末尾超過インジケータの変更	**F2、F5、F3**	フィールド末尾を超える現在のテキストを表示し、フィールド末尾超過インジケータをクリアします。 フィールド末尾超過インジケータは、入力された次の文字がフィールドの末尾を越える場合に、ACCEPT 操作中に表示されます。 フィールド末尾超過インジケータのクリアは、ACCEPT 操作の最後に表示され、現在表示されているインジケータを削除します。 2 つのフィールドの間を移動するには、矢印キー (↑ と ↓ キー) を使用します。 変更内容を保存するには **Enter** キーを押し、インジケータを変更しないで挿入 / 置換インジケータ変更メニューを出るには **Esc** キーを押します。
メッセージとインジケータの位置の変更	**F2、F7**	変更しようとするメッセージとインジケータの 4 つのカテゴリで表示し、各カテゴリで対応する位置を入力するフィールドを表示します。 画面上でフィールドの間を移動するには、矢印キー (↑ と ↓ キー) を使用します。 ACCEPT 操作中にエラーメッセージまたは特定のインジケータを表示させたい場合は、対応するフィールドに Y と入力してください。 それらを表示させたくない場合は、N と入力してください。 エラーメッセージを表示しないと選択した場合でも、メッセージ変更メニューを使用してこのメッセージに関連付けられたテキストに特別な変更をしていない限り、ACCEPT 操作中に中断キーを押すと、中止確認メッセージが必ず表示されます。

		メッセージまたは特定のインジケータを表示する選択をした場合は、ACCEPT 操作中にそれらが画面上に表示されるデフォルト位置を変更できます。 矢印キーを使用してカーソルを対応するフィールドに置き、メッセージまたはインジケータを表示したい新規の行とカラムを入力してください。 行 1 カラム 1 は、画面の左上隅になります。 同じ位置に複数のインジケータが表示され、同時に表示されるように構成する場合は、フィールドの末尾と自動クリアインジケータが、挿入 / 置換インジケータより優先されます。 どのインジケータについても行 255 を選択すると、そのインジケータは、画面の実際の長さにかかわらず、画面の下部の行に表示されます。 エントリ後に **Enter** キーを押して、構成変更メニューに戻り、変更内容を保存してください。 **Esc** キーを押して構成変更メニューに戻ると、変更内容は廃棄されます。
タブストップの変更	**F2、F4**	タブストップ変更メニューを表示します。
構成の選択	**F6**	使用可能な構成が表示され、現在アクティブな構成がハイライトされます。 プロンプトで番号を入力するか、または、矢印キー (↑ と ↓ キー) を使用して、対応する構成名にカーソルを合わせて、必要な構成を選択します。 **これで、この行がハイライトされます。** 選択したい構成がハイライトされた後に、**Enter** キーを押します。 ハイライトされた構成がアクティブな構成になり、Adiscf メインメニューに戻ります。 COBOL システムを使用してプログラムを次回実行したときに、拡張 ACCEPT/DISPLAY 構文は、自動的に選択された構成を使用します。 **Esc** キーを押すと、新しい構成を選択しないで、Adiscf メインメニューに戻ります。
構成の削除	**F5**	既存の構成が表示され、現在の構成がハイライトされます。 プロンプトで番号を入力するか、または、矢印キー (↑ と ↓ キー) を使用して、削除したい構成をハイライトさせます。 削除したい構成がハイライトされた後に、**Enter** キーを押します。 その構成は **adisctrl** から削除され、Adiscf メインメ

		ニューに戻ります。 **Esc** キーを押すと、構成を削除しないで Adiscf メインメニューに戻ります。 **注**:現在選択されている構成は削除できません。これを削除したい場合は、最初に選択メニューを使用して、変更する構成を選択する必要があります。
タブストップの削除	**F2、F4、F3**	現在のカーソル位置でタブストップ (T で表示) を削除します。 最初に、矢印キー (← と → キー) を使用して、カーソルを削除したいタブストップの位置に移動する必要があります。
拡張 ACCEPT/DISPLAY 構文キーの有効化と無効化	**F2、F8、F2**	拡張 ACCEPT/DISPLAY 構文キーで提供されているすべての機能を、各キーの現在の状態とともに表示します。 キーの状態を、次の状態に変更できます。 **状態** **説明** **D** 無効 - ACCEPT 操作中に関連する機能を実行しません。 **E** 有効 - ACCEPT 操作中に関連する機能を実行します。 **F** ファンクションキー - ACCEPT 操作中にファンクションキーとして動作します。 矢印キー (←、→、↑、および ↓ キー) を使用して、画面上のフィールドの間を移動し、変更したいフィールドに必要な文字を入力してください。 エントリ後に、**Enter** キーを押して、拡張 ACCEPT/DISPLAY 構文キーのコントロールメニューに戻り、変更内容を保存してください。 **Esc** キーを押して拡張 ACCEPT/DISPLAY 構文キーのコントロールメニューに戻ると、変更内容は廃棄されます。
エスケープ	**Esc**	上位レベルのメニューに戻るか、または、Adiscf メインメニュー上で Adiscf を終了します。 **Esc** と **Enter** キーが使用可能なオプションの場合に、**Esc** キーを押すと入力した更新を行わずに終了し、**Enter** キーを押すと更新を行った後に終了します。
構成のロード	**F3**	使用可能な構成のリストを表示します。

		構成は、ソフトウェアのバージョンによって異なることがあります。 現在ロードされている構成は、画面上にハイライトされます。 構成を選択するには、プロンプトで番号を入力するか、または、矢印キー (↑ と ↓ キー) を使用して、カーソルを必要な構成を指定する行に移動します。 カーソルを別の構成に移動すると、その構成が画面上にハイライトされ、画面の下部のプロンプト位置にその番号が表示されます。 必要な構成がハイライトされた後に、**Enter** キーを押してください。 Adiscf は選択された構成を **adisctrl** からメモリにロードします。 Adiscf メインメニューに戻ります。 選択された構成は、情報行に表示されます。 変更メニューを選択することによって、この構成を変更できます。 ロードする構成を選択したくない場合は、**Esc** キーを押して、Adiscf メインメニューに戻ります。
Adis キーのマップ	**F2、F8、F3**	サポートされている拡張 ACCEPT/DISPLAY 構文機能のページを表示します。 次の機能ページを見るには **F2=Next Page** を押し、元の機能ページに戻るには、もう一度 **F2** キーを押します。 この機能を使用すると、キーに通常関連付けられている機能を変更できます。 どの機能でもどのキーにもマップできます。 各ファンクションキーには、次に示すように、変更できる関連付けられたフィールドが 3 つあります。 **フィールド　　説明** 妥当性検査　　このフィールドを Y に設定すると、ACCEPT 操作中にカーソルがフィールドを出る前に、すべての妥当性検査基準が満足される必要があ 位置合わせ　　デフォルトでは、このフィールドは N に設定されています。Y に設定すると、ACCEPT 操作中に、カーソルの現在位置からフィールドの末尾ま

		で、フィールドがクリアされます。数字フィールドで、小数点が現在のカーソル位置の右側にある場合は、小数点に位置が合わされます。 **マップされた番号** このフィールドには、関連付けられたキーが現在マップされている機能の番号が含まれます。これを変更したい場合は、必要な機能の番号を入力してください。 次の機能マッピングページを表示するには、**F2=Next Page** を押してください。 矢印キー (←、→、↑、および ↓ キー) でフィールド間を移動します。変更をした場合は、**Enter** キーを押して、拡張 ACCEPT/DISPLAY 構文キーのコントロールメニューに戻り、変更内容を保存してください。変更内容を保存したくない場合、または変更していない場合は、**Esc** キーを押すと、拡張 ACCEPT/DISPLAY 構文キーのコントロールメニューに戻ります。
既存の構成の上書き	**F4、F3**	構成を既存の構成と同じ名前で **adisctrl** に保存します。このオプションは、その名前の既存の構成に上書きするので、注意して使用してください。Adiscf は、既存の構成の番号付きリストを表示します。上書きしたい構成の番号を入力するか、または、矢印キーの ↑ と ↓ キーを使用して、カーソルを関連付けられている行に移動します。指定した名前に構成を保存して、Adiscf メインメニューに戻るには、**Enter** キーを押します。保存しないで Adiscf メインメニューに戻るには、**Esc** キーを押します。 **adisctrl** には、構成を 16 個まで保存できます。これ以上保存しようとすると、次のようなメッセージが表示されます。 `The configuration file is full - No new entries allowed` このメッセージを受け取った場合には、既存の構成を削除するか、または、既存の構成を上書きしないと、新しい構成を保存できません。
保存	**F4**	保存メニューを表示します。
新しい構成の保存	**F4、F2**	構成を新しい構成として **adisctrl** に保存します。この構成の名前を入力するように求められます。**Enter** キーを押すと、入力した名前に構成を保存して、Adiscf メインメニュー

		に戻ります。 保存しないで Adiscf メインメニューに戻るには、Esc キーを押します。
新しいタブストップ設定の保存	**F2、F4、F4**	タブストップの位置の変更を保存し、構成変更メニューへ戻ります。
CRT-UNDER 属性を点滅に設定	**F2、F2、F5**	DISPLAY ... UPON CRT-UNDER 文が実行されたとき使用する属性を点滅に指定します。 情報行に BLINK という語句が表示されます。
CRT-UNDER 属性を太字に設定	**F2、F2、F2**	DISPLAY ... UPON CRT-UNDER 文が実行されたとき使用する属性を太字に指定します。 情報行に INTENSITY という語句が表示されます。
CRT-UNDER 属性を反転表示に設定	**F2、F2、F4**	DISPLAY ... UPON CRT-UNDER 文が実行されたとき使用する属性を反転表示に指定します。 情報行に REVERSE VIDEO という語句が表示されます。
CRT-UNDER 属性をアンダスコアに設定	**F2、F2、F3**	DISPLAY ... UPON CRT-UNDER 文が実行されたとき使用する属性をアンダスコアに指定します。 情報行に UNDERSCORE という語が表示されます。

3.4 構成可能な ACCEPT/DISPLAY オプション

すべてのオプション変更メニューまたは個別オプション変更メニューから変更できる、すべての ACCEPT/DISPLAY オプションのリストを次に示します。 各オプションの番号は、すべてのオプション変更と個別オプション変更メニューによって表示される番号です。

1. ユーザファンクションキーの有効化と無効化

   ユーザファンクションキーを有効化または無効化できるようにします。 これらは、通常、キーボード上のファンクションキーです。 次のどちらかを選択できます。

   - ユーザファンクションキーをすべて無効にします。 ACCEPT 操作中にユーザファンクションキーを押すと、無効なキーとして扱います。 これは、デフォルト設定です。
   - ユーザファンクションキーをすべて有効にします。 ACCEPT 操作中にユーザファンクションキーを押すと、ACCEPT 操作が終了します。

2. 取り込まれるデータキーの範囲

   ACCEPT 操作の入力中に使用できる文字を指定できます。 必要な次のオプション番号を入力するように求められます。

   1　0 ～ 127 の ASCII コード文字を許可する。

   2　0 ～ 255 の ASCII コード文字を許可する。

   3　32 ～ 127 の ASCII コード文字を許可する。

   4　32 ～ 255 の ASCII コード文字を許可する。これは、デフォルト設定です。

1 または 2 のオプションを使用して、0 ～ 31 の範囲内の文字を使用可能にする場合は、まだフィールドにこれらの文字の一部を入力できないことがあります。これは、これらの文字の一部がファンクションキーまたは矢印キーによって生成されるキーシーケンスの始まりを形成することがあるためです。これらのキーが有効である場合は、データキーよりも優先されます。

3. プロンプト文字

   画面節項目の ACCEPT 操作中、または PROMPT 句を指定する ACCEPT 操作中に、フィールドの空の部分に表示される文字を指定できます。システムは、まだデータを入力していないフィールドのすべての部分に選択された文字を表示します。選択された文字は、フィールドの範囲を示すためにも役立ちます。

   このプロンプト文字は、PIC G および PIC N を除くすべての PICTURE の種類に使用されます。

4. PIC G フィールドのプロンプト文字

   画面節項目の ACCEPT 操作中、または PROMPT 句を指定する ACCEPT 操作中に、PIC G フィールドの空の部分に表示される文字を指定できます。

5. ACCEPT の前にフィールドを事前表示します。

   ACCEPT 文の前に自動的にデータフィールドの内容が表示されるようにしたいかどうかを指定できます。ACCEPT 文の前にデータフィールドの自動表示を指定しない場合は、画面はそのままです。次のオプションを選択できます。

   1 カーソルが数字編集フィールドに移動したときに、数字編集を使用して、それらのフィールドの事前表示ができます。他の事前表示は発生しません。
   2 カーソルが数字フィールドに移動すると有効になる数字編集を使用して、すべての数字フィールドを事前表示できます。他の事前表示は発生しません。
   3 フィールドがデータを取り込む直前にすべてのフィールドの事前表示ができます。
   4 データエントリが許可される前にすべてのフィールドを事前表示します。これは、デフォルト設定です。

6. SECURE フィールドへの取り込み

   カーソルの動作方法、および SECURE フィールドへの ACCEPT 中のフィールドの外観を指定できます。指定できるオプションは、次のとおりです。

   1 各文字を入力するときは画面上に文字は表示されませんが、カーソルは次の文字の位置に進みます。これは、デフォルト設定です。
   2 各文字が入力されるたびにアスタリスク (*) が表示され、カーソルが次の文字の位置に進みます。

3 各文字が入力されるたびに空白文字が表示され、カーソルが次の文字の位置に進みます。

7. フィールド間の自動スキップ

現在のフィールドが埋まった後に、カーソルが自動的に次のフィールドに移動するかどうかを指定します。これは、1 つの ACCEPT 文内にある複数データフィールドのみに適用されます。使用できるオプションは、次のとおりです。

1 自動スキップは無効です。次のフィールドに移動するには、Tab キーとして設定されたキーを押すか、または、矢印キー (← キー以外) を使用する必要があります。

2 自動スキップは有効です。現在のフィールドが埋まった後に、カーソルを動かしたり、文字キーを押したりすると、次のフィールドにカーソルが移動します。これは、デフォルト設定です。

このオプションは、画面節項目の ACCEPT 操作に影響を与えません。これらの操作では、自動スキップはデフォルトでオフになっています。原始プログラムに AUTO 句を指定し、自動スキップをオンにできます。 See the AUTO clause for details.

8. ACCEPT の終了

ACCEPT 操作を終了できます。ACCEPT 操作を終了させるには、次のオプションを指定できます。

1 「ACCEPT の終了」キーを押します。これは、デフォルト設定です。

2 カーソルが ACCEPT 操作の最後のフィールドに達したときに、「次フィールド」キーを押します。

3 フィールド間の自動スキップが有効になっている場合は、ACCEPT 操作の最後の使用可能な文字位置でデータ文字を入力するか、または、再入力します。

このオプションは、正常終了のみ制御されます。ファンクションキーが有効になっている場合は、ファンクションキーも ACCEPT 処理を終了させます。

9. ACCEPT がファンクションキーで終了する場合の妥当性検査制御

ACCEPT がファンクションキーを使用して終了するときに、妥当性検査用の句が満たされる必要があるかどうかを指定できます。次のどちらかを選択できます。

1 妥当性検査を行わない。これは、デフォルト設定です。

2 現在のフィールドについて、通常の妥当性検査基準が満たされる必要があります。

10. フィールド末尾の影響

フィールドを埋めた後のデータ入力時に、カーソルにどのような動作をさせたいかを指定できます。次のオプションから 1 つを選択できます。

1 カーソルがフィールドの末尾を越えて移動し、上書きが拒否されます。

2 カーソルはフィールドの末尾に留まったままで、上書きが拒否されます。

3 カーソルはフィールドの末尾に留まったままで、上書きが可能です。これは、デフォルト設定です。

11. フィールドオーバーフローバッファの有効化と無効化

フィールドの末尾から出たときに、オーバーフローバッファにデータが保存されるかどうかを指定できます。次のオプションを選択できます。

1 末尾から出たデータは、オーバーフローバッファに保存されます。これは、デフォルト設定です。

2 末尾から出たデータは、オーバーフローバッファに保存されません。

12. 置換編集モードでのバックスペース中の自動復元

置換編集モードにあるときに、自由書式フィールドでの **Back Space** キーの動作を指定できます。次のオプションを選択できます。

1 自動復元が有効です。文字が削除されると、前に上書きした文字が復元されます。これは、デフォルト設定です。

2 自動復元は無効です。削除された文字は、空白文字で置換されます。

13. 数字編集フィールドへの取り込み

ACCEPT 操作中の数字編集フィールドの外観を指定できます。次のオプションを選択できます。

a 入力は、英数字フィールドとして取り込まれ、そのフィールドを出るときに正常化されて不正な文字が削除されます。

b オプション a と同様ですが、数字、符号、小数点、コンマのみ入力できます。

c 長さ 32 文字までの数字フィールドは、フォーマットモードで取り込まれます。つまり、数字、符号、小数点以外の文字は拒否されます。フィールドは再フォーマットされ、編集記号はデータとして入力されています。32 文字より長いフィールドは、オプション a の場合と同じように取り込まれます。これは、デフォルト設定です。

d オプション c と同様ですが、32 文字より長いフィールドはオプション b のように取り込まれます。

14. 非数字編集フィールドへの取り込み

ACCEPT 操作中の非数字編集フィールドの外観を指定できます。次のオプションを選択できます。

a 形式 9(*m*)V(*n*) の PICTURE 句の符号なし、および、埋め込みの符号付き非数字編集フィールドは、PIC 9(*n*) フィールドが後ろに続く PIC 9(*m*) フィールドとして扱います。SEPARATE 指定をもつフィールドは、PIC S9(*m*+*n*) として扱います。これは、デフォルト設定です。

b PICTURE 句の V をもつフィールドが PIC S9(*m*+*n*) として扱うことを除き、オプション a と同様です。

c 非数字編集フィールドはすべて、英数字フィールドとして扱います。

非ゼロ SIZE 句を指定すると、非数字編集フィールドは、このオプションの設定にかかわらず、すべて自由書式フィールドとして扱われます。

15. 自動クリアまたは事前クリアの有効化と無効化

カーソルが最初にフィールドに入ったときに、フィールドをどのように表示させたいかを指定できます。次のオプションのどれかを指定できます。

a 事前クリアまたは自動クリアは起こりません。これは、デフォルト設定です。

b 事前クリアモード。フィールドは、空白文字またはゼロでクリアされます。**Undo** キーを押すと、元のフィールド内容が復元されます。

c 自動クリアモード。最初のキーストロークが有効なデータ文字である場合は、その文字を処理する前に、そのフィールドが空白文字またはゼロでクリアされます。無効なデータ文字の場合は、自動クリアモードがオフになります。**Undo** キーを押すと、元のフィールド内容が復元されます。

d **Undo** キーを押しても、元のフィールド内容が復元されないことを除き、オプション b と同様です。

16. フィールドが更新されない場合のフィールドの強制更新

フィールドを変更しないでフィールドから出た場合に、フィールド内容がどのように表示されるかを指定します。次のオプションを選択できます。

1 フィールドが変更されない場合は、データ項目は更新されません。これは、デフォルト設定です。

2 フィールドが変更されなくても、データ項目は常に更新されます。このオプションは、フィールドが数字または数字編集であり、元のデータ項目が数字データを含まない場合、または、フィールドが右に桁寄せされていて元のフィールド内容はそうではない場合のみに、効果があります。

17. フィールドの末尾位置の記憶

拡張 ACCEPT/DISPLAY 構文がフィールド内の最後の文字位置を記憶するかどうか
を指定できます。次のオプションを選択できます。

1 拡張 ACCEPT/DISPLAY 構文は、最後の文字位置を記憶しません。これは、デフォルト設定です。

2 拡張 ACCEPT/DISPLAY 構文が、ACCEPT 操作中にフィールドに入力された最後の文字位置を記憶します。このオプションは、RM 形式の数字フィールドの桁寄せを提供するために必要です。

このオプションは、プロンプト文字が無効になっている場合のみに適用されます。

18. RM 形式の数字データエントリ

システムが数字データ項目の RM/COBOL 形式エントリをエミュレートしたいかどうか
を指定できます。次のオプションを選択できます。

1 数字データ項目の COBOL システムエントリが有効です。これは、デフォルト設定です。

2 RM/COBOL 形式の数字および数字編集データエントリが有効です。このオプションが選択されると、他の数字および数字編集オプションはすべて無視されます。

19. 最大フィールドサイズを 1 行に制限

ACCEPT 操作のフィールドのサイズが 1 行に制限されるかどうかを指定できます。
次のオプションを選択できます。

1 フィールドは 1 行に制限されません。これは、デフォルト設定です。

2 フィールドが 1 行に制限されます。

20. ACCEPT 後のカーソル位置の制御

ACCEPT 操作後に、カーソルをどこに置くかを制御できます。次のオプションを選択
できます。

1 カーソルは、現在のフィールドの末尾に達すると、次の文字の位置に移動します。これは、デフォルト設定です。

2 カーソルは、現在の位置のままです。

21. UPDATE 句が暗黙の CONVERT を実行するかどうかの制御

UPDATE 句が CONVERT 句を暗黙とするかどうかを指定できます。次のオプション
を選択できます。

1 CONVERT 句は暗黙とされません。これは、デフォルト設定です。

2 CONVERT 句は暗黙とされます。

このオプションは、RM/COBOL V2.0 および V2.1 の動作をエミュレートするために提供されています。

22. 使用されるファンクションキーリストの選択

コントロールコードをユーザファンクションキーにマップするために、システムがどのファンクションキーリストを使用すべきかを指定できます。次のオプションを選択できます。

1 標準ユーザファンクションキーのリストが使用されます。これは、デフォルト設定です。

2 互換ファンクションキーのリストが使用されます。COBOL システムとともに提供されるリストは、RM/COBOL ファンクションキーリストですが、COBOL の任意の方言と互換性をもつように、これを変更できます。

23. COLUMN + n 句の動作の選択

画面節のこの句の後続文字の数を決定できます。次のオプションを選択できます。

1 COLUMN + 1 句を使用すると、先行するフィールドの直後にそのフィールドが表示されます。2 つのフィールドの間に間隔は空きません。これは、デフォルト設定です。

2 COLUMN + 1 句を使用すると、2 つのフィールドの間に 1 文字分の間隔が空きます。

3 COLUMN + *n* 句で、*n* が 1 より大きい場合は、n 文字分の間隔が空きます。DG 形式の動作が必要なときは、このオプションを 2 に設定する必要があります。

24. カラーが指定されていない場合のデフォルト動作の選択

カラーが指定されていない場合に、画面節項目のデフォルトカラーを指定できます。

1 デフォルトのスクリーンカラーが使用されます。つまり、最後の BLANK SCREEN 句で指定されたカラーが使用されます。最初のカラーは、前景が白で、背景が黒です。これは、デフォルト設定です。

2 デフォルトカラーにかかわらず、前景は白、背景は黒で表示されます。

3 テキストは、前景は白、背景は黒で表示されます。または、最後の BLANK SCREEN 句で設定されたカラーを使用して表示されます。

25. 矢印キーの ←/→ キーでフィールドを出ることができるかどうかの制御

フィールドの先頭 / 末尾に位置しているときに、矢印キーの ←/→ キーを押すと、それぞれ前 / 次のフィールドに移動するかどうかを制御できます。次のオプションを選択できます。

1 キーを押すと、前または次のフィールドに移動します。これは、デフォルト設定です。

2 キーを押しても、フィールドを出られません。

26. 自由形式数字編集の左桁寄せ

自由形式数字編集フィールドに入力したときに、左桁寄せされるかどうかを制御できます。次のオプションを選択できます。

1 フィールドは、左桁寄せされません。これは、デフォルト設定です。

2 RM 数字処理がオフになっている場合には、フィールドは左桁寄せされます。これは、ACCEPT/DISPLAY オプション 18 で制御されます。

27. FULL/REQUIRED フィールドでの妥当性検査制御

ACCEPT 操作を終了する前に、FULL または REQUIRED 句のあるフィールドがすべて適切な条件を満たす必要があるかどうかを制御できます。次のオプションを選択できます。

1 変更されなかったフィールドは、妥当性検査を行いません。これは、デフォルト設定です。

2 FULL または REQUIRED 句のあるすべてのフィールドは、ACCEPT 操作の終了前に有効な内容である必要があります。

28. ファンクションキーとして定義された拡張 ACCEPT/DISPLAY 構文キーの制御

ファンクションキーとして定義された拡張 ACCEPT/DISPLAY 構文キーを制御できます。次のオプションを選択できます。

1 キーに関連付けられたマッピングが、そのキーの動作を決定します。このオプションは、Micro Focus 社の初期の製品と互換性をもたせるために提供されています。

2 ファンクションキーとして設定された任意の拡張 ACCEPT/DISPLAY 構文キーが、ファンクションマッピングにかかわらず、ACCEPT 処理を終了させます。これは、デフォルト設定です。

29. ACCEPT の画面読み込みオプションの制御

現在の画面内容から ACCEPT フィールドの初期内容が読めるようにする機能を制御できます。次のオプションを選択できます。

1 画面からの読み込みは行いません。フィールドの初期内容は変更されません。これは、デフォルト設定です。

2 フィールドの事前表示が無効になっている場合には、フィールドの初期内容が画面から

読み込まれます。これは、ACCEPT/DISPLAY オプション 5 で制御されます。

30. 隠されたフィールドがスキップされるかどうかの制御

隠されたフィールドをスキップできる状況を制御できます。次のオプションを選択できます。

1 隠されたフィールドはスキップされません。これは、デフォルト設定です。
2 隠されたフィールドはスキップされます。

31. 漢字修飾文字の特殊な動作の制御

ACCEPT 操作中に漢字修飾文字を指定する場合に、漢字修飾文字、濁音および半濁音の動作を制御できます。次のオプションを選択できます。

1 修飾文字は、他のすべての文字と同じように扱われます。これは、デフォルト設定です。
2 修飾文字を入力すると、前に入力された文字が修飾可能かどうかが、拡張 ACCEPT/DISPLAY 構文によって調べられます。修飾可能な場合は、修飾された文字に置換されます。

このオプションが関係するのは、日本語の 2 バイト文字集合のマシンのみです。

32. タイムアウトを計算するときに使用される単位の選択

タイムアウトを、秒または 10 分の 1 秒の単位で指定できます。

1 TIMEOUT 句で指定される時間の単位は秒です。これは、デフォルト設定です。
2 TIMEOUT 句で指定される時間の単位は 10 分の 1 秒です。

33. キーストロークのたびにタイムアウトがリセットされるかどうかの制御

文字が入力されるたびに TIMEOUT タイマがリセットされるか、または、入力にかかわらず、一定の時間が経過するとタイムアウトするかを指定できます。次のオプションを選択できます。

1 タイマはリセットされません。タイムアウトは、ACCEPT 操作開始後、指定された時間が経過したときに発生します。これは、デフォルト設定です。
2 タイマは文字が入力されるたびにリセットされます。

3.5 構成可能な Adis メッセージ

すべてのメッセージ変更メニューと個別メッセージ変更メニューを使用すると、ACCEPT 操作中に拡張 ACCEPT/DISPLAY 構文によって表示されるメッセージを変更できます。

次のリストには、メッセージが表示される条件が示されています。

Abort - Y/N ?

- 中断キーが押されました。

You must enter enough data to fill the entire field.

- FULL 句で定義されたフィールドが、完全に埋められていません。

You must not leave the field empty.

- REQUIRED 句で定義されたフィールドが、空のままになっています。

You are at the end of the field.

- カーソルがフィールドの末尾にあります。

The cursor is past the end of the field.

- カーソルがフィールドの末尾を越えました。

You have lost data from the end of the field.

- フィールドの末尾以降のデータは消えました。

You cannot insert here.

- この場所では挿入ができません。

You cannot delete here.

- この場所では削除ができません。

That keystroke has no meaning here.

- 無効なキーが押されました。

There is no field beyond here.

- これ以上移動できるフィールドがありません。

You cannot change character case here.

- 大文字小文字の変更ができない場所で、大文字小文字の変更を試行しました。

There is nothing available to retype.

- 打ち直すものがないときに、文字を再入力しようとしました。

There is nothing available to restore.

- 復元するものがないときに文字を復元しようとしました。

The leading part of the number is outside range.

- その数の先行数字が有効範囲にありません。

The trailing part of the number is outside range.

- その数の後続数字が有効範囲にありません。

You have used a sign improperly here.

- 符号が正しくない方法で使用されました。

You cannot use a negative value here.

- 無効な場所で負の値の使用を試行しました。

No message defined.

- 2 バイト文字を入れる空間がありません。

You cannot use more than one decimal point.

- 複数の小数点の使用を試行しました。

You must enter some numeric digits here.

- 数字フィールドに数字以外のデータを入力しようとしました。

第 4 章：キーボード構成ユーティリティ (Keybcf)

キーボード構成ユーティリティ Keybcf は、キーボード上のキーに割り当てられている機能を変更できます。ACCEPT 文の実行中にキーで行われる動作と、プログラムで使用されているファンクションキーを変更できます。

通常、Adis に組み込まれたデフォルトキーは、変更する必要はありません。ただし、COBOL の特定の方言をエミュレートするために、使用されているキーを変更する必要があるかも知れません。

COBOL システムで使用されるデフォルトのキーは、Adis に組み込まれています。Keybcf は、ファンクションキーに使用されるキーストロークと ACCEPT 操作中に Adis が使用するキーを含む **cobkeymp** と呼ばれるファイルを作成します。

4.1 Keybcf の起動

1. コマンド行で次のコマンドを入力します。
2. **keybcf**

   独自の **cobkeymp** ファイルが設定されます。

   または

   COBOL に関連するプログラムグループを作成している場合には、keybcf に対応するプログラムアイコンをダブルクリックします。

   Keybcf は、現在のディレクトリと COBDIR 環境変数で指定されたディレクトリで、**cobkeymp** ファイルを検索します。ファイルが見つかると、編集するかどうか尋ねられます。そのファイルを編集する場合は「Y」、デフォルトのキーで新しいファイルを作成する場合は「N」と答えます。ファイルが見つからない場合には、新しく作成されます。

3. 初期メニューから、オプション番号 (1-4) を押します。

4.2 既存のファンクションキー定義の見直し (オプション 1)

1. **Keybcf** のメインメニューから **1** を入力します。

   このメニューは、Adis、ユーザ、または互換の 3 つのどれかで現在定義されている、すべてのファンクションキーが一覧表示されます。

2. 出力するリストの番号を押します。

オプション **5** は、選択されたリストを画面に表示するか、または、ファイルに書き込むかを切り替えます。 画面表示を選択すると、リスト中の現在定義されているすべてのファンクションキーが 16 進表示されます。 次の画面に移動するには、任意のキーを押します。 リストの末尾に達した場合に、任意のキーを押すと、既存のファンクションキー定義の見直しメニューに戻ります。

3. リストをファイルに出力するように選択すると、**keylist.dmp** と呼ばれるファイルに同じ情報が書き込まれます。 **keylist.dmp** ファイルが存在していない場合に限り、ファイルを上書きするか、または、ファイルの末尾にリストを追加するかを尋ねられます。
   Press "O" to overwrite, "E" to extend.
4. **6** を押して、**Keybcf** のメインメニューに戻ります。

4.3 ファンクションキー定義の修正 (オプション 2)

1. **Keybcf** のメインメニューから **2** を入力します。
2. 変更するファンクションキーのセットを選択します。 Adis、ユーザ、または互換のファンクションキーのリストを選択できます。
3. 選択するファンクションキーのリストの各機能は、個別に表示されます。
4. 新しい機能を実行させようとするキーを押すか、または、**X** を押して、キーに対する 16 進数を入力します。 新しいキーを入力すると、わずかに停止した後に、次の機能へ自動的に移ります。

   限界を越える数のキーを定義しようとすると、次のような警告が表示されます。

   There is not enough room for that sequence

   この警告が表示された場合、新しいキーを追加するには、不必要なキーシーケンスを削除する必要があります。

5. **5** を押して、**Keybcf** のメインメニューに戻ります。

使用可能なキーのリストについては、『*拡張 ACCEPT/ DISPLAY 構文*』の章にある『*ファンクションキー*』を参照してください。

4.4 ファンクションキー定義の保存 (オプション 3)

Keybcf のメインメニューから **3** を押します。

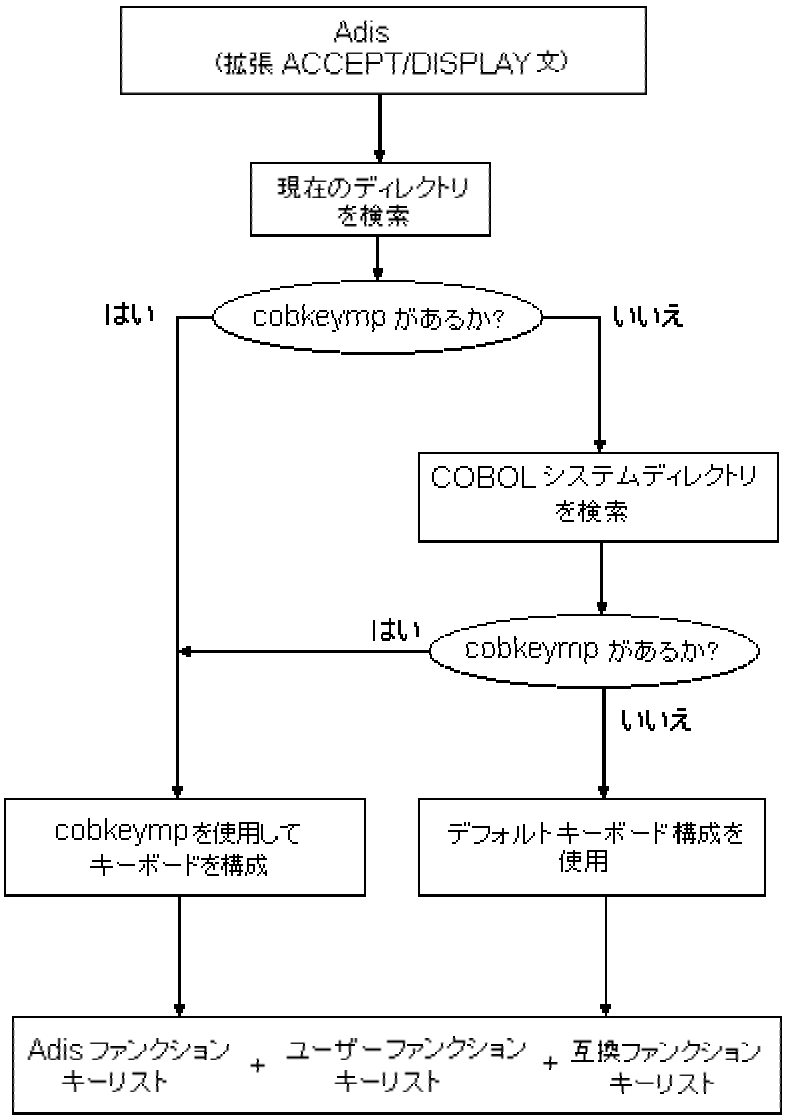
変更されたファンクションのリストを **cobkeymp** というファイルに保存します。

4.5 Keybcf の終了 (オプション 4)

Keybcf のメインメニューから **4** を押して、オペレーティングシステムのコマンド行へ戻ります。

4.6 実行時の検索機構

次の図は、実行時に使用される検索機構を示したもので、どのキーボード構成オプションを Adis で使用するかを決定します。



4.7 キーリスト

次のキーのリストは、Keybcf で見直し、変更できます。

- Adis ファンクションキーリスト

このリストは、COBOL プログラムで ACCEPT 文を実行するときに指定された機能を実行するキーを定義します。 キーの説明については、『*拡張 ACCEPT/ DISPLAY 構文*』の章にある『*ファンクションキー*』を参照してください。

- ユーザファンクションキーリスト

  ユーザファンクションキーリストは、ファンクションキーが押されたときにプログラムに返されるキーを定義します。 Adis とユーザファンクションキーのリストに同じキーを定義する場合は、『*拡張 ACCEPT/ DISPLAY 構文*』の章にある『*ユーザキーリストと Adis キーリストの両方でのキーの定義*』を参照してください。

- 互換ファンクションキーリスト
  - このリストは、COBOL の他の方言にキーの互換性をもたせるために使用されるユーザキーリストの代替を定義します。 キーから返される値が標準のユーザキーリストと異なる場合は、ユーザファンクションキーリストではなく、互換ファンクションキーリストを変更することをお奨めします。 デフォルトでは、互換ファンクションキーリストは、UNIX 上の RM/COBOL V2.0 への互換性のために設定されます。
  - Adiscf の ACCEPT/DISPLAY オプション 22 で、ユーザまたは互換キーリストを選択できます。

デフォルトでは、ユーザファンクションキーの初期設定は無効です。プログラム実行中にファンクションキーを使用するために、x"AF" ライブラリルーチンを呼び出すか、または、Adiscf を使用して構成を変更すると、ユーザファンクションキーを有効にできます。

ACCEPT 操作を終了させるために使用されるファンクションキーを確認するには、*CRT STATUS 句*を使用します。CRT STATUS 句の説明は、『言語リファレンス』を参照してください。

4.7.1 Adis ファンクションキーリストメニュー

次のキーが、Adis ファンクションキーリストメニューで使用できます。

キー	説明
Space	各機能に定義されているキーを変更しないで、次の機能に移ります。
Ins	表示されている機能を実行するために、別のキーを追加します。現在定義されているファンクションキーは保持されます。 Insert という語句が、画面の右下隅に表示されます。
Delete	定義されたファンクションキーをリストから削除します。 Keybcf は、自動的に次の機能に移ります。
HeX	ファンクションキーに対する 16 進数入力を指定します。 Hex という語句が、画面の右下隅に表示されます。 無効な 16 進数を入力すると、次の機能に移ろうとするときにエラーが通知され、有効な 16 進数の入力を求められます。 この機能は、使用しているキーボードに現れないキーを定義し、プログラム実行中

に使用できるようにするものです。

Quit　　変更メニューへ戻ります。

第 5 章 : Panels

Panels は、アプリケーションにウィンドウ機能を可能にするモジュールです。COBOL プログラムから呼び出せるルーチンのセットで構成されます。ウィンドウを重ねて表示したり、テキストと属性を分離または結合したりできます。また、ウィンドウ内でのテキストのスクロール、ポップアップメニューやプルダウンメニューの表示なども可能です。

Panels は、他のウィンドウで重なっている画面領域を自動的に維持します。これは、高性能な画面処理機能が必要なプログラマのためのものです。

Panels は、最大 65534 パネルを作成し、画面上に最大 254 パネルを表示できます (マシンに十分なメモリがある場合)。

5.1 パネルの概要

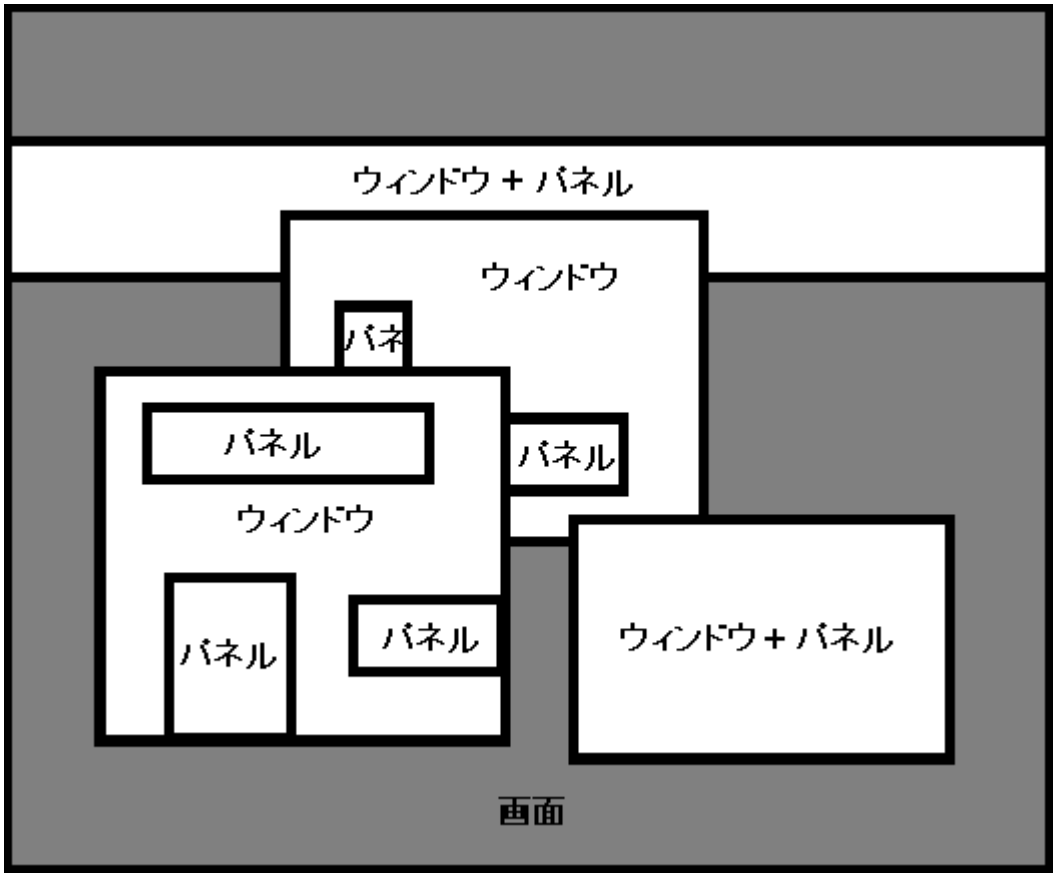
Panels の基本的な概念は、パネルです。パネルは、実際の画面のように書き込める仮想画面と考えることができます。ただし、パネルは、実際の画面よりも大きくしたり、小さくしたりできます。

パネルの内容は、ウィンドウを通じて表示できます。実際の画面に表示されるパネルに関連付けられたウィンドウです。ウィンドウはパネルの実際の大きさよりも小さく、パネルの内容の部分のみ表示できます。ウィンドウで表示できるパネルの部分は、プログラマが変更できます。任意の数のウィンドウが同時に画面上に表示でき、必要であれば、互いを重ねることができます。

特定のパネルに関連付けられたウィンドウは、画面上で表示できる (使用可能)か、または、表示できない (使用不能) のどちらかです。パネルが使用不能でもパネルへの書き込みができますが、そのパネルが使用可能にならない限り、表示できません。画面上のウィンドウの位置とサイズは、いつでも変更できます。

パネルのどの部分にでも、プログラムで書き込みや読み込みができます。また、長方形のパネル領域は、どの方向にもスクロールできます。

次の図は、画面、ウィンドウ、およびパネル間の関係を示しています。



5.2 Panels の使用方法

Panels は、次のように使用できます。

- CALL 文で Panels モジュールを呼び出します。

  CALL 文には、Panels で処理するときに定義する、フィールドをもつ panels-parameter-block を指定する必要があります。

  パネルに書き込んだり、パネルを読み込んだり、スクロールする場合には、CALL 文に text-buffer と attribute-buffer を指定する必要があります。

  パネルが作成されたり、ウィンドウが移動またはサイズが変更されたりすると、指定したサイズのウィンドウが作成できない場合は、ウィンドウのサイズが縮小される可能性があります。これを、ウィンドウの高さと幅のクリッピングと呼びます。

5.2.1 Panels の呼び出し

COBOL プログラムで次のような CALL 文を使用します。

```
call "PANELS" using   panels-parameter-block
                      [text-buffer]
                      [attribute-buffer]
```

パラメータは、次のとおりです。

panels-parameter-block	Panels で処理するときに定義したすべてのフィールドをもつレベル-01 項目。この集団項目に対する定義は、COBOL システムに提供されているコピーファイル **panlink.cpy** に定義されています。**panlink.cpy** を変更する必要はありませんが、Panels が実行する機能を指定するために、プログラムで **panlink.cpy** の特定のデータ項目にニーモニックを転記します。
text-buffer	パネルに含まれるテキスト、パネルに関連付けられた名前、または使用可能なパネルのリストを指定します。プログラムの作業場所節で定義される必要があります。
attribute-buffer	パネルに含まれる属性を指定します。プログラムの作業場所節で定義される必要があります。

すべての Panels の呼び出しに *panels-parameter-block* を指定する必要があります。パネルに書き込んだり、パネルを読み込んだり、スクロールする場合には、CALL 文に *text-buffer* と *attribute-buffer* を指定する必要があります。

その他の参照項目

『*Panels パラメータブロック*』および『*Panels での属性の指定*』

5.2.2 Panels のテキストと属性

Panels は、各テキスト文字に 1 つの属性符号化バイトを関連付けます。パネルが画面に表示されると、各テキスト文字は、関連する属性符号化バイトに従って属性を符号化して表示されます。

属性符号化には、次の 3 つの種類があります。

- 標準の PC モノクロ属性符号化。モノクロディスプレイシステム用の Panels におけるデフォルトの符号化です。
- 標準の PC カラー属性符号化。カラーディスプレイシステム用の Panels におけるデフォルトの符号化です。
- ユーザ選択、または、すべてのディスプレイ用の汎用的な属性符号化。

RTS ライブラリルーチンを呼び出して、Panels で使用される符号化の種別を決定できます。詳細については、『リリースノート』を参照してください。

ACCEPT と DISPLAY 文は、Adis を使用してターミナル I/O をサポートし、Panels で使用されると、Adis と Panels は同じ属性符号化を使用する必要があります。これは、自動的に行われます。

5.2.2.1 Panels でのテキストの指定

次の 3 つの方法の 1 つを使用します。

- プログラムの作業場所節で text-buffer を定義し、text-buffer をパラメータとして Panels を呼び出します。この方法は、ある領域を異なる文字で埋めるために使用されます。
- パラメータブロックの PPB-Fill-Character に対する値を指定します。この方法は、ある領域を同じ文字で埋めるために使用されます。
- ACCEPT と DISPLAY 文をパネル内でフォーマットされた画面 I/O に使用します。

5.2.2.2 Panels での属性の指定

次の 3 つの方法の 1 つを使用します。

- プログラムの作業場所節で attribute-buffer を定義し、attribute-buffer をパラメータとして Panels を呼び出します。この方法は、ある領域を異なる属性で埋めるために使用されます。
- パラメータブロックの PPB-Fill-Attribute に対する値を指定します。この方法は、ある領域を同じ属性で埋めるために使用されます。
- ACCEPT と DISPLAY 文をパネルに属性やカラーを指定するフォーマットされた画面 I/O に使用します。

5.2.2.3 テキスト指定なしで属性を指定

text-buffer を指定する場合は、常に CALL 文の 2 番目のパラメータになります。*attribute-buffer* は CALL 文の 3 番目のパラメータになります。そのため、テキスト指定なしで属性を指定したい場合は、CALL 文の 2 番目のパラメータにはダミーパラメータを設定する必要があります。ダミーパラメータは、任意の変数です。ダミーパラメータを指定する構文は、次のとおりです。

```
call "PANELS" using panels-parameter-block
                    dummy
                    attribute-buffer
```

5.3 Panels のパラメータブロック

次に示す Panels パラメータブロックは、Panels で特定の機能を実行するために初期設定されるフィールドで構成されます。Panels を呼び出す前に、機能に関連するすべてのフィール

ドを定義していることを確認してください。便宜上、パラメータブロックは Panels ソフトウェアとともに提供されるコピーファイル **panlink.cpy** にありますので、COPY 文を使用して、プログラムにインクルードします。

```
01 Panels-Parameter-Block.
    03 PPB-Function              PIC 9(2) COMP-X.
    03 PPB-Status                PIC 9(2) COMP-X.
    03 PPB-Panel-ID              PIC 9(4) COMP-X.
    03 PPB-Panel-Width           PIC 9(4) COMP-X.
    03 PPB-Panel-Height          PIC 9(4) COMP-X.
    03 PPB-Visible-Width         PIC 9(4) COMP-X.
    03 PPB-Visible-Height        PIC 9(4) COMP-X.
    03 PPB-First-Visible-Co      PIC 9(4) COMP-X.
    03 PPB-First-Visible-Row     PIC 9(4) COMP-X.
    03 PPB-Panel-Start-Column    PIC 9(4) COMP-X.
    03 PPB-Panel-Start-Row       PIC 9(4) COMP-X.
    03 PPB-Buffer-Offset         PIC 9(4) COMP-X.
    03 PPB-Vertical-Stride       PIC 9(4) COMP-X.
    03 PPB-Update-Group.
        05 PPB-Update-Coun       PIC 9(4) COMP-X.
        05 PPB-Rectangle-Offset  PIC 9(4) COMP-X.
        05 PPB-Update-Start-Col  PIC 9(4) COMP-X.
        05 PPB-Update-Start-Row  PIC 9(4) COMP-X.
        05 PPB-Update-Width      PIC 9(4) COMP-X.
        05 PPB-Update-Height     PIC 9(4) COMP-X.
    03 PPB-Fill.
        05 PPB-Fill-Character    PIC X.
        05 PPB-Fill-Attribute    PIC X.
    03 PPB-Update-Mask           PIC X.
    03 PPB-Scroll-Direction      PIC 9(2) COMP-X.
    03 PPB-Scroll-Count          PIC 9(4) COMP-X.
```

panlink.cpy のデータ項目は、『*Panels 機能用のパラメータ*』で説明されています。

特定の機能に関連するフィールドは、『*Panels の機能*』で説明されています。

5.3.1 Panels 機能用のパラメータ

Panels 機能に必要なパラメータは、コピーファイル **panlink.cpy** の Panels パラメータブロックに定義されています。

5.3.2 パラメータの説明

PPB-Buffer-Offset

PPB-Buffer-Offset に指定する値は、テキストバッファで検出される最初の文字位置、または、属性バッファの最初の属性を決定します。 値は、バッファ中の最初のバイトを示します。

PPB-Fill-Attribute

PPB-Fill-Attribute は、画面の背景文字の属性の定義、または、パネルの指定領域を満たす属性を決定するのに使用されます。背景属性のデフォルト (初期) 値は、IBM-PC では x"07" で、黒の背景で白のテキストです。

PPB-Fill-Character

PPB-Fill-Character は、画面の背景文字を定義したり、パネルの指定領域を単一文字で満たしたりするのに使用されます。 デフォルトの背景文字は、明示的に定義しなければ、空白文字です。

PPB-First-Visible-Col

PPB-First-Visible-Col に指定する値は、パネルの左端に対するウィンドウの相対位置を決定します。 PPB-First-Visible-Col を 0 に設定すると、ウィンドウの左端がパネルの左端に合わされます。

PPB-First-Visible-Row

PPB-First-Visible-Row に指定する値は、パネルの上部に対するウィンドウの相対位置を決定します。 PPB-First-Visible-Row を 0 に設定すると、ウィンドウの上部がパネルの上端に合わされます。

PPB-Function

Panels に実行させる機能は、呼び出しごとに指定する必要があります。 このフィールドに、機能の値、またはそのニーモニックを入力します。 各機能は、『*Panels の機能*』で説明されています。

PPB-Panel-Height

パネルの高さが、行数で定義されます。 PPB-Panel-Height でパネルの高さ (文字数) を指定します。 パネルの幅とパネルの高さを掛けた結果が、65535 文字を越えてはなりません。 65535 文字を越えると、エラーコード 6、PE-Panel-Too-Large を返します。

パネルを作成すると、その高さを変更できません。

PPB-Panel-ID

パネルを作成すると、Panels は PPB-Panel-ID で識別ハンドルを返します。作成された各パネルには、一意な識別ハンドルが与えられます。 特別なパネルを処理する場合には、PPB-Panel-ID は、パネルが作成されたときに返されるハンドルをもっています。 正しいハンドルの指定エラーは、Panels を使用する場合の共通な原因です。

Panels が特定のパネルに特定の識別子を割り当てるというわけではありません。 また、同じ識別子が同一プログラムが 2 つ実行される場合に使用されるというわけでもありません。 常に、パネルが作成されたときに、Panels から返される識別子を使用します。

PPB-Panel-Start-Column

PPB-Panel-Start-Column に指定する値は、画面上に表示するウィンドウの左端を決定します。 PPB-Panel-Start-Column を 0 に設定すると、画面上に表示するウィンドウは画面上の最初のカラムに合わされます。

PPB-Panel-Start-Row

PPB-Panel-Start-Row に指定する値は、画面上に表示するウィンドウの上部を決定します。 PPB-Panel-Start-Row を 0 に設定すると、画面上に表示するウィンドウは画面上の最初の行に合わされます。

PPB-Panel-Width

パネルの幅は、カラム数で定義されます。 PPB-Panel-Width でパネルの幅 (文字数) を指定します。 パネルの幅は、2000 文字を越えてはなりません。

パネルを作成すると、その幅を変更できません。

PPB-Rectangle-Offset

このフィールドは、ウィンドウの長方形内で更新しない文字数を指定します。PPB-Rectangle-Offset の値は、パネルの更新をする場合に、指定された長方形 (PPB-Update-Width と PPB-Update-Height で定義) 内の位置を指定します。領域全体を更新する場合は、PPB-Rectangle-Offset フィールドを 0 に設定します。

PPB-Update-Count と PPB-Rectangle-Offset フィールドで指定された四角形の不規則な形の領域を更新することもできます。

パネルの陰影でない部分を更新する場合は、PPB-Update-Count と PPB-Rectangle-Offset で更新する文字数と更新する位置を指定します。
1 行が 20 文字幅のパネルがあり、5 行目の 15 番目の文字から 406 文字分更新する場合には、次の手順を実行します。

1. 406 を PPB-Update-Count に転記すると、パネルの 406 文字 (5 行目の 6 文字と 6 行目から 20 行分) が更新されます。
2. 更新は、領域の 5 行目で、15 番目の文字から始まるので、領域の最初の 94 文字 (先頭から 4 行分と 5 行目の 14 文字) をスキップする必要があります。94 を PPB-Rectangle-Offset に転記して更新を始めます。

その他の参照項目

『*PPB-Update-Count*』、『*PPB-Update-Width*』、および『*PPB-Update-Height*』

PPB-Scroll-Count

PPB-Scroll-Count でスクロールする行数やカラム数を指定します。上や下へ (PPB-Scroll-Direction が 0 または 1) スクロールする場合は、PPB-Scroll-Count でスクロールする行を指定します。左や右へ (PPB-Scroll-Direction が 2 または 3) スクロールする場合は、PPB-Scroll-Count でスクロールするカラム数を指定します。

その他の参照項目

『*PPB-Scroll-Direction*』

PPB-Scroll-Direction.

PPB-Scroll-Direction で、パネルのテキストや属性がスクロールされる方向を指定します。スクロール方向の値は、次のとおりです。

値	方向
0	上
1	下
2	左
3	右

その他の参照項目

『*PPB-Scroll-Count*』

PPB-Status

Panels が呼び出すたびに、その実行結果の成功、不成功を示すために PPB-Status で状態値を返します。 機能の実行が成功の場合には、PPB-Status は 0 です。1 以上の値が返されると、Panels が機能の実行中にエラーが発生したことを示します。 状態値の完全なリストは、『Panel のエラーコード』に示されています。

Panels を呼び出すたびに、状態フィールドを検査することをお奨めします。

その他の参照項目

『*Panels のエラーコード*』

PPB-Scroll-Count

このフィールドは、指定された領域内で更新される文字数を指定します。PPB-Update-Count と PPB-Rectangle-Offset フィールドで指定された四角形の不規則な形の領域を更新することもできます。

領域全体を更新するには、このフィールドに、PPB-Update-Width と PPB-Update-Height を掛けた結果を設定します。

その他の参照項目

『*PPB-Rectangle-Offset*』、『*PPB-Update-Height*』、および『*PPB-Update-Width*』

PPB-Update-Height

このフィールドは、領域で更新されるサイズ (文字数) を定義します。このフィールドで、パネルの更新で影響を受ける行数を指定します。更新される領域の高さがパネルの高さより高いと、パネルの高さは縮小します。

その他の参照項目

『*PPB-Rectangle-Offset*』、『*PPB-Update-Count*』、および『*PPB-Update-Width*』

PPB-Update-Mask

PPB-Update-Mask フィールドは、次の場合に使用されます。

- 画面が自動的に更新されるかどうか。
- テキスト、属性、または両方が更新されるかどうか。
- 指定領域を埋めるデータの位置。

更新マスクは、1 バイトの値 (PIC X) です。次に示すように、このバイトの各ビットが PF-Flush-Panel、PF-Scroll-Panel、PF-Write-Panel、および PF-Read-Panel に定義されています。

ビット	設定時の意味
7	未使用。将来のために予約され、0 に設定する必要があります。
6	未使用。将来のために予約され、0 に設定する必要があります。
5	パネルの属性への更新は、直ちに画面に表示されます (パネルが使用可能な場合)。このビットが設定されていない場合には、更新は、次の PF-Flush-Panel 呼び出しが行われるまでは画面上に表示されません。
4	パネルのテキストへの更新は、直ちに画面に表示されます (パネルが使用可能な場合)。このビットが設定されていない場合には、更新は、次の PF-Flush-Panel 呼び出しが行われるまでは画面上に表示されません。
3	パネルの指定された領域の属性は、Panels-Parameter-Block の PPB-Fill-Attribute で指定された属性で置き換えられます。このビットは、ビット 1 と相互排他的であり、ビット 3 と 1 は予測できない結果になります。
2	パネルの指定された領域のテキストは、Panels-Parameter-Block の PPB-Fill-Character で指定された文字で置き換えら

	れます。このビットは、ビット 0 と相互排他的であり、ビット 2 と 0 は予測できない結果になります。
1	指定された領域の属性は、属性バッファで与えられた属性に置き換えられます。このバッファが使用される場合は、常に Panels の呼び出し時に 3 番目のパラメータで指定する必要があります。このビットは、ビット 3 と相互排他的であり、ビット 1 と 3 は予測できない結果になります。
0	指定された領域のテキストは、テキストバッファで与えられたテキストに置き換えられます。このバッファが使用される場合は、常に Panels の呼び出し時に 2 番目のパラメータで指定する必要があります。このビットは、ビット 2 と相互排他的であり、ビット 0 と 2 は予測できない結果になります。

注:

- ビット 7 が MSB または最初に置かれ、ビット 0 が LSB または最後に置かれます。
- PF-Read-Panel はビット 0 と 1 のみを使用します。

その他の参照項目

『*PF-Flush-Panel*』、『*PF-Scroll-Panel*』、『*PF-Write-Panel*』、『*PF-Read-Panel*』、『*PPB-Fill-Attribute*』、および『*PPB-Fill-Character*』

PPB-Update-Start-Col

このフィールドは、パネル内の領域の位置を定義します。このフィールドは、パネル更新時に影響を受ける領域の最初のカラムを定義します。 PPB-Update-Start-Col に 0 を指定すると、更新される最初のカラムは、パネルの最左端のカラムです。

PPB-Update-Start-Row

このフィールドは、パネル内の領域の位置を定義します。このフィールドは、パネル更新時に影響を受ける領域の最初の行を定義します。 PPB-Update-Start-Row に 0 を指定すると、更新される最初の行は、パネルの最上部の行です。

PPB-Update-Width

このフィールドは、領域で更新されるサイズ (文字数) を定義します。このフィールドで定義される数は、パネルの更新時に影響を受けるカラム数です。 更新する領域の幅がパネルの幅より広いと、パネルの幅は縮小されます。

その他の参照項目

『*PPB-Rectangle-Offset*』、『*PPB-Update-Count*』、および『*PPB-Update-Height*』

PPB-Vertical-Stride

PPB-Vertical-Stride に指定する値は、テキストや属性バッファ内の、ある行の始まりと次の行の始まりの間の間隔 (文字数) を決定します。 Panels は十分なデータが与えられると仮定しているので、バッファは、指定される領域を埋めるのに十分なデータを含むことを確認する必要があります。

PPB-Visible-Height

PPB-Visible-Height で表示するウィンドウの高さを行数で指定します。これは、パネルが使用可能な場合に、画面上で最初に表示する行数を設定します。 ウィンドウの高さは、パネルや画面の高さを越えることはできません。

PPB-Visible-Width

PPB-Visible-Width で表示するウィンドウの幅をカラム数で指定します。 これは、パネルが使用可能な場合に、画面上で最初に表示するカラム数を設定します。

ウィンドウの幅は、パネルや画面の幅を越えることはできません。

5.4 Panels のクリッピング

パネルが作成されたり、PF-Shift-Panel でウィンドウの移動やサイズが変更されたりすると、指定されたウィンドウのサイズが Panels で妥当性検査され、指定されたウィンドウサイズが作成できない場合は、サイズが縮小されます。 これを、ウィンドウの高さや幅のクリッピングと呼びます。

ウィンドウは、次の 2 つの理由から縮小されます。 次の例は、ウィンドウの幅が影響を受ける場合を示しています。ウィンドウの高さも同様に影響を受けますが、PPB-Panel-Start-Column と PPB-First-Visible-Col ではなく、PPB-Panel-Start-Row と PPB-First-Visible-Row の値に依存します。

1. PPB-Panel-Start-Column が大きすぎる場合。開始カラムの値が、指定されたサイズのウィンドウが画面の右端を越えると、ウィンドウの幅は画面に合うように縮小されます。

   たとえば、80 文字幅の画面で、ウィンドウが 40 文字幅 (PPB-Visible-Width = 40) と定義されているとします。ここで、ウィンドウが画面の 60 文字目 (PPB-Panel-Start-Column = 60) から始まると、そのウィンドウ全体を画面上に合わせるには、画面は 100 文字幅をもつ必要があります。画面は 80 文字幅なので、PPB-Visible-Width は 20 文字分クリップされます (PPB-Visible-Width は 20 縮小)。

2. PPB-First-Visible-Col が大きすぎる場合。開始カラムの値が、指定されたサイズのウィンドウがパネルの右端を越えると、ウィンドウの幅はパネルに合うように縮小されます。
3. たとえば、パネルが 40 (PPB-Panel-Width = 40) の幅で作成され、30 (PPB-Visible-Width = 30) のウィンドウ幅で、パネルの開始位置が 15 (PPB-First-Visible-Col = 15) と指定されると、Panels は自動的にウィンドウ幅を 25 に縮小します (PPB-Panel-Width は 25 縮小)。

ウィンドウの幅や高さは、ディスプレイからウィンドウが見えなくなると、0 に縮小されます。ただし、(PF-Disable-Panel 呼び出しによる結果とは異なり) 画面上のパネルのスタックの順序は保持されます。

サンプルプログラム **panelex1.cbl** は、この機能を説明したものです。デモンストレーションパネルが移動されると、プログラムは PF-Shift-Panel を使用して、ウィンドウの高さと幅をパネルの高さと幅と同じに設定します。

パネルは、PPB-Panel-Start-Column の値を増加させると右に移動します。画面の右端に達すると、幅のクリップが開始されます。そのため、パネルが画面の右端が消え去るように表示されます。

パネルは、PPB-Panel-Start-Column の値を減少させると左に移動します。この値が 0 になると、画面の左端に達します。画面の左端からパネルが消え去るような効果を得るために、PPB-First-Visible-Col の値が増加されます。その結果、ウィンドウの幅は、Panels で減少させられ、画面の端から消え去るように表示されます。

PF-Shift-Panel を呼び出す前に PPB-Visible-Width を常に最大サイズにすることで、**panelex1.cbl** はウィンドウのどれくらいの部分が実際に表示できるかを Panels に計算させます。

5.5 Panels の機能

Panels を使用すると、多くの機能を実行できます。機能は、次のとおりです。

PF-Create-Panel	パネルを作成します。
PF-Create-Panel-	陰影効果をもったパネルを作成します。

With-Shadow	
PF-Delete-Panel	パネルをシステムから削除します。
PF-Disable-Panel	パネルを表示しないように、使用不能にします。
PF-Enable-Panel	パネルを画面上に表示するように、使用可能にします。
PF-Flush-Panel	指定されたパネルへの変更が反映されていない画面を更新します。
PF-Get-First-Panel	最後に使用可能とされたパネルに関する識別ハンドルと情報を返します。
PF-Get-Next-Panel	最前面のパネルが削除された場合に、次に最前面となるパネルの識別ハンドルと情報を返します。
PF-Get-Panel-At-Position	特定の位置にパネルがあるかどうかを決定します。
PF-Get-Panel-Info	既存のパネルのサイズと位置を見つけます。
PF-Get-Panel-Stack	すべての使用可能なパネルの識別ハンドルのリストを返します。
PF-Get-Screen-Info	現在の画面のサイズに関する情報を返します。
PF-Read-Panel	パネルからテキストと属性を読み込み、呼び出し側プログラムで指定されているバッファに格納します。
PF-Redraw-Screen	画面全体の内容を最新表示します。
PF-Scroll-Panel	パネル領域をスクロールします。
PF-Set-Panel-Name	指定したパネルに名前を関連付けます。
PF-Set-Panel-Stack	使用可能なパネルのスタックをリセットし、画面を再描画します。
PF-Set-Screen-Backdrop	現在の背景文字と属性を変更します。
PF-Shift-Panel	ウィンドウを移動したり、そのサイズを変更したりします。
PF-Write-Panel	パネルへ書き込みます。

PF-Create-Panel (値 3)

パネルを作成します。 パネルのウィンドウは、使用不能です。つまり、表示されません。

パラメータ

PPB-First-Visible-Col	pic 9(4) comp-x.
PPB-First-Visible-Row	pic 9(4) comp-x.
PPB-Panel-Height	pic 9(4) comp-x.
PPB-Panel-ID	pic 9(4) comp.
PPB-Panel-Start-Column	pic 9(4) comp-x.

PPB-Panel-Start-Row	pic 9(4) comp-x.
PPB-Panel-Width	pic 9(4) comp-x.
PPB-Visible-Height	pic 9(4) comp-x.
PPB-Visible-Width	pic 9(4) comp-x.

起動時の設定

PPB-First-Visible-Col	パネル内でウィンドウを水平方向に位置付けます。値 0 は、ウィンドウの左端がパネルの左端に位置していることを示します。
PPB-First-Visible-Row	パネル内でウィンドウを垂直方向に位置付けます。値 0 は、ウィンドウの上端がパネルの上端に位置していることを示します。
PPB-Panel-Height	作成されるパネルのテキストの行数。
PPB-Panel-Start-Column	画面上でパネルを水平方向に位置付けます。値 0 は、パネルの左端が画面の左端に位置していることを示します。
PPB-Panel-Start-Row	画面上でパネルを垂直方向に位置付けます。値 0 は、パネルの上端が画面の上端に位置していることを示します。
PPB-Panel-Width	作成されるパネルのテキストのカラム数。
PPB-Visible-Height	表示するウィンドウのテキストの行数。
PPB-Visible-Width	表示するウィンドウのテキストのカラム数。

終了時の設定

PPB-Panel-ID	作成されたパネルの識別ハンドル。プログラムの作業場所節内のデータ項目にこの値をコピーし、このパネルに対する機能を使用する場合に指定します。

備考

Panels は、そのサイズに基づいて、新しいパネルにメモリを割り当てます。パネルに使用されるメモリのバイト数は、次のように決定されます。

```
2 × PPB-Panel-Height × PPB-Panel-Width
```

新しいパネルのサイズは、画面よりも大きくできますが、次のような制限があります。

- PPB-Panel-Width を 2000 以下にする。

- PPB-Panel-Height × PPB-Panel-Width を 65535 以下にする。

これらの制限のどちらかに違反すると、エラーコード 6 が返されます。

パネルを作成すると、PF-Shift-Panel を使用して、幅と高さを除く任意の変数を変更できます。新しく作成されたパネルは、現在の背景文字と属性で埋められます。

その他の参照項目

『*PPB-Panel-Width*』、『*PPB-Panel-Height*』、『*PF-Shift-Panel*』、および『*Panels のエラーコード*』

例

次の例は、20 文字の幅で 15 行の高さのパネルを作成します。このパネルは、画面上の 4 行目、1 カラムから始まります。このパネルのウィンドウは、パネルの最初の行、最初のカラムから始まります。このウィンドウは、20 文字幅で 10 行の高さです。

```
* パネルのサイズは、20 文字幅で 15 行です。
       move 20 to ppb-panel-width.
       move 15 to ppb-panel-height.
* ウィンドウは画面上で 4 行、1 カラムから始まります。
* PPB-Panel-Start-Column と PPB-Panel-Start-Row で
* 画面位置を指定します。
* ここで、0,0 は画面の左上隅です。
       move 3 to ppb-panel-start-row.
       move 0 to ppb-panel-start-column.
* パネルのウィンドウは、20 文字幅で 10 行です。
       move 20 to ppb-visible-width.
       move 10 to ppb-visible-height.
* ウィンドウは、パネルの最初の行、最初のカラムから始まります。
* 0,0 は、パネルの左上隅です。
       move 0 to ppb-first-visible-row.
       move 0 to ppb-first-visible-col.
* パネルを作成します。 使用不能 (表示しない) に初期設定されます。
       move pf-create-panel to ppb-function.
       call "PANELS" using panels-parameter-block.
       if ppb-status not = zero
   *         (中止処理)
             ...
* パネルの識別子を保存します。
       move ppb-panel-id to ws-save-panel-id.
```

PF-Create-Panel-With-Shadow (値 19)

陰影効果をもったパネルを作成します。パネルのウィンドウは、使用不能です。つまり、表示されません。

パラメータ

PPB-Fill-Attribute	pic x.
PPB-Fill-Character	pic x.
PPB-First-Visible-Col	pic 9(4) comp-x.
PPB-First-Visible-Row	pic 9(4) comp-x.
PPB-Panel-Height	pic 9(4) comp-x.
PPB-Panel-ID	pic 9(4) comp
PPB-Panel-Start-Column	pic 9(4) comp-x.
PPB-Panel-Start-Row	pic 9(4) comp-x.
PPB-Panel-Width	pic 9(4) comp-x.
PPB-Update-Mask	pic x.
PPB-Visible-Height	pic 9(4) comp-x.
PPB-Visible-Width	pic 9(4) comp-x.

起動時の設定

PPB-Fill-Attribute	陰影に使用される属性。
PPB-Fill-Character	陰影に使用される文字。
PPB-First-Visible-Col	パネル内でウィンドウを水平方向に位置付けます。値 0 は、ウィンドウの左端がパネルの左端に位置していることを示します。
PPB-First-Visible-Row	パネル内でウィンドウを垂直方向に位置付けます。値 0 は、ウィンドウの上端がパネルの上端に位置していることを示します。
PPB-Panel-Height	作成されるパネルのテキストの行数。
PPB-Panel-Start-Column	画面上でパネルを水平方向に位置付けます。値 0 は、パネルの左端が画面の左端に位置していることを示します。
PPB-Panel-Start-Row	画面上でパネルを垂直方向に位置付けます。値 0 は、パネルの上端が画面の上端に位置していることを示します。
PPB-Panel-Width	作成されるパネルのテキストのカラム数。
PPB-Update-Mask	陰影の幅。0 は 1 文字幅を示し、それ以外は 2 文字幅を示します。

PPB-Visible-Height　　表示するウィンドウのテキストの行数。

PPB-Visible-Width　　表示するウィンドウのテキストのカラム数。

終了時の設定

PPB-Panel-ID　　作成されたパネルの識別ハンドル。プログラムの作業場所節内のデータ項目にこの値をコピーし、このパネルに対する機能を使用する場合に指定します。

備考

Panels は、そのサイズに基づいて、新しいパネルにメモリを割り当てます。陰影をもつパネルに使用されるメモリのバイト数は、次のように決定されます。

```
4 × PPB-Panel-Height × PPB-Panel-Width
```

新しいパネルのサイズは、画面よりも大きくできますが、次のような制限があります。

- PPB-Panel-Width を 2000 以下にする。
- PPB-Panel-Height × PPB-Panel-Width を 65535 以下にする。

これらの制限のどちらかに違反すると、エラーコード 6 が返されます。

陰影をもつパネルを作成すると、幅、高さ、および陰影の詳細を除く任意の変数を変更できます。PF-Shift-Panel を使用します。新しく作成されたパネルは、現在の背景文字と属性で埋められます。陰影をもつパネルでの陰影効果は透過ではありません。

その他の参照項目

『*PF-Create-Panel*』、『*PF-Shift-Panel*』、および『*Panels のエラーコード*』

例

次の例は、25 文字の幅で 8 行の高さのパネルを作成します。このパネルは、画面上の 1 行目、1 カラムから始まります。このパネルのウィンドウは、パネルの最初の行、最初のカラムから始まります。このウィンドウは、25 文字幅で 6 行の高さです。

パネルの陰影は、1 文字幅で、グレーの帯状に見えます。

```
* パネルのサイズは、25 文字幅で 8 行です。
       move 25 to ppb-panel-width.
       move  8 to ppb-panel-height.
```

```
* ウィンドウは画面上で 1 行、1 カラムから始まります。
* PPB-Panel-Start-Column と PPB-Panel-Start-Row で
* 画面位置を指定します。
* ここで、0,0 は画面の左上隅です。
      move 0 to ppb-panel-start-row.
      move 0 to ppb-panel-start-column.
* パネルのウィンドウは、25 文字幅で 6 行です。
      move 25 to ppb-visible-width.
      move  6 to ppb-visible-height.
* ウィンドウは、パネルの最初の行、最初のカラムから始まります。
* 0,0 は、パネルの左上隅です。
      move 0 to ppb-first-visible-row.
      move 0 to ppb-first-visible-col.
* パネルの陰影を指定します。
      move 0 to ppb-update-mask
      move space to ppb-fill-character
      move gray  to ppb-fill-attribute
* パネルを作成します。 使用不能（表示しない）に初期設定されます。
      move pf-create-panel to ppb-function.
      call "PANELS" using panels-parameter-block.
      if ppb-status not = zero
   *         (中止処理)
             ...
* パネルの識別子を保存します。
      move ppb-panel-id to ws-save-panel-id.
```

PF-Delete-Panel (値 6)

パネルをシステムから削除します。

パラメータ

PPB-Panel-ID　pic 9(4) comp-x.

起動時の設定

PPB-Panel-ID　削除されるパネルのハンドル。

終了時の設定

なし

備考

パネルを削除すると、パネル識別子とパネルに割り当てられていた記憶領域が再利用できるようになります。削除されたパネルへアクセスすると、エラーコード 1 が返され、アクセスが不成功であったことが示されます。

使用可能であるパネルを削除すると、画面から見えなくなります。

その他の参照項目

『*Panels のエラーコード*』

例

この例は、変数 ws-save-panel-id に保存された識別子のパネルを削除します。

```
       move ws-save-panel-id to ppb-panel-id.
       move pf-delete-panel to ppb-function.
       call "PANELS" using panels-parameter-block.
       if ppb-status not = zero
   *         (中止処理)
```

PF-Disable-Panel (値 8)

パネルを表示しないように、使用不能にします。

パラメータ

PPB-Panel-ID　　pic 9(4) comp-x.

起動時の設定

PPB-Panel-ID　　パネルを使用不能にする識別ハンドル。

終了時の設定

なし

備考

使用不能のパネルに重なっていたウィンドウが現れます。画面上に表示されていない場合でも、使用不能になったパネルを扱うことはできます。使用不能のパネルへの変更結果を確認するには、単に再度使用可能にするのみです。

その他の参照項目

『*PF-Create-Panel*』および『*PF-Enable-Panel*』

例

この例は、変数 ws-save-panel-id に保存された識別子のパネルを使用不能にします。

```
       move ws-save-panel-id to ppb-panel-id.
       move pf-disable-panel to ppb-function.
       call "PANELS" using panels-parameter-block.
       if ppb-status not = zero
   *         (中止処理)
```

PF-Enable-Panel (値 7)

パネルを画面上に表示するように、使用可能にします。

パラメータ

PPB-Panel-ID　　pic 9(4) comp-x.

起動時の設定

PPB-Panel-ID　　パネルを使用可能にする識別ハンドル。

終了時の設定

なし

備考

パネルを使用可能にする前に、パネルの作成、パネルの移動、またはパネルへの書き込みができます。ただし、これらの動作の結果を確認するには、パネルを使用可能にする必要があります。

パネルを使用可能にすると、実際は、そのパネルに関連付けられたウィンドウが表示されます。ウィンドウは、パネルを変更や移動しなかった場合は、パネルを作成したときに定義された位置に表示されます。

使用可能なパネルが以前に作成されたパネルと同じ画面領域を共有すると、新しく使用可能になったパネルは以前に作成されたパネルに重なります。

プログラムで複数のパネルがあると、使用可能にされた順序でスタックされます。最後に使用可能にされたパネルが、画面上の他のパネルの上に表示されます。

パネルが部分的または完全に他のパネルと重なっていても、パネルの移動、パネルの変更、パネルからの読み込み、またはパネルへの書き込みができます。

その他の参照項目

『*PF-Create-Panel*』

例

この例は、変数 ws-save-panel-id に保存された識別子のパネルを使用可能にします。

```
       move ws-save-panel-id to ppb-panel-id.
       move  pf-enable-panel to ppb-function.
       call "PANELS" using panels-parameter-block.
       if ppb-status not = zero
   *         (中止処理)
```

PF-Flush-Panel (値 9)

以前の呼び出し中に PPB-Update-Mask のビット 4 と 5 がオフにされたために、指定されたパネルへの変更が反映されていない画面を更新します。

パラメータ

PPB-Panel-ID	pic 9(4) comp-x.
PPB-Update-Height	pic 9(4) comp-x.
PPB-Update-Mask	pic x.
PPB-Update-Start-Col	pic 9(4) comp-x.
PPB-Update-Start-Row	pic 9(4) comp-x.
PPB-Update-Width	pic 9(4) comp-x.

起動時の設定

PPB-Panel-ID	フラッシュするパネルの識別ハンドル。
PPB-Update-Height	更新する領域の高さ。
PPB-Update-Mask	『*Panels のパラメータブロック*』を参照してください。
PPB-Update-Start-Col	更新される領域の最初のカラム。
PPB-Update-Start-Row	更新される領域の最初の行。
PPB-Update-Width	更新する領域の幅。

終了時の設定

なし

備考

PF-Flush-Panel は、パネルが使用可能でなければ、影響がありません。

その他の参照項目

『*パネルの概要*』、『*Panels のパラメータブロック*』、および『*PPB-Update-Mask*』

例

この例は、30 文字の幅で 17 行の高さのパネルを仮定しています。元のパネルの画面表示は、2 行目、1 カラム目から始まり、16 行で更新されます。元のパネルのハンドルは、データ項目 ws-save-panel-id に保存されています。

```
* フラッシュする領域のサイズを定義します。
       move 30 to ppb-update-width.
       move 15 to ppb-update-height.
* パネルの領域が始まる位置を定義します
* (0,0 はパネルの左上隅)。 更新パネルは
* パネルの 2 行目、1 カラムから始まります。
       move 1 to ppb-update-start-row.
       move 0 to ppb-update-start-col.
* テキストと属性の更新を適用するために、更新マスクの
* ビット 4 と 5 を設定します。 16 進数 30 は、2 進表示で 00110000 です。
       move X"30" to ppb-update-mask.
* パネルが作成されると、変数 ws-save-panel-id に
* パネルの識別子が保存されています。
       move ws-save-panel-id to ppb-panel-id.
   * 更新を適用します。
       move pf-flush-panel to ppb-function.
       call "PANELS" using panels-parameter-block.
       if ppb-status not = zero
   *         (中止処理)
```

PF-Get-First-Panel (値 15)

最後に使用可能とされたパネルに関する識別ハンドルと情報を返します。

パラメータ

Panel-Name-Buffer.	
Panel-Name-Length	pic 99 comp-x.
Panel-Name-Text	pic x(30).
PPB-First-Visible-Col	pic 9(4) comp-x.
PPB-First-Visible-Row	pic 9(4) comp-x.
PPB-Panel-Height	pic 9(4) comp-x.
PPB-Panel-ID	pic 9(4) comp-x.
PPB-Panel-Start-Column	pic 9(4) comp-x.
PPB-Panel-Start-Row	pic 9(4) comp-x.
PPB-Panel-Width	pic 9(4) comp-x.
PPB-Visible-Height	pic 9(4) comp-x.
PPB-Visible-Width	pic 9(4) comp-x.

起動時の設定

なし

終了時の設定

Panel-Name-Length	パネルに関連付けられた名前の長さ。
Panel-Name-Text	PF-Set-Panel-Name 呼び出しでパネルに関連付けられた名前。
PPB-First-Visible-Col	パネル上に表示するウィンドウの水平位置。
PPB-First-Visible-Row	パネル上に表示するウィンドウの垂直位置。
PPB-Panel-Height	パネルの高さ。
PPB-Panel-ID	最後に使用可能とされたパネルの識別ハンドル。
PPB-Panel-Start-Column	画面上のパネルの水平位置。
PPB-Panel-Start-Row	画面上のパネルの垂直位置。
PPB-Panel-Width	パネルの幅。
PPB-Visible-Height	パネルに表示するウィンドウの高さ。
PPB-Visible-Width	パネルに表示するウィンドウの幅。

備考

この機能の終了時、Panel-Name-Length の MSB は、パネルが現在使用可能になっているかを示すフラグです。0 はパネルが使用不能で、0 以外はパネルが使用可能です。これは、Panel-Name-Length > 127 で確認します。

PF-Get-First-Panel を呼び出したときにパネルがなければ、PPB-Status は、値 Error-Invalid-ID を含みます。

その他の参照項目

『*PF-Set-Panel-Name*』および『*PF-Get-Next-Panel*』

例

この例は、前述のように、プログラムの作業場所節に Panel-Name-Buffer を定義していると仮定しています。

```
       move pf-set-panel-name to ppb-function.
       call "PANELS" using panels-parameter-block
       panel-name-buffer.
*
* パネルが使用可能かどうかを確認するコードです。
*       if ppb-status not = zero
*         (中止処理)
```

PF-Get-Next-Panel (値 16)

最前面のパネルが削除された場合に、次に最前面となるパネルの識別ハンドルと情報を返します。

パラメータ

Panel-Name-Buffer.	
Panel-Name-Length	pic 99 comp-x.
Panel-Name-Text	pic x(30).
PPB-First-Visible-Col	pic 9(4) comp-x.
PPB-First-Visible-Row	pic 9(4) comp-x.
PPB-Panel-Height	pic 9(4) comp-x.
PPB-Panel-ID	pic 9(4) comp-x.
PPB-Panel-Start-Column	pic 9(4) comp-x.
PPB-Panel-Start-Row	pic 9(4) comp-x.
PPB-Panel-Width	pic 9(4) comp-x.

PPB-Visible-Height	pic 9(4) comp-x.
PPB-Visible-Width	pic 9(4) comp-x.

起動時の設定

なし

終了時の設定

Panel-Name-Length	パネルに関連付けられた名前の長さ。
Panel-Name-Text	PF-Set-Panel-Name 呼び出しでパネルに関連付けられた名前。
PPB-First-Visible-Col	パネル上に表示するウィンドウの水平位置。
PPB-First-Visible-Row	パネル上に表示するウィンドウの垂直位置。
PPB-Panel-Height	パネルの高さ。
PPB-Panel-ID	最後に使用可能とされたパネルの識別ハンドル。
PPB-Panel-Start-Column	画面上のパネルの水平位置。
PPB-Panel-Start-Row	画面上のパネルの垂直位置。
PPB-Panel-Width	パネルの幅。
PPB-Visible-Height	パネルに表示するウィンドウの高さ。
PPB-Visible-Width	パネルに表示するウィンドウの幅。

備考

この機能の終了時、Panel-Name-Length の MSB は、パネルが現在使用可能になっているかを示すフラグです。0 はパネルが使用不能で、0 以外はパネルが使用可能です。これは、Panel-Name-Length > 127 で確認します。

PF-Get-Next-Panel の直前に使用できる Panels の呼び出しは、PF-Get-First-Panel と他の PF-Get-Next-Panel のみです。つまり、PF-Get-Next-Panel の 2 つの連続した呼び出しの間、または、PF-Get-First-Panel と PF-Get-Next-Panel の間に、他の Panels 呼び出しがないということです。

PF-Get-Next-Panel を呼び出したときに次のパネルがなければ、PPB-Status は、値 Error-Invalid-ID を含みます。

その他の参照項目

『*PF-Set-Panel-Name*』および『*PF-Get-First-Panel*』

例

この例は、Panels が把握しているすべてのパネルの詳細を返します。情報を保持するためにプログラムの作業場所節に表、および、添字として使用するデータ項目 panel-num を宣言していると仮定しています。

```
*
* 最初のパネルの詳細を取得します。
*
       move 1 to panel-num
       move pf-get-first-panel to ppb-function
       perform make-panels-call
       if ppb-status not = zero
*         (中止処理)
       perform move-values-to-working-storage
       move 1 to panel-num
*
* ppb-status が zero として返されるまで、
* 他のすべてのパネルの詳細を取得します。
*
       move pf-get-next-panel to ppb-function
       call "PANELS" using panels-parameter-block
       panel-name-buffer
       perform until ppb-status = zero
       perform move-values-to-working-storage
       add 1 to panel-num
       perform make-panels-call
       end-perform
         ...
       move-values-to-working-storage section.
*
* 終了時のすべてのパラメータの値を保持するために、
* 作業場所節へ転記します。
*
       move ppb-panel-id       to ws-panel-id (panel-num)
       move ppb-panel-height   to ws-panel-height (panel-num)
       move ppb-panel-width    to ws-panel-width (panel-num)
       move ppb-visible-height
                       to ws-visible-height (panel-num)
       move ppb-visible-width
                       to ws-visible-width (panel-num)
       move ppb-panel-start-column
                       to ws-panel-start-column (panel-num)
```

```
move ppb-panel-start-row
                to ws-panel-start-row (panel-num)
move ppb-first-visible-col
                to ws-first-visible-col (panel-num)
move ppb-first-visible-row
                to ws-first-visible-row (panel-mum)
move panel-name-buffer
                to ws-panel-name-buffer (panel-num)
move panel-name-length
                to ws-panel-name-length (panel-num)
move panel-name-text
                to ws-panel-name-text (panel-num)
  .
make-panels-call section.
call "PANELS" using panels-parameter-block
                    panel-name-buffer
```

PF-Get-Panel-At-Position (値 13)

特定の位置にパネルがあるかどうかを決定します。 パネルがあると、パネルの外観の詳細が返されます。

パラメータ

PPB-First-Visible-Col	pic 9(4) comp-x.
PPB-First-Visible-Row	pic 9(4) comp-x.
PPB-Panel-Height	pic 9(4) comp-x.
PPB-Panel-ID	pic 9(4) comp-x.
PPB-Panel-Start-Column	pic 9(4) comp-x.
PPB-Panel-Start-Row	pic 9(4) comp-x.
PPB-Panel-Width	pic 9(4) comp-x.
PPB-Visible-Height	pic 9(4) comp-x.
PPB-Visible-Width	pic 9(4) comp-x.

起動時の設定

PPB-Panel-Start-Column	パネルの存在を確認する x-座標。
PPB-Panel-Start-Row	パネルの存在を確認する y-座標。

終了時の設定

PPB-First-Visible-Col	表示するウィンドウのパネルへの相対 x-座標。パネルの最初のカラムは 0。
PPB-First-Visible-Row	表示するウィンドウのパネルへの相対 y-座標。パネルの最初の行は 0。
PPB-Panel-Height	指定された位置で見つかったパネルの高さ。
PPB-Panel-ID	指定された位置で見つかったパネルの識別ハンドル。使用可能なパネルがない場合は、値 0 が返されます。指定された場所に複数のパネルがあると、最後に使用可能となったパネルのハンドルが返されます。
PPB-Panel-Start-Column	画面上でのパネルの左端の x-座標。画面の最初のカラムは 0。
PPB-Panel-Start-Row	画面上でのパネルの最初の y-座標。パネルの最初の行は 0。
PPB-Panel-Width	指定された位置で見つかったパネルの幅。
PPB-Visible-Height	パネル上に表示するウィンドウの高さ。
PPB-Visible-Width	パネル上に表示するウィンドウの幅。

備考

パネルのハンドルが判明していて、その外観に関する情報を返したい場合は、PF-Get-Panel-Info を使用します。

その他の参照項目

『*PF-Get-Panel-Info*』

例

この例は、4 行目、6 カラムにあるパネルを識別します (ここで、0,0 は画面の左上隅です)。

```
       move 3 to ppb-panel-start-row.
       move 5 to ppb-panel-start-column.
       move pf-get-panel-at-position to ppb-function.
       call "PANELS" using panels-parameter-block.
       if ppb-status not = zero
      *        (中止処理)
```

PF-Get-Panel-Info (値 5)

既存のパネルのサイズと位置を見つけます。

パラメータ

PPB-First-Visible-Col	pic 9(4) comp-x.
PPB-First-Visible-Row	pic 9(4) comp-x.
PPB-Panel-Height	pic 9(4) comp-x.
PPB-Panel-ID	pic 9(4) comp-x.
PPB-Panel-Start-Column	pic 9(4) comp-x.
PPB-Panel-Start-Row	pic 9(4) comp-x.
PPB-Panel-Width	pic 9(4) comp-x.
PPB-Visible-Height	pic 9(4) comp-x.
PPB-Visible-Width	pic 9(4) comp-x.

起動時の設定

PPB-Panel-ID	情報を返したいパネルのハンドル。この値は、PF-Create-Panel 呼び出しで返されます。

終了時の設定

PPB-First-Visible-Col	指定されたパネル上に表示するウィンドウの水平位置。
PPB-First-Visible-Row	指定されたパネル上に表示するウィンドウの垂直位置。
PPB-Panel-Height	指定されたパネルの高さ。
PPB-Panel-Start-Column	画面上の指定されたパネルの水平位置。
PPB-Panel-Start-Row	画面上の指定されたパネルの垂直位置。
PPB-Panel-Width	指定されたパネルの幅。
PPB-Visible-Height	指定されたパネルに表示するウィンドウの高さ。
PPB-Visible-Width	指定されたパネルに表示するウィンドウの幅。

その他の参照項目

『*PF-Create-Panel*』

例

この例は、変数 ws-save-panel-id に保存された識別子のパネル情報を返します。

```
       move ws-save-panel-id to ppb-panel-id.
       move pf-get-panel-info to ppb-function.
       call "PANELS" using panels-parameter-block.
       if ppb-status not = zero
   *         (中止処理)
```

PF-Get-Panel-Stack (値 17)

すべての使用可能なパネルの識別ハンドルのリストを返します。

パラメータ

Panels-Order-List	集団項目には、次のものが含まれます。
POL-Count	pic 9(4) comp-x.
POL-ID	pic 9(4) comp-x occurs 254.

起動時の設定

なし

終了時の設定

POL-Count	リスト中の使用可能なパネル数。
POL-ID	識別ハンドルのリスト。

備考

リスト (POL-ID(1)) に返された最初の項目は、最後に使用可能となったパネルを保持しています。

その他の参照項目

『*PF-Set-Panel-Stack*』

例

この例は、すべての使用可能なパネルの識別ハンドルのリストを返します。 前述のように、プログラムの作業場所節に Panels-Order-List を宣言していると仮定しています。

```
move pf-get-panel-stack to ppb-function.
call "PANELS" using panels-parameter-block.
panels-order-list.
```

PF-Get-Screen-Info (値 0)

現在の画面のサイズに関する情報を返します。

パラメータ

PPB-Fill-Attribute	pic 9(4) comp-x.
PPB-Fill-Character	pic 9(4) comp-x.
PPB-First-Visible-Col	pic 9(4) comp-x.
PPB-First-Visible-Row	pic 9(4) comp-x.
PPB-Panel-Height	pic 9(4) comp-x.
PPB-Panel-Start-Column	pic 9(4) comp-x.
PPB-Panel-Start-Row	pic 9(4) comp-x.
PPB-Panel-Width	pic 9(4) comp-x.
PPB-Visible-Height	pic 9(4) comp-x.
PPB-Visible-Width	pic 9(4) comp-x.

起動時の設定

なし

終了時の設定

PPB-Fill-Attribute	現在の背景属性。
PPB-Fill-Character	現在の背景文字。
PPB-First-Visible-Col	0
PPB-First-Visible-Row	0
PPB-Panel-Height	画面上に表示する行数。
PPB-Panel-Start-Column	0
PPB-Panel-Start-Row	0
PPB-Panel-Width	画面上のカラム数。
PPB-Visible-Height	画面上に表示する行数。
PPB-Visible-Width	画面上のカラム数。

備考

PF-Set-Screen-Backdrop を使用して、背景文字と属性を変更できます。

その他の参照項目

『*PF-Set-Screen-Backdrop*』

例

```
       move pf-get-screen-info to ppb-function.
       call "PANELS" using panels-parameter-block.
       if ppb-status not = zero
*         (中止処理)
```

PF-Read-Panel (値 12)

パネルからテキストと属性を読み込み、呼び出し側プログラムで指定されているバッファに格納します。

パラメータ

attribute-buffer	pic x(n).
PPB-Buffer-Offset	pic 9(4) comp-x.
PPB-Panel-ID	pic 9(4) comp-x.
PPB-Rectangle-Offset	pic 9(4) comp-x.
PPB-Update-Count	pic 9(4) comp-x.
PPB-Update-Height	pic 9(4) comp-x.
PPB-Update-Mask	pic x.
PPB-Update-Start-Col	pic 9(4) comp-x.
PPB-Update-Start-Row	pic 9(4) comp-x.
PPB-Update-Width	pic 9(4) comp-x.
PPB-Vertical-Stride	pic 9(4) comp-x.
text-buffer	pic x(n).

起動時の設定

attribute-buffer	PPB-Update-Mask のビット 1 が設定されている場合は、パネルから読み込まれた属性を保持するバッファを指定します。

PPB-Buffer-Offset	更新領域から読み込まれた最初の文字は、テキストバッファに置かれます。
PPB-Panel-ID	読み込むパネルの識別ハンドル。
PPB-Rectangle-Offset	更新領域から読み込まれない文字数。
PPB-Update-Count	更新領域から読み込まれる文字数。
PPB-Update-Height	読み込まれる更新領域の高さ。
PPB-Update-Mask	ビット 0 が設定されている場合は、パネルから読み込まれるテキストがテキストバッファに保存されます。ビット 1 が設定されている場合は、パネルから読み込まれる属性が属性バッファに保存されます。
PPB-Update-Start-Col	読み込む指定パネルの最初のカラム。
PPB-Update-Start-Row	読み込む指定パネルの最初の行。
PPB-Update-Width	読み込む更新領域の幅。
PPB-Vertical-Stride	テキストバッファの行幅。
text-buffer	PPB-Update-Mask のビット 0 が設定されている場合は、パネルから読み込まれたテキストを保持するバッファを指定します。

終了時の設定

なし

備考

PF-Read-Panel は ACCEPT 文のような動作をしません。つまり、ユーザはパネルへ情報を入力できません。ユーザからの情報を取得するには、ACCEPT 文を使用する必要があります。

Panels が格納するデータを収容するのに十分なバッファであることを確認します。バッファが十分でない場合は、Panels はプログラムの作業場所節にあるバッファの次の項目にも情報を上書きします。

その他の参照項目

『*PPB-Update-Mask*』

例

この例は、10 文字の幅で 30 行の高さのパネルを仮定しています。パネルのハンドルは、ws-save-panel-id に保存されています。

この例は、7 行目の最初のカラムから始まり、パネルの末尾までの部分を読み込みます。テキストのみ (属性を含まず) が、プログラムの作業場所節に定義されている text-buffer に読み込まれます。

```
* 更新領域を読み込むために定義します。この場合は、次の作業を行います。
* パネルの 7 行目から 30 行目の
* 領域を定義します。合計 24 行、各行は 10 文字です。

       move 10 to ppb-update-width.
       move 24 to ppb-update-height.

* パネルの 7 行目、最初のカラムから読み込みを開始します
* (0,0 はパネルの左上隅)。

       move 6 to ppb-update-start-row.
       move 0 to ppb-update-start col.

* 領域内の最初の文字から
* 読み込まれます
* (0 は領域の左上隅)。
       move 0 to ppb-rectangle-offset.
* バッファへ 240 文字を読み込みます
*  (10 文字の行を 24 行)。
       move 240 to ppb-update-count.
* バッファの最初の文字から始まる更新を
* 読み込みます。
       move 1 to ppb-buffer-offset.
* 更新領域の 1 行は、10 文字幅です。
       move 10 to ppb-vertical-stride.
* テキストのみを読み込みます。属性は含みません
* (PPB-Update-Mask のビット 0 を設定)。
       move x"01" to ppb-update-mask.
* パネルの識別子は ws-save-panel-id に保存されています。
       move ws-save-panel-id to ppb-panel-id.
* パネルから text-buffer へテキストを読み込みます。
* text-buffer は、PIC X(240) で定義されています。
       move pf-read-panel to ppb-function.
       call "PANELS" using panels-parameter-block
       text-buffer.
       if ppb-status not = zero
    *          (中止処理)
```

PF-Redraw-Screen (値 2)

画面全体の内容を最新表示します。

パラメータ

なし

備考

現在画面に表示されているパネルは、出現した順序で最新表示されます。 使用可能なパネルで隠されていない画面部分は、現在の背景の文字や属性を表示します。

この機能が最初に呼ばれると、画面がクリアされ、画面全体が現在の背景文字と属性に設定されます。

この機能は、Panels 以外の方法で画面更新を行うようなプログラムで画面が乱れたときに、特に有効です。 この機能の呼び出しで、画面を Panels で認識できる状態にリセットします。

例

```
       move pf-redraw-screen to ppb-function.
       call "PANELS" using panels-parameter-block.
       if ppb-status not = zero
*         (中止処理)
```

PF-Scroll-Panel (値 10)

パネル領域をスクロールします。

領域は、パネル内の特定の領域です。 この領域は、カラムで上下、左右にスクロールできます。

PPB-Update-Mask に設定されるビットに対応して、一定のデータ項目を指定します。

ビット	必要なデータ項目
0	text-buffer、PPB-Buffer-Offset、および PPB-Vertical-Stride
1	attribute-buffer、PPB-Buffer-Offset、および PPB-Vertical-Stride
2	PPB-Fill-Character
3	PPB-Fill-Attribute

パラメータ

attribute-buffer	pic x(n).
PPB-Buffer-Offset	pic 9(4) comp-x.
PPB-Fill-Attribute	pic x.
PPB-Fill-Character	pic x.
PPB-Panel-ID	pic 9(4) comp-x.
PPB-Scroll-Count	pic 9(4) comp-x.
PPB-Scroll-Direction	pic 9(2) comp-x.
PPB-Update-Height	pic 9(4) comp-x.
PPB-Update-Mask	pic x.
PPB-Update-Start-Col	pic 9(4) comp-x.
PPB-Update-Start-Row	pic 9(4) comp-x.
PPB-Update-Width	pic 9(4) comp-x.
PPB-Vertical-Stride	pic 9(4) comp-x.
text-buffer	pic x(n)。

起動時の設定

attribute-buffer	PPB-Update-Mask のビット 1 が設定されている場合は、領域を埋める属性バッファを指定します。
PPB-Buffer-Offset	PPB-Update-Mask のビット 0 が設定されている場合は、text-buffer から表示する最初の文字位置を指定します。 PPB-Update-Mask のビット 1 が設定されている場合は、attribute-buffer から表示する最初の属性位置を指定します。
PPB-Fill-Attribute	PPB-Update-Mask のビット 3 が設定されている場合は、領域を埋める属性を指定します。
PPB-Fill-Character	PPB-Update-Mask のビット 2 が設定されている場合は、領域を埋める文字を指定します。
PPB-Panel-ID	スクロールされるパネルの識別ハンドル。この値は、PF-Create-Panel 呼び出しで返されます。
PPB-Scroll-Count	スクロールする行数。
PPB-Scroll-Direction	スクロールの方向。 0 - 上 1 - 下 2 - 左 3 - 右

PPB-Update-Height	スクロールする領域の高さ。
PPB-Update-Mask	『*Panels のパラメータブロック*』を参照してください。
PPB-Update-Start-Col	スクロールで影響を受ける領域の最初のカラム。
PPB-Update-Start-Row	スクロールで影響を受ける領域の最初の行。
PPB-Update-Width	スクロールする領域の幅。
PPB-Vertical-Stride	PPB-Update-Mask のビット 0 または 1 が設定されている場合は、text-buffer または attribute-buffer の行の長さを指定します。
text-buffer	PPB-Update-Mask のビット 0 が設定されている場合は、領域を埋めるテキストバッファを指定します。

終了時の設定

なし

備考

Panels では、テキストと属性を別々にスクロールできません。

その他の参照項目

『*パネルの概要*』および『*PPB-Update-Mask*』

例

パネルは 50 文字幅で 15 行の高さで定義されています。パネルのハンドルは、ws-save-panel-id に保存されています。

この例では、パネルのテキストと属性を 15 行、上スクロールし、空きになっているテキストを、テキストバッファと属性バッファからのテキストと属性で埋めます。テキストと属性バッファの最初の 15 行は、すでに使用されています (画面に表示されている)。そのため、テキストと属性バッファの 16 行目からスクロールが始まります (テキストと属性バッファの 1 行は 50 文字幅です)。

```
* 更新領域を定義します。
* これは、スクロールするパネルのブロックです。
* ここでは、パネル全体を領域として定義します。
      move 50 to ppb-update-width.
      move 15 to ppb-update-height.
* 更新領域は、パネルと同じサイズなので、
```

```
* パネルの 1 行目、1 カラムから始まります
* (0,0 はパネルの左上隅)。
       move 0 to ppb-update-start-row.
       move 0 to ppb-update-start-col.
* 領域を上スクロールします。
       move 0 to ppb-scroll-direction
* 15 行分スクロールします。
       move 15 to ppb-scroll-count.
* パネルは、ユーザ定義のテキストと属性バッファで埋められます。
* 最初の 15 行 (バッファ内の各行は 50 文字幅)
* は、すでに表示されています。
* バッファの 16 行目からスクロールを開始します。
* PPB-Buffer-Offset は、バッファのどの文字から始まるかを指定します
* (ここでは 1 は最初の文字)。
* そこで、751 番目の文字からスクロール開始します
* (50 文字の行を 15 行の 750 文字が
* すでに表示されています)。
       move 751 to ppb-buffer-offset.
* 更新領域の 1 行は、50 文字幅です。
       move 50 to ppb-vertical-stride.
* テキストと属性バッファを使用して、
* パネルの空きになっている部分を更新します (PPB-Update-Mask のビット 0 と 1
を設定)。
* この変更は、使用可能なパネルの領域スクロールが
* 画面に表示されます (PPB-Update-Mask のビット 4 と 5 を設定)。
* 16 進数 x"33" は 2 進表示で 00110011 です。
       move x"33" to ppb-update-mask.
* パネルの識別子は、ws-save-panel-id に保存されています。
       move ws-save-panel-id to ppb-panel-id.
* パネルをスクロールします。
* テキストバッファは text-buffer、属性バッファは attribute-buffer です。
       move pf-scroll-panel to ppb-function.
       call "PANELS" using panels-parameter-block
       text-buffer
       attribute-buffer
       if ppb-status not = zero
*         (中止処理)
```

PF-Set-Panel-Name (値 14)

指定したパネルに名前を関連付けます。

パラメータ

PPB-Panel-ID	pic 9(4) comp-x.
Panel-Name-Buffer.	
Panel-Name-Length	pic 99 comp-x.
Panel-Name-Text	pic x(30).

起動時の設定

PPB-Panel-ID	名前を関連付けたいパネルの識別ハンドル。
Panel-Name-Length	名前の長さ
Panel-Name-Text	PPB-Panel-ID で指定されたパネルに関連付けられた名前。

終了時の設定

なし

備考

この機能を使用すると、PF-Get-First-Panel と PF-Get-Next-Panel を使用して、各パネルに関連付けられた名前を参照できます。

その他の参照項目

『*PF-Get-First-Panel*』および『*PF-Get-Next-Panel*』

例

この例は、パネルを作成し、前述のように、プログラムの作業場所節に Panel-Name-Buffer を定義していると仮定しています。

```
       move 11 to panel-name-length.
       move "Dummy Panel" to panel-name-text.
       move pf-set-panel-name to ppb-function.
       call "PANELS" using panels-parameteer-block
       panel-name-buffer.
       if ppb-status not = zero
   *         (中止処理)
```

PF-Set-Panel-Stack (値 18)

指定された使用可能なパネルのリストに従い、使用可能なパネルのスタックをリセットし、画面を再描画します。

パラメータ

Panels-Order-List	集団項目には、次のものが含まれます。
POL-Count	pic 9(4) comp-x.
POL-ID	pic 9(4) comp-x occurs 254.

起動時の設定

POL-Count	リスト中の使用可能なパネル数。
POL-ID	再描画される使用可能なパネルの識別ハンドルのリスト。

終了時の設定

なし

備考

POL-ID で指定されていないパネルは使用不能になり、POL-ID で指定されているパネルは使用可能になり表示されます。

その他の参照項目

『*PF-Get-Panel-Stack*』

例

次の例は、アプリケーションのパネルの状態を、パネルを変更する前に保存し、その後、元の状態に復元するために、PF-Get-Panel-Stack と PF-Set-Panel-Stack 機能をともに使用する方法を示しています。

この例は、副プログラムを呼び出す主プログラムの抜粋を示し、主プログラムのパネルの状態が副プログラム呼び出し前に保存され、その後、復元される場合を示しています。副プログラムは、主プログラムで表示される会社のロゴである 1 つのパネルを必要とします。

```
working-storage section.
01 panels-order-list.
    03  pol-count          pic 9(4) comp-x.
    03  pol-id             pic 9(4) comp-x occurs 254.
01 save-panels-order-list  pic x(510).
```

```
 01 company-logo-panel-id   pic 9(4) comp-x.
  ...
 procedure division.
       ...
      call-subprogram section.
       ...
* 副プログラムを呼び出す前に
* 現在のパネルのスタックをコピーします。
      move pf-get-panel-stack to ppb-function
      perform make-panels-call
* スタックのコピーを保存します。
      move panels-order-list to save-panel-order-list
* 次に、画面から主プログラムのすべてのパネルを削除します。
* ただし、会社ロゴのパネルは削除しません。
      move 1 to pol-count
      move company-logo-panel-id to pol-id(1)
      move pf-set-panel-stack to ppb-function
      perform make-panels-call
* 会社ロゴを除き、スタックは空となり、
* 副プログラムを呼び出します。
      call "SUBPROG"
* 副プログラムから戻ると、
* パネルを副プログラム呼び出し前の状態に復元します。
      move pf-set-panel-stack to ppb-function
      perform make-panels-call
       ...
      make-panels-call section.
      call "PANELS" using panels-parameter-block
      panels-order-list.
```

PF-Set-Screen-Backdrop (値 1)

現在の背景文字と属性を変更します。

パラメータ

PPB-Fill-Attribute	pic x.
PPB-Fill-Character	pic x.

起動時の設定

PPB-Fill-Attribute	新しい背景属性。
PPB-Fill-Character	新しい背景文字。

終了時の設定

なし

備考

この機能は、画面を自動的には更新しません。単に Panels に対する背景文字と属性を定義します。変更された背景文字と属性が表示される前に、PF-Redraw-Screen を使用して画面を再描画する必要があります。

その他の参照項目

『*PF-Redraw-Screen*』

例

この例は、背景文字をアスタリスク (*) にします。背景属性は、青の背景に赤の前景に設定されます。

```
     move "*" to ppb-fill-character.
     move x"14" to ppb-fill-attribute.
     move pf-set-screen-backdrop to ppb-function.
     call "PANELS" using panels-parameter-block.
     if ppb-status not = zero
*         (中止処理)
```

PF-Shift-Panel (値 4)

ウィンドウを移動したり、そのサイズを変更したりします。

パラメータ

PPB-First-Visible-Col	pic 9(4) comp-x.
PPB-First-Visible-Row	pic 9(4) comp-x.
PPB-Panel-ID	pic 9(4) comp-x.
PPB-Panel-Start-Column	pic 9(4) comp-x.
PPB-Panel-Start-Row	pic 9(4) comp-x.
PPB-Visible-Height	pic 9(4) comp-x.
PPB-Visible-Width	pic 9(4) comp-x.

起動時の設定

PPB-First-Visible-Col	パネル内のウィンドウの新しい水平位置。値 0 は、ウィンドウの左端がパネルの左端に位置していることを示します。
PPB-First-Visible-Row	パネル内のウィンドウの新しい垂直位置。値 0 は、ウィンドウの上端がパネルの上端に位置していることを示します。
PPB-Panel-ID	外観を変更したいパネルの識別ハンドル。この値は、PF-Create-Panel 呼び出しで返されます。
PPB-Panel-Start-Column	画面上のパネルの新しい水平位置。値 0 は、パネルの左端が画面の左端に位置していることを示します。
PPB-Panel-Start-Row	画面上のパネルの新しい垂直位置。値 0 は、パネルの上端が画面の上端に位置していることを示します。
PPB-Visible-Height	表示するウィンドウの新しいテキスト行数。
PPB-Visible-Width	表示するウィンドウの新しいテキストのカラム数。

終了時の設定

なし

備考

PPB-Panel-Height と PPB-Panel-Width を除くすべてのフィールドは、この機能で変更できます。パネルのサイズを定義すると、変更できません。PPB-Panel-Height と PPB-Panel-Width に任意の値を定義すると、無視されます。

ウィンドウに割り当てられた値を変更しない場合は、PF-Get-Panel-Info を使用して、ウィンドウの現在のサイズと位置を取得し、再設定できます。

ウィンドウのサイズや位置の変更は、そのウィンドウに関連付けられたパネルを使用可能にしたときにのみ表示できます。ただし、パネルが使用不能でも、ウィンドウは変更できます。その変更は、次に PF-Enable-Panel を使用してパネルを使用可能にすると表示されます。

その他の参照項目

『*PF-Create-Panel*』、『*PF-Get-Panel-Info*』、および『*PF-Enable-Panel*』

例

この例は、80 文字の幅で 20 行の高さのパネルを仮定しています。このパネルは、画面上の 1 行目、1 カラムから始まります。パネルのウィンドウは、現在 80 カラム幅、5 行の高さでパ

ネルの最初の位置 (左上隅) から始まります。パネルのハンドルは、データ項目 ws-save-panel-id に保存されています。

この例は、ウィンドウの表示する部分を 5 行から 10 行に増加させて、パネル表示を広げます。 すべての他のパラメータは同じままです。

```
* ウィンドウは画面上で 1 行目、1 カラムで始まります。
* ウィンドウの画面上の位置が定義される場合は、
* 0 相対です (行 0 は画面の左上隅)。
*
       move 0 to ppb-panel-start-column.
       move 0 to ppb-panel-start-row.
* パネルのウィンドウは、
* 20 文字幅で、10 行です。
       move 20 to ppb-visible-width.
       move 10 to ppb-visible-height.
* ウィンドウは、画面上の 1 行目、1 カラムから始まります
* (0,0 は画面の左上隅)。
       move 0 to ppb-first-visible-col.
       move 0 to ppb-first-visible-row.
* パネルが最初に作成された場合は、
* 作業場所節の変数 ws-save-panel-id に保存されているパネルの識別子が返されます。
* ここで、Panels は、呼び出しが適用されるパネルを認識する必要があります。
       move ws-save-panel-id to ppb-panel-id.
* ウィンドウを移動します。
       move pf-shift-panel to ppb-function.
       call "PANELS" using panels-parameter-block.
       if ppb-status not = zero
*         (中止処理)
```

PF-Write-Panel (値 11)

パネルへ書き込みます。

パネルに書き込むテキストと属性は、PPB-Update-Mask のビット 4 と 5 を使用し、パネルが使用可能である場合に、画面上に表示されます。

PPB-Update-Mask に設定されるビットに対応して、一定のデータ項目を指定します。

ビット	必要なデータ項目
0	text-buffer、PPB-Buffer-Offset、および PPB-Vertical-Stride

1	attribute-buffer、PPB-Buffer-Offset、および PPB-Vertical-Stride
2	PPB-Fill-Character
3	PPB-Fill-Attribute

パラメータ

attribute-buffer	pic x(n).
PPB-Buffer-Offset	pic 9(4) comp-x.
PPB-Fill-Attribute	pic x.
PPB-Fill-Character	pic x.
PPB-Panel-ID	pic 9(4) comp-x.
PPB-Update-Height	pic 9(4) comp-x.
PPB-Update-Mask	pic x.
PPB-Update-Start-Col	pic 9(4) comp-x.
PPB-Update-Start-Row	pic 9(4) comp-x.
PPB-Update-Width	pic 9(4) comp-x.
PPB-Vertical-Stride	pic 9(4) comp-x.
text-buffer	pic x(n).

起動時の設定

attribute-buffer	PPB-Update-Mask のビット 1 が設定されている場合は、領域を埋める属性バッファを指定します。
PPB-Buffer-Offset	PPB-Update-Mask のビット 0 が設定されている場合は、text-buffer から表示する最初の文字位置を指定します。 PPB-Update-Mask のビット 1 が設定されている場合は、attribute-buffer から表示する最初の属性位置を指定します。
PPB-Fill-Attribute	PPB-Update-Mask のビット 3 が設定されている場合は、領域を埋める属性を指定します。
PPB-Fill-Character	PPB-Update-Mask のビット 2 が設定されている場合は、領域を埋める文字を指定します。
PPB-Panel-ID	書き込むパネルの識別ハンドル。
PPB-Update-Height	更新領域の高さ。
PPB-Update-Mask	『*Panels のパラメータブロック*』を参照してください。

PPB-Update-Start-Col	書き込みで影響を受ける領域の最初のカラム。
PPB-Update-Start-Row	書き込みで影響を受ける領域の最初の行。
PPB-Update-Width	更新領域の幅。
PPB-Vertical-Stride	PPB-Update-Mask のビット 0 または 1 が設定されている場合は、text-buffer または attribute-buffer の行の長さを指定します。
text-buffer	PPB-Update-Mask のビット 0 が設定されている場合は、領域を埋めるテキストバッファを指定します。

その他の参照項目

『*PF-Enable-Panel*』、『*パネルの概要*』、および『*PPB-Update-Mask*』

例

この例は、50 文字の幅で 15 行の高さのパネルを仮定しています。パネルのハンドルは、ws-save-panel-id に保存されています。

この例は、テキストと属性バッファから最初の 15 行をパネルに書き込みます。

```
* 更新領域を定義します。
* これは、更新するパネルのブロックです。
* ここでは、パネル全体を領域として定義します。
     move 50 to ppb-update-width
     move 15 to ppb-update-height
* 「更新領域」は、パネルと同じサイズなので、
* 更新するウィンドウは、パネル内のオフセットが
* ありません。
     move 0 to ppb-update-start-row
     move 0 to ppb-update-start-col
* バッファの最初の文字で始まるテキストと
* 属性バッファを使用して書き込みます。
     move 1 to ppb-buffer-offset
* 更新領域の 1 行は、50 文字幅です。
     move 50 to ppb-vertical-stride
* 更新するウィンドウの最初の文字から書き込みを
* 開始します
* (0 はウィンドウの左上隅)。
     move 0 to ppb-rectangle-offset
* パネルへ 750 文字書き込みます
* (50 文字の行を 15 行分)。
     move 750 to ppb-update-count
```

```
* テキストと属性バッファを使用します (PPB-Update-Mask のビット 0 と 1 を設定)。
* パネルが使用可能であれば、テキストは画面上にすぐに表示されます
* (PPB-Update-Mask のビット 4 と 5 を設定)。
* 16 進数 x"33" は 2 進表示で 00110011 です。
     move x"33" to ppb-update-mask
* パネルの識別子は ws-save-panel-id に保存されています。
     move ws-save-panel-id to ppb-panel-id
* パネルを記述します。
* テキストバッファは text-buffer、属性バッファは attribute-buffer です。
     move pf-write-panel to ppb-function
     call "PANELS" using panels-parameter-block
     ws-text-buffer
     ws-attribute-buffer.
     if ppb-status not = zero
*         (中止処理)
```

5.6 Adis 機能を Panels で使用

拡張 ACCEPT/DISPLAY 構文 (Adis) は、Panels を含むプログラム内で ACCEPT と DISPLAY 文を使用できます。次の点を考慮する必要があります。

- 拡張 ACCEPT と DISPLAY 文のみがパネルを操作できます。標準 ANSI ACCEPT と DISPLAY 文を画面へ直接書き込むために使用すると、問題が発生します。x"B7" COBOL システムライブラリルーチンや名前による呼び出しの画面処理ルーチンを Panels とともに使用すると、同じような問題が発生します。
- AT、LINE、または COLUMN 句で指定する位置は、現在のパネルの左上隅に対する相対位置であり、画面に対するものではありません。
- 任意の数のパネルを作成でき、同時に 254 パネルまで使用可能にできます。Adis が Panels で使用される場合は、6 つのパネルが Adis で使用されます。

5.6.1 Adis 機能

次の Adis 機能が Panels で使用可能です。

- 1 - Adis インジケータやエラーメッセージを有効化または無効化します。
- 49 - Panels で使用するために Adis を初期化します。
- 56 - ACCEPT と DISPLAY 文で操作するパネルを指定します。
- 57 - 1 回の呼び出しで複数の DISPLAY 文を使用して画面を更新します。
- 62 - Adis で Panels の使用を中断します。
- 63 - Adis での Panels の使用状態を返します。

Adis インジケータの有効化と無効化 - x"AF" 機能 1

x"AF" 機能 1 は、Adis エラーメッセージとインジケータを有効にしたり、無効にしたりします。

x"AF" 機能 49 を使用し、Adis に Adis と Panels をともに使用することを通知させると、この機能で 6 つの定義済みパネルが作成されます。4 つのパネルは、Adis インジケータ (挿入、置換、自動スキップ、およびフィールド末尾超過インジケータ) を表示するために使用されます。他のフィールドは、Adis エラーメッセージを表示するために使用されます。Panels を使用する場合には、これらのメッセージを抑制できます。

構文

```
call x"AF" using    function-code
                    parameter
```

パラメータ

function-code	PIC X COMP-X VALUE 1. この機能を実行中にエラーが発生すると、終了時に function-code が 255 に設定されます。
parameter	PIC X(4) フィールドは、次の値のどれかです。 x"01322C01" はエラーメッセージパネルを無効にする。 x"03323803" はインジケータパネルを無効にする。 x"01322C02" はエラーメッセージパネルを有効にする。 x"03323800" はインジケータパネルを有効にする。

Panels を使用するための Adis の初期化 - x"AF" 機能 49

x"AF" 機能 49 は、画面と同じ寸法のパネルを作成します。このパネルは自動的に使用可能になり、黒の背景に白の前景属性で埋められた領域です。画面は、クリアされます。

この機能は、プログラムの最初に 1 回のみ呼び出します。

構文

```
call x"AF" using    init-panels-fn
                    parameter
```

パラメータ

init-panels-fn	PIC X COMP-X VALUE 49.
parameter	任意の英数字データ項目。Adis は、そのサイズと内容を無視します。

例

```
 01 init-panels-fn    pic x comp-x value 49.
 01 parameter         pic x.
           ...
     call x"af" using init-panels-fn
                      parameter
```

この呼び出しで作成されたパネルは、自動的に使用可能になり、画面から以前の内容をクリアします。 そのパネルは、デフォルトの背景文字で埋められた領域です。 デフォルトの属性設定は、黒の背景に白の前景です。 すべての ACCEPT と DISPLAY 文は、他のパネルが指定されない限り、このパネルを操作します。 この詳細については、『ACCEPT/DISPLAY 用のパネル指定』の機能を参照してください。この呼び出しで作成されるパネル識別子 (Panel-ID) の値は、常に 0 です。

この呼び出しは、Adis で使用される 6 つのパネルを作成します。 次のリストの最初のパネル以外は、すべて画面の下部に表示されます。

- 全画面のパネル。
- Adis エラーメッセージパネル (1 カラム目から 77 カラム目まで)。
- 挿入インジケータ用の 1 文字のパネル。
- フィールド末尾超過インジケータ用の 1 文字のパネル。
- 自動スキップインジケータ用の 1 文字のパネル。
- 置換インジケータ用の 1 文字のパネル。

インジケータの詳細については、『*Adiscf を使用した Adis の構成*』の章を参照してください。

この 6 つのパネルは、プログラムでは使用できません。

その他の参照項目

『*Adis インジケータの無効化と有効化 - x"AF" 機能 1*』

ACCEPT/DISPLAY 用のパネル指定 - x"AF" 機能 56

x"AF" 機能 56 は、プログラムの ACCEPT と DISPLAY 文に、どのパネルを使用するかを Adis に通知します。 すべての ACCEPT と DISPLAY 文は、他のパネルを指定してこの機能を別に呼び出さない限り、指定されたパネルを操作します。

構文

```
call x"AF" using    specify-panel-fn
                    PPB-Panel-ID
```

パラメータ

init-panels-fn	PIC X COMP-X VALUE 56.
work-panel-ID	PIC XX COMP-X フィールドは、ACCEPT と DISPLAY 文で使用されるパネルの識別ハンドルを含みます。 x"AF" 機能 49 の呼び出しの結果で Adis で作成されたパネルを使用すると、パネルの識別子は 0 です。

備考

Adis で作成されたデフォルトのパネルを操作しない場合は、この機能を呼び出す前に、パネルを作成する必要があります。

指定されたパネルが存在しない場合は、値 255 が specify-panel-function に返され、以前に指定されたパネルが ACCEPT と DISPLAY 文で使用されます。

ACCEPT 文でパネルを操作すると、そのパネルは、他のパネルで表示される部分が隠されないように、自動的に使用可能になります。 Panels のウィンドウがパネルよりも小さく、ACCEPT 文からの結果でのカーソル位置が表示されていない場合には、Adis は、カーソル位置のテキストが表示されるように、ウィンドウを位置付けします。 パネルを操作する DISPLAY 文は、テキストと属性を更新したときの PF-Write-Panel と同じです。

例

```
01 specify-panel-fn    pic x comp-x value 56.
01 work-panel-id       pic xx comp-x.
           ...
*      (パネルを作成し、その識別子を
*      work-panel-id に転記するコード)
           ...
     call x"af" using specify-panel-fn
                      work-panel-id.
```

その他の参照項目

『*PF-Write-Panel*』

遅延 DISPLAY の有効化と無効化 - x"AF" 機能 57

x"AF" 機能 57 は、いくつかの DISPLAY 文を、実行するたびに画面を更新しないで実行します。 画面を更新する 1 つのみの呼び出しを実行でき、結果的に画面更新処理が高速になることがあります。

画面の更新は、ACCEPT 文を実行するまで、または、画面を更新し、画面更新を再度有効にするこの呼び出しを使用するまで、遅延されます (次を参照)。

構文

```
call x"AF" using    control-update-fn
                    control-update-param
```

パラメータ

control-update-fn	PIC X COMP-X VALUE 57.
control-update-param	PIC X COMP-X は、次の値のどちらかを含みます。 1 遅延更新方式を使用する。 0 遅延更新方式を無効にし、画面を直ちに更新する。

備考

ACCEPT 文の実行結果は、パラメータを 0 に設定してこの呼び出しを実行する場合と同じです。そのため、遅延更新方式を使用したい場合は、ACCEPT 文を実行後、パラメータを 1 に設定してこの呼び出しを別に実行します。

例

```
 01 control-update-fn       pic x comp-x value 57.
 01 control-update-param    pic x comp-x.
   ...
     move 1 to control-update-param.
     call x"af" using control-update-fn
                      control-update-param.
            ...
* ここでは、DISPLAY 文は画面に反映されません。
       ...
     move 0 to control-update-param.
     call x"af" using control-update-fn
                              control-update-param.
* ここで、以前の表示情報が表示されます。
```

Adis での Panels の使用中断 - x"AF" 機能 62

x"AF" 機能 62 は、パネルが必要でない場合に、画面への直接書き込みを可能にするために、Adis による Panels の使用を中断します。

この機能を使用する前に、x"AF" 機能 49 を使用し、Panels を使用するために Adis を初期化する必要があります。

構文:

```
call x"AF" using  suspend-function
                  suspend-parameter
```

パラメータ

suspend-function	PIC X COMP-X VALUE 62.
suspend-parameter	PIC X COMP-X は、次の値のどちらかを含みます。 0 Panels の使用を中断。 1 Panels の使用を再開。

備考

Adis が Panels を処理するために初期化されていない場合は、値 255 が *suspend-function* に返されます。

例

```
 01 suspend-function     pic x comp-x value 62.
 01 suspend-parameter    pic x comp-x.
 ...
*   (Adis が Panels を使用するように設定するコード)
           ...
* Panels 処理の中断
     move 0 TO suspend-parameter.
     call x"af" using suspend-function
                      suspend-parameter.
           ...
*  Panels 処理の再開
     move 1 TO suspend-parameter.
     call x"af" using suspend-function
                      suspend-parameter.
```

Adis での Panels 状態の返却 - x"AF" 機能 63

x"AF" 機能 63 は、どのように Adis が ACCEPT と DISPLAY 文に対して Panels を使用しているのかの詳細を取得します。

構文:

```
call x"AF" using  get-status-fn
```

get-status-param

パラメータ

get-status-fn PIC X COMP-X VALUE 63.

get-status-param 次の定義をもつ集団項目。

usage-flag	PIC X COMP-X で定義され、Adis が Panels を使用するために初期化されたかどうかが指定されます。0 は、Adis が Panels を使用していないことを示します。
status-flag	PIC X COMP-X で定義され、Panels の処理が Adis 上で現在有効かどうかが指定されます。
current-id	PIC XX COMP-X で定義され、Adis で使用されているパネルのハンドルが指定されます。

例

```
01 get-status-fn          pic x comp-x value 63.
01 get-status-param.
    03 panels-usage-flag     pic x comp-x.
    03 panels-status-flag    pic x comp-x.
    03 current-panel-id      pic xx comp-x.
 ...
    call x"af" using get-status-fn
                     get-status-param.
    if panels-usage-flag = 0
        display "Adis は Panels を使用していません。"
    else
        if panels-status = 0
            display "Adis による Panels の使用は中断されています。"
        else
            display "Adis により Panels が使用されています。"
            display "Adis が使用している PANEL の"
                    "ID は次のとおりです。"
                    current-panel-id
        end-if
    end-if.
```

5.6.2 Adis 機能の呼び出し

Adis 機能の構文は、次のとおりです。

```
call x"AF" using    function-code
                    parameter
```

パラメータ

function-code	実行される機能の番号を含む PIC X COMP-X フィールド。
parameter	機能によって変化します。*parameter* の設定については、各機能の説明を参照してください。

注:

- すべての Adis 呼び出しは、無視されることもありますが、2 番目のパラメータが必要です。
- この呼び出しの実行中にエラーが発生すると、パラメータは 255 の値を返します。

その他の参照項目

『*Panels の機能*』および『*Panels 機能用のパラメータ*』

5.7 Panels アプリケーションのパッケージ化

プログラムで、Adis 機能なしの Panels を使用する場合は、**panels.obj** をリンクします。

プログラムで、Adis 機能を Panels とともに使用する場合は、次をリンクします。

- **panels.obj**
- **adis.obj**
- **adisinit.obj**
- **adiskey.obj**

5.8 Panels でよく使用される用語

Adis

拡張 ACCEPT/DISPLAY 文を実行するランタイムモジュール。Adis モジュールは、データを画面に直接表示しないで、パネルへ DISPLAY 文で表示したり、パネルからデータを ACCEPT 文で取り込んだりできます。

属性バッファ

プログラムの作業場所節で定義され、パネルに含まれている属性。

クリッピング

指定されたウィンドウのサイズが作成されるサイズよりも大きい場合に、そのウィンドウサイズを縮小します。 たとえば、開始カラムの値が、指定されたサイズのウィンドウで画面の右端を越えると、ウィンドウの幅は自動的に画面に合うように縮小されます。

使用不能

特定のウィンドウが、画面上で表示されません。

使用可能

特定のウィンドウが、画面上で表示されます。

パネル

書き込むことができる仮想画面で、実際の画面より大きくも、小さくもできます。 パネルが使用される前に、パネルの高さと幅、画面上での表示部分の寸法と位置、パネルのどの部分を実際に表示するかを指定して作成される必要があります。

パラメータブロック

プログラムと Panels 間の通信手段。 パラメータブロックは、Panels で特定の機能を実行するために、初期設定するフィールドで構成されます。 機能に関連するフィールドは、Panels を呼び出す前に初期化する必要があります。

長方形 (四角形または領域と表記されている場合もあります)

独立してスクロールできるパネルの長方形の部分。

テキストバッファ

プログラムの作業場所節で定義され、パネルに含まれているテキスト。

ウィンドウ

パネルの表示部分。

5.9 Panels のエラーコード

ここでは、Panels の呼び出しで、PPB-Status に返される値を説明します。 **panlink.cpy** ファイルに出現する順序で、ニーモニックが示されています。

PE-No-Error (値 0)

- 要求された機能は成功しました。

PE-Not-Created (値 1)

- 要求されたパネルが存在しません。この値は、作成されていないパネルへある機能を実行した場合に返されます。パネルを使用する前に、パネルを作成する必要があります。

PE-Invalid-ID (値 2)

- パネルの識別子が有効なパネルの識別子の範囲を越えています。同時に表示できるパネルの最大数は、254 個です。

PE-Invalid-Parameter (値 3)

- 指定された更新パラメータが、パネル内の領域を定義していません。PPB-Update-Start-Row や PPB-Update-Start-Col で指定された値が、パネル外であり、パネルの高さや幅よりも大きい。

PE-Invalid-Function (値 4)

- PPB-Function で指定された機能番号が、有効な機能番号ではありません。

PE-No-Room-For-Panel (値 5)

- プログラムが多数のパネルを作成しています。最大数のパネルが、すでに作成されています。最大数 (65534) のパネルがすでに作成されている場合には、このエラーは PF-Create-Panel の呼び出しで発生します。

PE-Panel-Too-Large (値 6)

- 指定されたパネルが大きすぎます。プログラムで大きすぎるパネルを指定すると、この値が返されます。これには、次の理由が考えられます。
    1. プログラムで、とても大きいパネルを作成しています。つまり、PPB-Panel-Width × PPB-Panel-Height > 65535 であるようなパネルを作成しています。
    2. パネルの幅が、2000 文字よりも大きい。
    3. パネルのテキストバッファや属性バッファを保持するのに十分なメモリがありません。

PE-Invalid-Direction (値 7)

- PPB-Scroll-Direction に指定された値が、0 ～ 3 の範囲にありません。

PE-Invalid-Scroll-Count (値 8)

- スクロールする行やカラムが多すぎます。PPB-Scroll-Count の指定値が、パネルの行数やカラム数よりも大きい (スクロールの方向に依存)。

PE-Cannot-Initialise (値 9)

- Panels が初期化できません。システムのプログラムで最初に Panels を呼び出したときにのみ発生する致命的エラーです。 何らかの原因で、Panels が初期化できません。

その他の参照項目

『PPB-Status』

HTML ファイルが開けません

- HTML ファイルを正常に開くことができませんでした。使用しているシステムに World Wide Web ブラウザがインストールされていないか、使用しているブラウザと .htm ファイルの拡張子の関連付けが設定されていません。

  Net Express には Microsoft Internet Explorer V4 の World Wide Web ブラウザが含まれており、これは Net Express CD からインストールできます。また、それ以外のブラウザも使用できます。 オンライン ブウクを完全に表示するには、フレームが表示できる Microsoft Internet Explorer のようなウェブ ブラウザを使用する必要があります。

第 6 章 : COBOL システムライブラリルーチン

COBOL システムライブラリは、文字画面に文字や属性を描画したり、キーボードで押されたキーを読み込んだり、マウスと通信するルーチンのセットです。

ここでは、それらのルーチンを一覧表示し、詳しく説明します。

6.1 画面ルーチン

CBL_CLEAR_SCR	画面をクリアします。
CBL_GET_CSR_POS	カーソル位置を取得します。
CBL_GET_SCR_SIZE	画面のサイズを取得します。
CBL_READ_SCR_ATTRS	属性を読み込みます。
CBL_READ_SCR_CHARS	文字を読み込みます。
CBL_READ_SCR_CHATTRS	文字と属性を読み込みます。
CBL_SET_CSR_POS	カーソル位置を設定します。
CBL_SWAP_SCR_CHATTRS	文字と属性をスワップします。
CBL_WRITE_SCR_ATTRS	属性を書き込みます。
CBL_WRITE_SCR_CHARS	文字を書き込みます。
CBL_WRITE_SCR_CHARS_ATTR	属性をもつ文字を書き込みます。
CBL_WRITE_SCR_CHATTRS	文字と属性を書き込みます。
CBL_WRITE_SCR_TTY	TTY 形式の文字を書き込みます。
CBL_WRITE_SCR_N_ATTR	属性の書き込みを繰り返します。
CBL_WRITE_SCR_N_CHAR	文字の書き込みを繰り返します。
CBL_WRITE_SCR_N_CHATTR	文字と属性の書き込みを繰り返します。
X"A7" 機能 17	カーソルタイプを設定します。
X"A7" 機能 18	リダイレクト可能なコンソール I/O。
X"A7" 機能 25	スクリーンタイプを取得します。
X"AF" 機能 1	Adis の構成を行います。
X"AF" 機能 18	文字を表示します。
X"AF" 機能 22	アラームを鳴らします。
X"AF" 機能 26	文字を取得します。
X"E5"	ベルを鳴らします。

6.1.1 画面ルーチンの概要

画面ルーチンは、画面位置 (screen-position) パラメータを指定します。この COBOL システムでは、画面の左上隅が行 0、カラム 0 です。たとえば、行 5、カラム 8 から始まる属性を変更する場合は、行番号 4 と カラム番号 7 を指定します。

CBL_CLEAR_SCR ルーチンは、原始プログラムを変更しないで、どのような環境でも使用可能なグラフィック文字の表示を行う汎用的な線描画機能を提供します。

X"A7" ルーチンを画面処理に使用する前には、CBL_CLEAR_SCR ルーチンを使用します。

6.2 キーボードルーチン

CBL_GET_KBD_STATUS	キーボードで文字を検査します。
CBL_READ_KBD_CHAR	キーボードから文字を読み込みます (エコーなし)。
X"B0" 機能 0	ファンクションキーテーブルを作成します。
X"B0" 機能 2	シフトキーの状態を検査します。
X"B0" 機能 4	キーボード割り込みを無効にします。

6.3 マウスルーチン

CBL_GET_MOUSE_MASK	マウスイベントマスクを取得します。
CBL_GET_MOUSE_POSITION	マウスの画面上の座標を取得します。
CBL_GET_MOUSE_STATUS	キュー内のイベントの数を取得します。
CBL_HIDE_MOUSE	マウスポインタを隠します。
CBL_INIT_MOUSE	マウスサポートの初期化を行います。
CBL_READ_MOUSE_EVENT	マウスのイベントキューを読み込みます。
CBL_SET_MOUSE_MASK	マウスイベントマスクを設定します。
CBL_SET_MOUSE_POSITION	マウスの画面上の座標を設定します。
CBL_SHOW_MOUSE	マウスポインタを表示します。
CBL_TERM_MOUSE	マウスサポートを終了します。
PC_GET_MOUSE_SHAPE	マウスポインタの形状を取得します。
PC_SET_MOUSE_HIDE_AREA	マウスを隠す領域を設定します。
PC_SET_MOUSE_SHAPE	マウスポインタの形状を設定します。

6.3.1 マウスルーチンの概要

これらのルーチンは、UNIX では動作しません。

UNIX:
UNIX 環境では、CBL_INIT_MOUSE は、マウスが存在しないことを示す 0 でない RETURN-CODE を返します。 前述の他の CBL_ の呼び出しは、スタブとして扱われ、何の影響もありません。

6.3.2 マウスルーチンの使用方法

マウスは、オプションのリストから選択したり、画面上でオブジェクトを移動したりするアプリケーションで役に立ちます。

これらのルーチンを使用するためには、接続されているマウスがシステムで認識されていることを確認してください。

マウスポインタがある画面の領域で ANSI ACCEPT や DISPLAY 文操作の実行中は、マウスを隠す必要があります。

ルーチンの説明で参照される属性は、画面の属性であり、ユーザ属性ではありません。画面の左上隅が、行 0、カラム 0 です。

6.3.3 マウスイベント

マウスが移動されたり、ボタンが押されたり放されたりすると、マウスのハードウェアが割り込みを発生させます。 マウスのデバイスドライバは、制御を行い、設定するマスクによって、イベントをキューに保存したり、無視したりします。 これにより、アプリケーションがイベントを読み込む前に、後続するイベントが発生した場合でも、イベントが失われることはありません。 マウスルーチンで、イベントのキューを読み込み、キュー内のイベント数を決定できます。

イベントが生成されると、その内容が event-data と呼ばれるデータ構造体に保存されます。 マスクが許可されている場合 (次を参照) には、キューに追加されます。event-data の構成は、次のとおりです。

```
event-type          pic x(2) comp-x.
event-time          pic x(4) comp-x.
event-row           pic x(2) comp-x.
event-col           pic x(2) comp-x.
```

パラメータの詳細は、次のとおりです。

event-type 動作 (つまり、状態の変更) は、次のとおりです。

ビット 7-4 予約済み。

ビット 3 ボタン 3 を押した。

ビット 2 ボタン 2 を押した。

ビット 1 ボタン 1 を押した。

ビット 0　マウスが移動した。

ボタンを放すと、対応するボタンのビットが 1 から 0 に変更されます。たとえば、マウスを移動し、同時にボタン 1 が押されると、event-type は 3 (ビット 0 とビット 1) になります。

event-time　イベントが発生したときと、任意の開始時間との間の経過時間。

event-row、*event-col*　イベントが発生したときのマウスの位置。

6.3.4 イベントマスク

イベントマスクは、どのイベントがキューに置かれたか、または、どのイベントを無視するかをシステムに通知します。event-type と同じ構造です。イベントは、その対応するマスクのビットがオンであるか、または、マスクビットがオンになっている他の状態がオンである場合のみに、キューに置かれます。event-data がキューに置かれると、各状態のビットは正しく設定され、マスクは無効になります。

たとえば、「マウスが移動した」に対するマスクビットがオンであったり、マウスボタンを押し続けてそのボタンのマスクビットがオンであったりしても、オペレータがマウスを移動すると、イベントが発生します。

6.4 キー

名前による呼び出しのルーチンの説明は、アルファベット順に現れます。各説明には、ルーチン名、機能、および次のような項目が必要に応じて含まれています。

構文

ルーチンを呼び出すときの CALL 文を示します。

オプションの RETURNING 句も示されています。各ルーチンは、処理結果を示す値を返します。特に示さない限り、0 は成功、1 は不成功を示します。この値は、RETURNING 句で指定されたデータ項目、ここでは、status-code に置かれます。この RETURNING 句が省略されると、特殊レジスタ RETURN-CODE に置かれます (call-convention ビット 2 が設定されている場合は、RETURN-CODE は変更されません)。

status-code は数字データ項目で、0 ～ 65535 の正数を保持します。たとえば、PIC X(2) COMP-5 となります。

ルーチン名は、大文字で記述する必要があります。

パラメータ

RETURNING や USING 句に指定するパラメータを説明します。角かっこで囲まれたパラメータ、たとえば、[parameter1] はオプションで、ルーチンのすべての形式に必要とされません。

起動時の設定

起動時にどのパラメータが渡されたかを示します。

終了時の設定

終了時にどのパラメータに返されたかを示します。

1 バイト以上のビットが参照されます。ビット 0 が LSB です。

備考

ルーチン使用上の必要な追加情報を示します。

その他の参照項目

関連項目のリスト。

6.5 ルーチンの説明

CBL_CLEAR_SCR

指定した文字と属性で全画面をクリアします。

構文

```
call "CBL_CLEAR_SCR"    using       character
                  attribute
                          returning status-code
```

パラメータ

character	pic x.
attribute	pic x.
status-code	『キー』を参照してください。

起動時の設定

character	書き込む文字。
attribute	書き込む属性。

終了時の設定

なし

CBL_GET_CSR_POS

カーソル位置を返します。

構文

```
call "CBL_GET_CSR_POS" using      screen-position
                       returning status-code
```

パラメータ

screen-position	次の定義をもつ集団項目。
row-number	pic x comp-x.
column-number	pic x comp-x.
status-code	『*キー*』を参照してください。

起動時の設定

なし

終了時の設定

screen-position	カーソルの画面位置。左上隅は行 0、カラム 0 です。カーソルが表示されていない場合は、row-number と column-number は、ともに 255 に設定されています。『*画面ルーチンの概要*』を参照してください。

CBL_GET_KBD_STATUS

キーボードの読み込みを待機している文字があるかどうかを検査します。

構文

```
call "CBL_GET_KBD_STATUS" using      key-status
                          returning status-code
```

パラメータ

key-status　　pic x comp-x.
status-code　　『*キー*』を参照してください。

起動時の設定

なし

終了時の設定

key-status　　キーボードの状態。
0　文字がない。
1　文字がある。

備考

いくつかの UNIX では、この方法でキーボードをポーリングできません。UNIX システムのマニュアルを調べ、この条件に該当するかどうか確認してください。

CBL_GET_MOUSE_MASK

マウスイベントマスクを返します。

構文

```
call "CBL_GET_MOUSE_MASK" using      mouse-handle
                                event-mask
                                 returning status-code
```

パラメータ

mouse-handle　　pic x(4) comp-x.
event-mask　　pic x(2) comp-x.
status-code　　『*キー*』を参照してください。

起動時の設定

mouse-handle　　以前の CBL_INIT_MOUSE 呼び出しで取得したマウス識別子。

終了時の設定

event-mask　　『*マウスルーチンの概要*』を参照してください。

備考

このルーチンは、UNIX 環境では効果がありません。

その他の参照項目

『*マウスルーチンの概要*』

CBL_GET_MOUSE_POSITION

マウスポインタの画面位置を返します。

構文

```
call "CBL_GET_MOUSE_POSITION" using     mouse-handle
                                        mouse-position
                                    returning status-code
```

パラメータ

mouse-handle	pic x(4) comp-x.
mouse-position	次の定義をもつ集団項目。
mouse-row	pic x(2) comp-x.
mouse-col	pic x(2) comp-x.
status-code	『*キー*』を参照してください。

起動時の設定

mouse-handle　以前の CBL_INIT_MOUSE 呼び出しで取得したマウス識別子。

終了時の設定

mouse-position　マウスポインタの画面位置。

備考

このルーチンは、UNIX 環境では効果がありません。

その他の参照項目

『*マウスルーチンの概要*』

CBL_GET_MOUSE_STATUS

キュー内のイベント数を検査します。

構文

```
call "CBL_GET_MOUSE_STATUS" using      mouse-handle
                                       queued-events
                                   returning status-code
```

パラメータ

mouse-handle	pic x(4) comp-x.
queued-events	pic x(2) comp-x.
status-code	『*キー*』を参照してください。

起動時の設定

mouse-handle	以前の CBL_INIT_MOUSE 呼び出しで取得したマウス識別子。

終了時の設定

queued-events	キュー内のイベント数。

備考

このルーチンは、UNIX 環境では効果がありません。

その他の参照項目

『*マウスルーチンの概要*』

CBL_GET_SCR_SIZE

画面のサイズに関する情報を返します。

構文

```
call "CBL_GET_SCR_SIZE" using      depth
                           width
                                   returning status-code
```

パラメータ

depth pic x comp-x.
width pic x comp-x.
status-code 『*キー*』を参照してください。

起動時の設定

なし

終了時の設定

depth 行数。
width カラム数。

CBL_HIDE_MOUSE

マウスポインタを隠します。

構文

```
call "CBL_HIDE_MOUSE" using      mouse-handle
                      returning status-code
```

パラメータ

mouse-handle pic x(4) comp-x
status-code 『*キー*』を参照してください。

起動時の設定

mouse-handle 以前の CBL_INIT_MOUSE 呼び出しで取得したマウス識別子。

終了時の設定

なし

備考

このルーチンは、UNIX 環境では効果がありません。

このルーチンが呼び出された後は、マウスイベントは有効ですが、マウスポインタは表示されません。

その他の参照項目

『*マウスルーチンの概要*』

CBL_INIT_MOUSE

マウスサポートの初期化を行います。このルーチンは、他のマウスルーチンの前に呼び出される必要があります。

構文

```
call "CBL_INIT_MOUSE" using      mouse-handle
                            mouse-buttons
                                 returning status-code
```

パラメータ

mouse-handle	pic x(4) comp-x
mouse-buttons	pic x(2) comp-x
status-code	『*キー*』を参照してください。

起動時の設定

なし

終了時の設定

mouse-handle	マウス識別子。このマウス識別子は、この後呼び出されるマウスルーチンに渡されます。
mouse-buttons	マウスボタンの数。

備考

このルーチンから返される *status-code* または RETURN-CODE が 0 でない状態をもっている場合は、マウス処理が使用不能で、他のマウスルーチンの呼び出しはできません。

UNIX 環境では、このルーチンは常に 0 でない *status-code* を返します。

その他の参照項目

『*マウスルーチンの概要*』

CBL_READ_KBD_CHAR

文字が入力されるのを待機し、エコーしないで読み込みます。

構文

```
call "CBL_READ_KBD_CHAR" using      char
                         returning  status-code
```

パラメータ

char	pic x.
status-code	『*キー*』を参照してください。

起動時の設定

なし

終了時の設定

char	入力された文字は ASCII です。

CBL_READ_MOUSE_EVENT

マウスのイベントをキューから読み込み、イベントに関する情報を返します。

構文

```
call "CBL_READ_MOUSE_EVENT" using  mouse-handle
                                   event-data
                                   read-type
                            returning status-code
```

パラメータ

mouse-handle	pic x(4) comp-x.
event-data	『*マウスルーチンの概要*』を参照してください。
read-type	pic x comp-x.
status-code	『*キー*』を参照してください。

起動時の設定

mouse-handle	以前の CBL_INIT_MOUSE 呼び出しで取得したマウス識別子。
read-type	キュー内にイベントが存在しない場合の処理を示します。 0　直ちに戻ります。 1　イベントを待機し、戻ります。

終了時の設定

event-data　　『*マウスルーチンの概要*』を参照してください。

備考

このルーチンは、UNIX 環境では効果がありません。

イベントがイベントキュー内に存在しない場合には、このルーチンからの戻りは *read-type* の値に依存します。*read-type* が 0 の場合は、event-data にすべて 0 を格納して直ちに戻ります。*read-type* が 1 の場合には、イベントがキューに置かれるまで戻りは遅延します。

その他の参照項目

『*マウスルーチンの概要*』

CBL_READ_SCR_ATTRS

画面から属性を読み込みます。

構文

```
call "CBL_READ_SCR_ATTRS" using      screen-position
                                     attribute-buffer
                                     string-length
                           returning status-code
```

パラメータ

screen-position	次の定義をもつ集団項目。
row-number	pic x comp-x.
column-number	pic x comp-x.
attribute-buffer	pic x(*n*).
string-length	pic x(2) comp-x.
status-code	『*キー*』を参照してください。

起動時の設定

screen-position	読み込みを始める画面位置。左上隅は、行 0、カラム 0 です。『*画面ルーチンの概要*』を参照してください。
string-length	読み込む文字列の長さ。

終了時の設定

attribute-buffer	属性が画面から読み込まれます。データ項目は、少なくとも *string-*

	length で指定された長さで、その長さを越えた位置は変更されません。
string-length	画面の末尾に達すると、読み込まれた長さが返されます。

CBL_READ_SCR_CHARS

画面から文字を読み込みます。

構文

```
call "CBL_READ_SCR_CHARS" using      screen-position
                                     character-buffer
                                     string-length
                          returning  status-code
```

パラメータ

screen-position	次の定義をもつ集団項目。
row-number	pic x comp-x.
column-number	pic x comp-x.
character-buffer	pic x(*n*).
string-length	pic x(2) comp-x.
status-code	『*キー*』を参照してください。

起動時の設定

screen-position	読み込みを始める画面位置。左上隅は、行 0、カラム 0 です。『*画面ルーチンの概要*』を参照してください。
string-length	読み込む文字列の長さ。

終了時の設定

character-buffer	文字が画面から読み込まれます。データ項目は、少なくとも *string-length* で指定された長さで、その長さを越えた位置は変更されません。
string-length	画面の末尾に達すると、読み込まれた長さが返されます。

CBL_READ_SCR_CHATTRS

画面から文字と属性を読み込みます。

構文

```
call "CBL_READ_SCR_CHATTRS" using     screen-position
                                      character-buffer
                                      attribute-buffer
                                      string-length
                            returning status-code
```

パラメータ

screen-position	次の定義をもつ集団項目。
row-number	pic x comp-x.
column-number	pic x comp-x.
character-buffer	pic x(*n*).
attribute-buffer	pic x(*n*).
string-length	pic x(2) comp-x.
status-code	『*キー*』を参照してください。

起動時の設定

screen-position	読み込みを始める画面位置。左上隅は、行 0、カラム 0 です。『*画面ルーチンの概要*』を参照してください。
string-length	読み込む文字列の長さ。

終了時の設定

character-buffer	文字が画面から読み込まれます。データ項目は、少なくとも *string-length* で指定された長さで、その長さを越えた位置は変更されません。
attribute-buffer	属性が画面から読み込まれます。 データ項目は、少なくとも *string-length* で指定された長さで、その長さを越えた位置は変更されません。
string-length	画面の末尾に達すると、読み込まれた長さ (セル。つまり、文字と属性の組) が返されます。

CBL_SET_CSR_POS

カーソルを移動します。

構文

```
call "CBL_SET_CSR_POS" using     screen-position
                       returning status-code
```

パラメータ

screen-position　　次の定義をもつ集団項目。
　　row-number　　pic x comp-x.
　　column-number　　pic x comp-x.
status-code　　『*キー*』を参照してください。

起動時の設定

screen-position　　カーソルを置く画面位置。左上隅は、行 0、カラム 0 です。『*画面ルーチンの概要*』を参照してください。

終了時の設定

なし

備考

カーソルを表示しない場合は、*row-number* と *column-number* を 255 に設定します。

CBL_SET_MOUSE_MASK

マウスイベントマスクを設定します。

構文

```
call "CBL_SET_MOUSE_MASK" using      mouse-handle
                                     event-mask
                          returning  status-code
```

パラメータ

mouse-handle　　pic x(4) comp-x.
event-mask　　pic x(2) comp-x.
status-code　　『*キー*』を参照してください。

起動時の設定

mouse-handle　　以前の CBL_INIT_MOUSE 呼び出しで取得したマウス識別子。
event-mask　　『*マウスルーチンの概要*』を参照してください。

終了時の設定

なし

備考

このルーチンは、UNIX 環境では効果がありません。

CBL_GET_MOUSE_MASK は、どのイベントが有効になったかを検索する場合に、最初に呼び出されます。

その他の参照項目

『*マウスルーチンの概要*』

CBL_SET_MOUSE_POSITION

マウスポインタを移動します。

構文

```
call "CBL_SET_MOUSE_POSITION" using      mouse-handle
                                         mouse-position
                                   returning status-code
```

パラメータ

mouse-handle	pic x(4) comp-x.
mouse-position	次の定義をもつ集団項目。
mouse-row	pic x(2) comp-x.
mouse-col	pic x(2) comp-x.
status-code	『*キー*』を参照してください。

起動時の設定

mouse-handle	以前の CBL_INIT_MOUSE 呼び出しで取得したマウス識別子。
mouse-position	マウスポインタを移動させる画面位置。

終了時の設定

なし

備考

このルーチンは、UNIX 環境では効果がありません。

その他の参照項目

『*マウスルーチンの概要*』

CBL_SHOW_MOUSE

マウスポインタを表示します。

構文

```
call "CBL_SHOW_MOUSE" using      mouse-handle
                      returning  status-code
```

パラメータ

mouse-handle	pic x(4) comp-x
status-code	『*キー*』を参照してください。

起動時の設定

mouse-handle	以前の CBL_INIT_MOUSE 呼び出しで取得したマウス識別子。

終了時の設定

なし

備考

このルーチンは、UNIX 環境では効果がありません。

マウスサポートが CBL_INIT_MOUSE 呼び出しで初期化されると、ポインタはこのルーチンが呼ばれるまでは表示されません。この呼び出し後に、ルーチンでマウスを隠したり、マウスサポートを終了させるルーチンが呼び出したりするまで、システムはマウスポインタを表示します。このルーチンは、以前の PC_SET_MOUSE_HIDE_AREA で定義された競合領域を解除します。

その他の参照項目

『*マウスルーチンの概要*』

CBL_SWAP_SCR_CHATTRS

画面から文字と属性をスワップします。

構文

```
call "CBL_SWAP_SCR_CHATTRS" using     screen-position
                                      character-buffer
                                      attribute-buffer
                                      string-length
                                returning status-code
```

パラメータ

screen-position	次の定義をもつ集団項目。
row-number	pic x comp-x.
column-number	pic x comp-x.
character-buffer	pic x(*n*).
attribute-buffer	pic x(*n*).
string-length	pic x(2) comp-x.
status-code	『*キー*』を参照してください。

起動時の設定

screen-position	書き込みを始める画面位置。左上隅は、行 0、カラム 0 です。『*画面ルーチンの概要*』を参照してください。
character-buffer	書き込む文字。
attribute-buffer	書き込む属性。
string-length	書き込む文字列の長さ。画面の末尾に達すると、書き込みは画面の末尾で終了します。

終了時の設定

character-buffer	文字が画面から読み込まれます。データ項目は、少なくとも *string-length* で指定された長さで、その長さを越えた位置は変更されません。
attribute-buffer	属性が画面から読み込まれます。データ項目は、少なくとも *string-length* で指定された長さで、その長さを越えた位置は変更されません。
string-length	画面の末尾に達すると、スワップされた長さ (セル。つまり、文字と属性の組) が返されます。

その他の参照項目

『*画面ルーチンの概要*』

CBL_TERM_MOUSE

マウスサポートを終了し、内部リソースを解放します。

構文

```
call "CBL_TERM_MOUSE" using      mouse-handle
                      returning status-code
```

パラメータ

mouse-handle	pic x(4) comp-x
status-code	『*キー*』を参照してください。

起動時の設定

mouse-handle	以前の CBL_INIT_MOUSE 呼び出しで取得したマウス識別子。

終了時の設定

なし

備考

このルーチンは、UNIX 環境では効果がありません。

このルーチンは CBL_INIT_MOUSE で割り当てられた内部リソースを解放します。このルーチン終了後は、*mouse-handle* は有効でなくなり、CBL_INIT_MOUSE 以外のマウスルーチンの呼び出しはエラーとなります。

その他の参照項目

『*マウスルーチンの概要*』

CBL_WRITE_SCR_ATTRS

画面に属性を書き込みます。

構文

```
call "CBL_WRITE_SCR_ATTRS" using      screen-position
                                      attribute-buffer
                                      string-length
                           returning status-code
```

パラメータ

screen-position　次の定義をもつ集団項目。
　　row-number　pic x comp-x.
　　column-number　pic x comp-x.
attribute-buffer　pic x(*n*).
string-length　pic x(2) comp-x.
status-code　『*キー*』を参照してください。

終了時の設定

なし

コメント:

この COBOL システムでは、点滅の属性はサポートされません。点滅指定した文字は、結果的に明るい背景色で表示されます。

その他の参照項目

『*画面ルーチンの概要*』

CBL_WRITE_SCR_CHARS

画面に文字を書き込みます。

構文

```
call "CBL_WRITE_SCR_CHARS" using      screen-position
                                      character-buffer
                                      string-length
                               returning status-code
```

パラメータ

screen-position　次の定義をもつ集団項目。
　　row-number　pic x comp-x.
　　column-number　pic x comp-x.
character-buffer　pic x(*n*).
string-length　pic x(2) comp-x.
status-code　『*キー*』を参照してください。

起動時の設定

screen-position　書き込みを始める画面位置。左上隅は、行 0、カラム 0 です。『*画面ルーチンの概要*』を参照してください。

character-buffer 書き込む文字。

string-length 書き込む文字列の長さ。画面の末尾に達すると、書き込みは画面の末尾で終了します。

終了時の設定

なし

その他の参照項目

『*画面ルーチンの概要*』

CBL_WRITE_SCR_CHARS_ATTR

画面に書き込む文字には、すべて同じ属性を与えます。

構文

```
call "CBL_WRITE_SCR_CHARS_ATTR"
                          using      screen-position
                                     character-buffer
                                     string-length
                                     attribute
                          returning  status-code
```

パラメータ

screen-position	次の定義をもつ集団項目。
row-number	pic x comp-x.
column-number	pic x comp-x.
character-buffer	pic x(*n*).
string-length	pic x(2) comp-x.
attribute	pic x.
status-code	『*キー*』を参照してください。

起動時の設定

screen-position 書き込みを始める画面位置。左上隅は、行 0、カラム 0 です。『*画面ルーチンの概要*』を参照してください。

character-buffer 書き込む文字。

attribute 書き込む属性。

string-length 書き込む文字列の長さ。画面の末尾に達すると、書き込みは画面の末尾で終了します。

終了時の設定

なし

コメント

この COBOL システムでは、点滅の属性はサポートされません。点滅指定した文字は、結果的に明るい背景色で表示されます。

その他の参照項目

『*画面ルーチンの概要*』

CBL_WRITE_SCR_CHATTRS

画面に文字と属性を書き込みます。

構文

```
call "CBL_WRITE_SCR_CHATTRS" using    screen-position
                                      character-buffer
                                      attribute-buffer
                                      string-length
                            returning status-code
```

パラメータ

screen-position	次の定義をもつ集団項目。
row-number	pic x comp-x.
column-number	pic x comp-x.
character-buffer	pic x(*n*).
attribute-buffer	pic x(*n*).
string-length	pic x(2) comp-x.
status-code	『*キー*』を参照してください。

起動時の設定

screen-position 書き込みを始める画面位置。左上隅は、行 0、カラム 0 です。『*画面ルーチンの概要*』を参照してください。

character-buffer 書き込む文字。

attribute-buffer 書き込む属性。

string-length 書き込む文字列の長さ。画面の末尾に達すると、書き込みは画面の末尾で終了します。

終了時の設定

なし

コメント:

この COBOL システムでは、点滅の属性はサポートされません。点滅指定した文字は、結果的に明るい背景色で表示されます。

その他の参照項目

『*画面ルーチンの概要*』

CBL_WRITE_SCR_N_ATTR

画面上の文字列の位置に指定された属性を書き込みます。

構文

```
call "CBL_WRITE_SCR_N_ATTR" using     screen-position
                                      attribute
                                      fill-length
                            returning status-code
```

パラメータ

screen-position	次の定義をもつ集団項目。
row-number	pic x comp-x.
column-number	pic x comp-x.
attribute	pic x.
fill-length	pic x(2) comp-x.
status-code	『*キー*』を参照してください。

起動時の設定

screen-position 書き込みを始める画面位置。左上隅は、行 0、カラム 0 です。『*画面ルーチンの概要*』を参照してください。

attribute 書き込む属性。

string-length 属性を書き込む画面位置の番号。画面の末尾に達すると、書き込みは画面の末尾で終了します。

終了時の設定

なし

コメント:

この COBOL システムでは、点滅の属性はサポートされません。点滅指定した文字は、結果的に明るい背景色で表示されます。

その他の参照項目

『*画面ルーチンの概要*』

CBL_WRITE_SCR_N_CHAR

画面上の位置の文字列に指定された文字を書き込みます。

構文

```
call "CBL_WRITE_SCR_N_CHAR" using      screen-position
                                       character
                                       fill-length
                            returning  status-code
```

パラメータ

screen-position	次の定義をもつ集団項目。
row-number	pic x comp-x.
column-number	pic x comp-x.
character	pic x.
fill-length	pic x(2) comp-x.
status-code	『*キー*』を参照してください。

起動時の設定

screen-position 書き込みを始める画面位置。左上隅は、行 0、カラム 0 です。『*画面ルーチンの概要*』を参照してください。

attribute-buffer 書き込む属性。

string-length　　文字を書き込む画面位置の番号。画面の末尾に達すると、書き込みは画面の末尾で終了します。

終了時の設定

なし

その他の参照項目

『*画面ルーチンの概要*』

CBL_WRITE_SCR_N_CHATTR

画面上の位置の文字列に指定された文字と属性を書き込みます。

構文

```
call "CBL_WRITE_SCR_N_CHATTR" using    screen-position
                                       character
                                       attribute
                                       fill-length
                             returning status-code
```

パラメータ

screen-position	次の定義をもつ集団項目。
row-number	pic x comp-x.
column-number	pic x comp-x.
character	pic x.
attribute	pic x.
fill-length	pic x(2) comp-x.
status-code	『*キー*』を参照してください。

起動時の設定

screen-position	書き込みを始める画面位置。左上隅は、行 0、カラム 0 です。『*画面ルーチンの概要*』を参照してください。
character	書き込む文字。
attribute	書き込む属性。
string-length	文字と属性の組を書き込む画面位置の番号。画面の末尾に達すると、書き込みは画面の末尾で終了します。

終了時の設定

なし

コメント

この COBOL システムでは、点滅の属性はサポートされません。点滅指定した文字は、結果的に明るい背景色で表示されます。

その他の参照項目

『*画面ルーチンの概要*』

CBL_WRITE_SCR_TTY

画面の現在位置から文字を書き込み、スクロールします。

構文

```
call "CBL_WRITE_SCR_TTY" using      character-buffer
                                    string-length
                         returning  status-code
```

パラメータ

character-buffer	pic x(*n*).
string-length	pic x(2) comp-x.
status-code	『*キー*』を参照してください。

起動時の設定

character-buffer	書き込む文字。
string-length	書き込む文字列の長さ。画面の端から出てしまうと、画面が 1 行、上スクロールされ、その下部の行で書き込みが続けられます。

終了時の設定

なし

PC_GET_MOUSE_SHAPE

マウスポインタの形状に関する情報を返します。

構文

```
call "PC_GET_MOUSE_SHAPE" using      mouse-handle
                                     reserved-item
                                     mouse-ptr-shape
                          returning  status-code
```

パラメータ

mouse-handle	pic x(4) comp-x.
reserved-item	pic x(10).
mouse-ptr-shape	次の定義をもつ集団項目。
char_AND_mask	pic x comp-x.
attr_AND_mask	pic x comp-x.
char_XOR_mask	pic x comp-x.
attr_XOR_mask	pic x comp-x.
status-code	『*キー*』を参照してください。

起動時の設定

mouse-handle	以前の CBL_INIT_MOUSE 呼び出しで取得したマウス識別子。
reserved-item	将来のために予約されています。

終了時の設定

mouse-ptr-shape	ポインタの現在の形状を作成したビットマップ。

備考

このルーチンは、DOS、Windows、および OS/2 環境のみで使用可能です。

mouse-ptr-shape のマスクは、ポインタの画面位置に適用されるビットマップで、その上にマウス形状を重ね合わせます。ポインタの形状は、画面上のマウス位置の画面文字と char_AND_mask で論理積がとられ、その結果と char_XOR_mask で排他的論理和がとられて、PC_SET_MOUSE_SHAPE で作成されます。そして、その結果を画面に表示します。 属性も同じように作成されます。

その他の参照項目

『*マウスルーチンの概要*』

PC_SET_MOUSE_HIDE_AREA

マウスが表示されない領域 (「競合領域」) を定義します。

構文

```
call "PC_SET_MOUSE_HIDE_AREA" using      mouse-handle
                                         collision-area
                                  returning status-code
```

パラメータ

mouse-handle	pic x(4) comp-x.
collision-area	次の定義をもつ集団項目。
top-row	pic x(2) comp-x.
left-col	pic x(2) comp-x.
bottom-row	pic x(2) comp-x.
right-col	pic x(2) comp-x.
status-code	『*キー*』を参照してください。

起動時の設定

mouse-handle	以前の CBL_INIT_MOUSE 呼び出しで取得したマウス識別子。
collision-area	競合領域を定義します。この領域にポインタがあるときは必ず、ポインタが隠されます。この項目の値 0 は、全画面を競合領域にします。競合領域は、同時に 1 つのみです。

終了時の設定

なし

備考

このルーチンは、DOS、Windows V3.1 および 16 ビット版の OS/2 環境のみで使用可能です。

PC_SET_MOUSE_SHAPE

マウスポインタの形状を設定します。

構文

```
call "PC_SET_MOUSE_SHAPE" using      mouse-handle
                                     reserved-item
                                     mouse-ptr-shape
                            returning status-code
```

パラメータ

mouse-handle	pic x(4) comp-x.
reserved-item	pic x(10).
mouse-ptr-shape	次の定義をもつ集団項目。
char_AND_mask	pic x comp-x.
attr_AND_mask	pic x comp-x.
char_XOR_mask	pic x comp-x.
attr_XOR_mask	pic x comp-x.
status-code	『*キー*』を参照してください。

起動時の設定

mouse-handle	以前の CBL_INIT_MOUSE 呼び出しで取得したマウス識別子。
reserved-item	将来のために予約されています。
mouse-ptr-shape	希望するポインタの形状を作成するビットマップ。

終了時の設定

なし

備考

このルーチンは、DOS、Windows、および OS/2 環境のみで使用可能です。

mouse-ptr-shape のマスクは、ポインタの画面位置に適用されるビットマップで、その上にマウス形状を重ね合わせます。ポインタの形状は、画面上のマウス位置の画面文字と char_AND_mask で論理積がとられ、その結果と char_XOR_mask で排他的論理和がとられて、PC_SET_MOUSE_SHAPE で作成されます。そして、その結果を画面に表示します。 属性も同じように作成されます。

このルーチンを呼び出す前に、PC_GET_MOUSE_SHAPE を同一の *mouse-handle* で呼び出す必要があります。 データ項目 reserved-item は、その呼び出しから維持される必要があります。

その他の参照項目

『*マウスルーチンの概要*』

X"A7" 機能 17 - カーソルタイプの設定

カーソルタイプを設定します。

構文

```
call X"A7" using function-code
                 cursor-type
```

パラメータ

function-code	pic x comp-x.
cursor-type	次の定義をもつ集団項目。
cursor-end	pic x comp-x.
cursor-start	pic x comp-x.

起動時の設定

function-code	値 17。
cursor-end	カーソル終了行を指定します。
cursor-start	カーソル開始行を指定します。

終了時の設定

なし

備考

このルーチンは、DOS、Windows、および OS/2 環境のみで使用可能です。

カーソル開始行と終了行の詳細については、『*IBM BIOS テクニカルリファレンス*』に記載されています。

エラーを発生させないために、このルーチンを使用する前に、CBL_CLEAR_SCR ルーチンを呼び出します。

X"A7" 機能 18 - リダイレクト可能なコンソール I/O

コンソール I/O をリダイレクト可能にします。

構文

```
call X"A7" using function-code
                    parameter
```

パラメータ

function-code pic x comp-x.
parameter pic x comp-x.

起動時の設定

function-code 値 18。
parameter 実行する動作
0 DOS 形式のコンソール I/O を無効にします。
1 DOS 形式のコンソール I/O を有効にします。

終了時の設定

なし

備考

このルーチンは、DOS、Windows、および OS/2 環境のみで使用可能です。

通常、RTE は BIOS を使用し、キーボードを読んだり、ディスプレイのビデオマップへ直接書き込んだりします。このルーチンは、オペレーティングシステムの機能のかわりに、RTE を使用します。これにより、オペレータはオペレーティングシステムの機能である、キーボード入力や画面出力をリダイレクトさせることができます。

エラーを発生させないために、このルーチンを使用する前に、CBL_CLEAR_SCR ルーチンを呼び出します。

X"A7" 機能 25 - スクリーンタイプの取得

RTS で認識された、現在のスクリーンタイプを返します。

構文

```
call x"A7" using function-code
                 parameter
```

パラメータ

function-code pic x comp-x.
parameter pic x comp-x.

起動時の設定

function-code 値 25。

終了時の設定

parameter 次のスクリーンタイプを示すビット設定。

ビット 0　0 = モノクロ画面。
ビット 1　予約済み。
ビット 2　予約済み。
ビット 3　1 = EGA 画面。
ビット 4　1 = VGA 画面。
ビット 5　予約済み。
ビット 6　予約済み。
ビット 7　予約済み。

備考

エラーを発生させないために、このルーチンを使用する前に、CBL_CLEAR_SCR ルーチンを呼び出します。

X"AF" 機能 1 - Adis の構成

個々のユーザ機能、Adis、データキー、または実行時の一連の連続したキーの有効化と無効化を含む Adis の機能を構成します。

構文

```
call x"AF" using function-code
                 parameter-block
```

パラメータ

function-code pic x comp-x
parameter-block 次の定義をもつ集団項目。
　key-setting pic x comp-x
　sub-fn-code pic x comp-x
　first-key-id pic x comp-x
　number-of-keys pic x comp-x

起動時の設定

function-code 値 1。
sub-fn-code 次のどれかを指定します。

1 ユーザファンクションキーと互換キーを有効 / 無効にします。
2 Adis 制御キーと他の構成可能なキーを有効 / 無効にします。
3 データキーを有効 / 無効にします。
4 シフトキーを有効 / 無効にします。
5 ロックキーを有効 / 無効にします。

備考

ファンクションキーは、別の x"AF" 呼び出しによって明示的に変更されるか、または、アプリケーションが終了されるまで、有効または無効にされます。ユーザファンクションキーの初期状態は、実行時のオペレーティングシステムに依存します (詳細については、『*拡張 ACCEPT/DISPLAY 構文*』の章を参照してください)。ファンクションキーを有効または無効に設定する呼び出しは、追加的なものです。たとえば、**F1** ファンクションキーを有効にする x"AF" を呼び出し、**F10** キーを有効にする呼び出しをすると、両方のキーが有効になります。

『*拡張 ACCEPT/DISPLAY 構文*』の章で、機能とパラメータの詳細が説明されています。

エラーを発生させないために、このルーチンを使用する前に、CBL_CLEAR_SCR ルーチンを呼び出します。

その他の参照項目

『*X"AF" ルーチン*』

X"AF" 機能 18 - 文字の表示

画面上の現在のカーソル位置に文字を表示します。

構文

```
call x"AF" using function-code
                 char
```

パラメータ

function-code	pic x comp-x
char	pic x

起動時の設定

function-code	値 18。
char	表示される文字。

終了時の設定

なし

備考

エラーを発生させないために、このルーチンを使用する前に、CBL_CLEAR_SCR ルーチンを呼び出します。

その他の参照項目

『*X"AF" ルーチン*』

X"AF" 機能 22 - アラームを鳴らす

端末のアラームを鳴らします。

構文

```
call x"AF" using function-code
                 parameter
```

パラメータ

function-code	pic x comp-x
parameter	pic x

起動時の設定

function-code	値 22。
parameter	予約済み。

終了時の設定

なし

その他の参照項目

『*X"AF" ルーチン*』

X"AF" 機能 26 - 文字の取得

キーボードから文字を取得します。

構文

```
call x"AF" using function-code
                 key-status
```

パラメータ

function-code	pic x comp-x
key-status	次の定義をもつ集団項目。
key-type	pic x.
key-code-1	pic x comp-x.
key-code-2	pic x comp-x.

起動時の設定

function-code 値 26。

終了時の設定

key-type 読み込まれるキータイプは、次のとおりです。

1 ユーザ定義のファンクションキー

2 Adis ファンクションキー

3 データキー

9 エラー

key-code-1 *key-type* が 1 または 2 の場合には、ユーザ定義キーには 0 ～ 127、Adis キーには 0 ～ 39 のキー番号が設定されます。ファンクションキーの詳細については、『*拡張 ACCEPT/DISPLAY 構文*』の章を参照してください。

key-type が 3 の場合は、押されたキーの ASCII コードです。

key-type が 9 の場合は、エラーコードです。

8 使用不能な文字が入力された場合は、key-code-2 にはその文字が格納されます。

9 無効なキーストローク (1 バイト以上) が発生しました。

備考

エラーを発生させないために、このルーチンを使用する前に、CBL_CLEAR_SCR ルーチンを呼び出します。

その他の参照項目

『*X"AF" ルーチン*』

X"B0" 機能 0 - ファンクションキーテーブルの作成

ファンクションキーテーブルを作成します。

構文

```
call X"B0" using function-code
                 key-table
```

パラメータ

function-code	pic x comp-x.	
key-table	次の定義をもつ集団項目。	
key-pressed	pic x comp-x	
key-entry-list	各キーを検出する	
繰り返しの定義をもつ集団項目。		
key-entry-len	pic x comp-x	
key-entry	pic x(*n*)	
key-list-end	pic x comp-x	

起動時の設定

function-code	0 を含んでいます。
key-entry-len	*key-entry* のバイト長 (1 または 2)。
key-entry	必要なキーで生成されるコードシーケンス。
key-list-end	0 を含んでいます。

終了時の設定

key-pressed	使用されているキーと一致するテーブルエントリ (一致しない場合は 0)。

備考

このルーチンは、DOS、Windows、および OS/2 のみで使用可能です。IBM PC 互換環境で特有です。アプリケーションを非 IBM PC 環境で使用する場合は、x"AF" 呼び出しを使用します。

この呼び出しで定義されたキーは、Adiscf で定義されたものよりも優先されます。

Adiscf で設定され、x"AF" で使用されるファンクションキーテーブルは、x"B0" で使用されるファンクションキーテーブルとは区別します。キャリッジリターン x"0D" を Adis に返すことで、x"B0" テーブル内のすべてのキーシーケンスで同時に両方を使用可能にできます。

X"B0" 機能 2 - シフトキーの状態を検査

Shift キー、Alt キー、Ctrl キー、およびロックキーを検査します。

構文

```
call X"B0" using function-code
                 status-block
```

パラメータ

function-code	pic x comp-x
status-block	次の定義をもつ集団項目。
status-inds	pic x
status-id	pic x comp-x
status-res	pic x(6)

起動時の設定

function-code	値 2。
status-id	2 に設定する必要があります。呼び出しで上書きされるために、呼び出しのたびに 2 にリセットする必要があります。
status-res	予約済み。

終了時の設定

status-inds	次のような、キーの状態 (1 = キーが押された、0 = キーは押されていない) を示すビット設定を含んでいます。 ビット 7 Ins ビット 6 Caps Lock ビット 5 Num Lock ビット 4 Scroll Lock ビット 3 Alt ビット 2 Ctrl

ビット 1　Shift(左 Shift のみ)
ビット 0　Shift(右 Shift のみ)

備考

このルーチンは、DOS、Windows、および OS/2 環境のみで使用可能です。

Alt、**Ctrl**、および **Shift** キーに対しては、ルーチンが呼び出されたときに、実際にキーが押された場合に 1 を返します。それ以外では、キーが押されるたびに関連するビットが交互に設定され、解除されます。

機能 2 を有効に使用するには、この呼び出しを一定の間隔で行う必要があります。

X"B0" 機能 4 - キーボード割り込みの無効化

キーボード割り込み (DOS、Windows、および OS/2 上の **Ctrl+Break**) を無効にします。

構文

```
call X"B0" using function-code                    parameter
```

パラメータ

function-code	pic x comp-x.
parameter	pic x comp-x.

起動時の設定

function-code	値 4。
parameter	値 1。

終了時の設定

なし

備考

このルーチンは、アプリケーションの休止に対するキーボード割り込みを無効にします。副プログラムが呼び出される前に、呼び出される必要があります。

この機能は、-i ランタイムスイッチと同じ機能です。

X"E5" - ベルを鳴らす

ベルを鳴らします。

構文

```
call x"E5"
```

パラメータ

なし

備考

ベルの鳴動時間と回数は、環境に依存します。

第 7 章 : IBM PC 表示属性

ここでは、グラフィカル環境でテキストベースの環境やテキストエミュレーションの表示属性を説明します。グラフィカルオブジェクトやグラフィカルオブジェクト内でのテキストとは関連がありません。

7.1 テキスト属性

端末にテキスト文字が表示されるときは、文字が画面上にどう現れるかを記述したテキストに関連付けられた属性のセットをもっています。 文字は、特定の前景と背景色で表示されます。通常のテキストより輝いたり暗くなったり、前景と背景色が反転したり、光ったり、他の属性で表示されたりします。 使用可能な属性は、端末のハードウェアとソフトウェアに依存しますが、多くの共通な使用可能属性は、移植可能な COBOL アプリケーションで、端末の物理的な特性を知らなくても表示できます。

COBOL プログラムは、句に関連付けられた属性を指定する COBOL ACCEPT や DISPLAY 文を使用してこれらの属性をもったテキストを表示できます。 COBOL プログラムは、Panels などの各種の呼び出しインターフェイスを通じて表示属性を指定できます。

7.2 属性の符号化

呼び出しインターフェイスを使用して属性を指定する場合、属性のテキスト文字を対応する属性バイトに符号化すると便利です。 UNIX 環境では、COBOL RTS が属性バイトを解釈し、端末の terminfo エントリ値を通じて端末を制御します。

画面属性は、特定のテキスト文字に関連付けられた概念上の属性バイトで、画面上の各文字位置に 1 つあります。 いくつかの呼び出しインターフェイスは、直接設定できる画面属性の物理的な配列をもっています。

Adis が画面や Panels にアクセスすると、各テキスト文字に参照される画面属性を更新します。ACCEPT や DISPLAY 文が参照しているテキストに属性を定義すると、それが画面属性に使用されます。属性が指定されない場合には、Adis はデフォルトの RTS 属性またはユーザ属性を使用します。使用可能であれば、参照される各テキスト文字に画面属性を使用します。Adis が、たとえば、BLANK SCREEN 句を使用したり、Panels がパネルやスクロールされている行を初期化したりすると、画面属性はデフォルトまたはユーザ属性に設定されます。

デフォルトの RTS 属性は、DOS、Windows、および OS/2 上では黒の背景に茶色の前景です。UNIX では、terminfo データベースに属性が定義されていない限り、黒の背景に白の前景です。 UNIX を使用する場合は、『*Terminfo データベースと端末装置*』を参照してください。

7.2.1 符号化の種類

属性バイトには、4 つの符号化の種類があります。

- 標準の PC モノクロ属性符号化。これは、モノクロ表示の DOS、Windows、および OS/2 システムのデフォルトの符号化です。
- 標準の PC カラー属性符号化。これは、カラー表示の DOS、Windows、および OS/2 システムのデフォルトの符号化です。
- 標準の UNIX モノクロ属性符号化。これは、端末に対する terminfo 記述がカラーであるかどうかに関わりなく、UNIX 上でのデフォルトの符号化です。
- ユーザ選択、または、すべてのディスプレイ用の汎用的な属性符号化。

UNIX:
汎用的な属性符号化を使用して、PC と UNIX 環境間で新しいアプリケーションの移植性をもたせて UNIX のデフォルトの符号化を上書きできます。既存のアプリケーションに対しては、CBL_SCR_SET_PC_ATTRIBUTES ルーチンを呼び出して明示的な PC 符号化を選択します。ディスプレイに対する terminfo エントリは、モノクロまたはカラーのどちらかの符号化を使用するかを決定します。

モノクロとカラーの PC 属性符号化と UNIX 属性符号化の詳細を次に示します。

7.2.2 IBM PC モノクロ属性符号化

DOS、Windows、OS/2:
次の表は、モノクロ表示の PC に対する属性バイトの構造を示しています。ビット 0 が LSB の各バイトの最右で、ビット 7 が MSB です。属性の各ビットの設定 (ビットを 1)、解除 (ビットを 0) の効果を次に示します。

DOS、Windows、および OS/2 では、COBOL システムライブラリルーチンでこれらの属性を設定できます。

ビット	属性
7	点滅
6-4	表示停止または反転表示設定
3	輝度
2-0	通常表示または下線表示

例

x"82" = フラッシュまたは通常表示
x"07" = 通常表示

他のビットの組み合わせは、次のとおりです。

	ビット 6、5、4 設定	ビット 6、5、4 解除	ビット 6、5、4 他の設定

ビット 2、1、0 解除	反転表示	非表示	通常表示
ビット 2、1 解除、ビット 0 設定		下線表示	
他の設定		通常表示	

モノクロ表示での「通常」テキスト (黒の背景にハイライトなしのテキスト) は、カラー表示に使用されるいくつかのビットがモノクロディスプレイでは無視されるので、異なるビット設定でも可能になります。ただし、次の組み合わせのみが、モノクロとカラー表示の両方で「通常」テキストを作成します。

「通常」設定 = 0 0 0 0 0 1 1 1 (10 進 = 007 / 16 進 = 07)

カラーとモノクロ表示で使用されるプログラムから結果の一貫性を確保するために、この設定を使用し「通常」表示を行うことを確認する必要があります。

7.2.3 IBM PC のカラー属性符号化

DOS、Windows、OS/2:
次の表は、PC のカラー表示の属性バイトの構造を示しています。 各ビットの設定 (ビットを 1)、解除 (ビットを 0) の効果を次に示します。

DOS、Windows、および OS/2 では、COBOL システムライブラリルーチンでこれらの属性を設定できます。

ビット	属性
7	点滅
6-4	背景色
3	前景の輝度
2-0	前景 (テキスト) 色

ビット 6、5、および 4 は、次のように背景色を制御します。

ビット 6 5 4	効果
0 0 0	黒
0 0 1	青
0 1 0	緑
0 1 1	シアン
1 0 0	赤

1 0 1	マゼンタ
1 1 0	茶
1 1 1	明るい灰色

ビット 2、1、および 0 は、次のように前景 (テキスト) 色を制御します。

ビット 2 1 0	ビット 3 解除	ビット 3 設定
0 0 0	黒	暗い灰色
0 0 1	青	明るい青
0 1 0	緑	明るい緑
0 1 1	シアン	明るいシアン
1 0 0	赤	明るい赤
1 0 1	マゼンタ	明るいマゼンタ
1 1 0	茶	黄
1 1 1	明るい灰色	白

例

値	効果
1 0 1 0 1 1 0 1 (10 進 = 173、 16 進 = x"AD")	緑の背景に明るいマゼンタでフラッシュするテキスト。
0 0 0 0 0 1 0 0 (10 進 = 4、 16 進 = x"04")	黒の背景に赤のテキスト。
0 0 0 1 0 1 1 1 (10 進 = 23、 16 進 = x"17")	青の背景に明るい灰色のテキスト。

7.2.4 UNIX システムの属性符号化

UNIX:
デフォルトの UNIX 属性バイトは、カラー符号化をもたず、次のように単にモノクロ属性に符号化します。

ビット	属性
7-4	0 に設定
3	点滅

2　　反転表示

1　　下線

0　　ハイライト

各ビットは、属性タイプを示します。これらのビットは、ライブラリの番号による呼び出しルーチン x"A7 " を使用して設定できます。たとえば、下線と点滅表示のような属性の組み合わせを可能にするために、属性バイトの複数ビットを設定できます。使用可能な属性やその有効な組み合わせは、使用する端末の種別、terminfo エントリに依存します。

7.3 UNIX システムのハイライト表示

UNIX:
UNIX のデフォルトでは、COBOL プログラムが HIGHLIGHT や LOWLIGHT 句でテキストを表示する場合、RTS はそれぞれ、端末の terminfo に指定された太字モードと淡色表示モードを使用します。太字や淡色表示モードが指定されていない場合は、HIGHLIGHT や LOWLIGHT 句は効果がありません。

バージョン 3.2 以前の COBOL 製品のデフォルト動作は異なっています。これらの以前のバージョンに対して、COBOL プログラムが HIGHLIGHT 句でテキストを表示した場合には、その効果は、淡色表示モードが端末の terminfo に指定されているかどうかに依存します。淡色表示モードが指定されている場合には、RTS は、デフォルトモードをハイライトテキストに、淡色表示モードを通常テキストに使用します。淡色表示モードが指定されていない場合には、RTS は、太字モードをハイライトテキストに、デフォルトモードを通常テキストに使用します。LOWLIGHT 句は、効果がなく、テキストは通常テキストで表示されます。

旧デフォルト動作を、COBATTR 環境変数を設定して選択することもできます。

7.4 属性ルーチン

次のルーチンは、キャラクタユーザインターフェイスで属性を使用する方法として使用可能です。

X"A7" 機能 6/7　　ユーザ属性

X"A7" 機能 16　　ユーザ属性の無効化と有効化

X"A7" 機能 20/21　　システム属性

次に、これらのルーチンを説明します。

7.4.1 ルーチンの説明

X"A7" 機能 6/7 - ユーザ属性

ユーザ属性の読み込みと設定を行います。

構文

```
call x"A7" using      function-code
                      user-attribute
           returning status-code
```

パラメータ

function-code	pic x comp-x.
user-attribute	pic x comp-x.
status-code	『*COBOL システムライブラリルーチン*』の章にある『*キー*』を参照してください。

起動時の設定

function-code	次の動作を定義します。 6　ユーザ属性の読み込み。 7　ユーザ属性の設定。
user-attribute	機能 7 に対して、ユーザ属性を設定する値。

終了時の設定

user-attribute	機能 6 に対して、現在のユーザ属性の値。

備考

このライブラリルーチンを使用して、UNIX 上でユーザ属性を動的に変更できます。このルーチンを使用する場合は、他の使用のときと同じように、どこでも使用できる通常の COBOL 構文のみを使用するという注意が必要です。

エラーを発生させないために、このルーチンを使用する前に、CBL_CLEAR_SCR ルーチンを呼び出します。

その他の参照項目

『*X"A7" 機能 16 - ユーザ属性の On/Off 設定*』
『*X"A7" 機能 20/21 - システム属性*』

X"A7" 機能 16

ユーザ属性を有効化または無効化します。

構文

```
call x"A7" using function-code
                 parameter
```

パラメータ

function-code	pic x comp-x.
parameter	pic x comp-x.

起動時の設定

function-code	値 16
parameter	次の必要な動作を示します。 0　ユーザ属性を有効にします。 1　ユーザ属性を無効にします。

終了時の設定

なし

備考

UNIX では、ユーザ属性の有効化のみ可能です。ただし、ユーザ属性を通常に設定するために x"A7" 機能 7 を呼び出し、x"A7" 機能 16 をパラメータ値 0 で呼び出して、属性タイプを通常に設定することで、ユーザバイトの無効化をエミュレートできます。

エラーを発生させないために、このルーチンを使用する前に、CBL_CLEAR_SCR ルーチンを呼び出します。

その他の参照項目

『*ユーザ属性の使用*』
『*X"A7" 機能 6/7 - ユーザ属性*』

X"A7" 機能 20/21 - システム属性

システム属性の読み込みと設定を行います。

構文

```
call X"A7" using      function-code
                      sys-attr-array
           returning status-code
```

パラメータ

function-code	pic x comp-x.
sys-attr-array	次の定義をもつ集団項目。
	sys-attr pic x comp-x occurs 16 times.
status-code	『*COBOL システムライブラリルーチン*』の章にある『*キー*』を参照してください。

起動時の設定

function-code	次の動作を定義します。 20 システム属性の読み込み。 21 システム属性の設定。
sys-attr-array	設定するシステム属性を含みます。次に示す属性が使用されます。

sys-attr-1	非表示
sys-attr-2	反転表示
sys-attr-3	通常表示
sys-attr-4	ハイライト表示
sys-attr-5	通常表示で下線付き
sys-attr-6	ハイライト表示で下線付き
sys-attr-7	システムの通常表示
sys-attr-8	フラッシュするハイライト表示
sys-attr-9	未使用
sys-attr-10	ユーザ定義の反転表示
sys-attr-11	ユーザ定義の通常表示
sys-attr-12	ユーザ定義のハイライト表示
sys-attr-13	未使用
sys-attr-14	未使用
sys-attr-15	未使用
sys-attr-16	オペレーティングシステムの通常表示

終了時の設定

sys-attr-array　　読み込んだシステム属性を含みます。

備考

システム属性は、システムのある部分がいろいろな色や陰影付き (画面タイプに依存) で画面を表示するために使用されます。このことは、システム属性を変更することで、表示の色付けを個別に選択できることを意味しています。 パラメータの一部として属性値を必要とする Panels のようなコンポーネントを使用する場合は、これらの呼び出しを使用できます。

いくつかの属性を変更したい場合は、まず現在の属性の設定を読み込み、行いたい変更で更新し、属性を設定します。このようにすることで、関与していない属性の設定に影響なく設定を行えます。

エラーを発生させないために、このルーチンを使用する前に、CBL_CLEAR_SCR ルーチンを呼び出します。

その他の参照項目

『*X"A7" 機能 6/7 - ユーザ属性*』

第 8 章 : ACUCOBOL ウィンドウ構文

ウィンドウ構文は、下位互換性のためのみに提供されています。 新しいアプリケーションを作成するときには、この機能を使用しないことをお奨めします。

COBOL システムは、ACUCOBOL ウィンドウ構文を部分的にサポートし、端末の画面に線やボックスを描画したり、物理的な端末で仮想端末ウィンドウを作成できます。 すべての ACCEPT/DISPLAY 文は、現在のウィンドウ (ACCEPT 書き方 1、2 や 3、DISPLAY 書き方 1、DISPLAY WINDOW/LINE/BOX 文を除く) 内で動作します。 この構文では、基礎となる表示を保持したり、回復したりできます。

8.1 ウィンドウ構文の概要

この COBOL システムには、次のウィンドウサポート構文を含んでいます。

- ACCEPT 文での BEFORE TIME 指定

  ACCEPT 文の書き方 5 は、タイムアウト期間を指定できる BEFORE TIME 指定があります。 ユーザがこの期間内にデータを入力しない場合には、ACCEPT 文は自動的に終了します。

- DISPLAY WINDOW

  DISPLAY WINDOW は、端末ウィンドウ (画面の領域) を作成し、それを現在のウィンドウにします。 これは仮想端末のようなものであり、この後の ACCEPT/DISPLAY 文で使用される画面位置は、そのウィンドウの左上隅に対しての相対位置です。

- DISPLAY LINE

  DISPLAY LINE は、端末 (実際の端末または仮想端末) 上に線を描画します。端末上の最適なモードが自動的に使用されます。 DISPLAY BOX 文とともに使用することで、DISPLAY LINE 文は端末上にフォームを描画できます。

- DISPLAY BOX

  DISPLAY BOX は、端末上にボックスを描画します。端末上の最適なモードが自動的に使用されます。 DISPLAY LINE 文とともに使用することで、DISPLAY LINE 文は端末上にフォームを描画できます。

- CLOSE WINDOW

  CLOSE WINDOW は、ウィンドウを削除します。 ウィンドウを POP-UP ウィンドウに指定すると、基礎となる表示を復元できます。

8.2 ウィンドウサポートの有効化

ウィンドウ構文を利用するためには、PREPROCESS"WINDOW1" コンパイラ指令を使用する必要があります。

この指令は、次の 2 つのどちらかの方法で指定します。

- ソースファイルに、次の行を記述します。

  ```
  $SET preprocess"window1"
  ```

- コマンド行から、PREPROCESS"WINDOW1" 指令をインクルードします。

PREPROCESS"WINDOW1" 指令は、NOERRQ、AUTOCLOSE や COLOR とは異なる最新のコンパイラ指令です。エラーが発生すると、コンパイラは処理を続けるかを尋ね、応答を待ちます。この機能を無効にするには、PREPROCESS"WINDOW1" の後に NOERRQ を指定します。

8.3 ウィンドウサポート構文

次に、PREPROCESS"WINDOW1" 指令で使用可能なウィンドウ構文の詳細を説明します。

8.3.1 ACCEPT 文

ACCEPT 文の書き方 5 は、『*言語リファレンス*』で説明されていますが、次の指定があります。

```
BEFORE TIME time-out
```

一般規則

1. BEFORE TIME 指定は、一定時間経過後に ACCEPT 文を自動的に終了させます。タイムアウト値は、100 分の 1 秒単位で指定します。たとえば、「BEFORE TIME 500」は、タイマ値を 5 秒と指定します。
2. ユーザは、タイマが切れるまでに ACCEPT 文にデータを入力する必要があります。ユーザがデータ入力を始めるとすぐに、タイマが解除され、ユーザはエントリを完了するのに十分な時間をとれます。ユーザがタイマが切れるまでにデータを入力しない場合には、ACCEPT 文は終了します。

8.3.2 CLOSE WINDOW 文

CLOSE WINDOW 文は、現在の端末ウィンドウを閉じます。

形式

```
CLOSE WINDOW window-save-area
```

構文規則

1. *window-save-area* は、PIC X(10) 句で記述される基本データ項目です。 DISPLAY WINDOW 文の POP-UP AREA 指定のオブジェクトである必要があります。

一般規則

1. CLOSE WINDOW 文は、DISPLAY WINDOW 文の POP-UP AREA オプションで作成されるポップアップウィンドウを削除するために使用されます。

2. *window-save-area* は、この実行単位で実行される DISPLAY WINDOW 文の POP-UP 指定のオブジェクトである必要があります。 さらに、その実行以降は CLOSE WINDOW 文のオブジェクトではなくなり、他の文でも変更することはできません。 この規則に違反すると、未定義の結果になります。

3. CLOSE WINDOW 文は、対応する DISPLAY WINDOW 文が実行されたときに、アクティブなウィンドウにある端末の画面内容を復元します。 言い換えれば、DISPLAY WINDOW 文で作成されたウィンドウが画面から削除され、ポップアップウィンドウの下にあった画面内容で置き換えられるということです。

4. 対応する DISPLAY WINDOW 文の実行でアクティブになったウィンドウが、アクティブウィンドウになります。その結果、最前面のウィンドウとなります。また、他のウィンドウに重なる場合もあります。

現在のウィンドウは、それぞれの *window-save-area* データ項目で識別されるウィンドウを閉じることによって選択されます。次に例を示します。

5 つのポップアップウィンドウを、*a*、*b*、*c*、*d*、*e* の順で、作成したと仮定します。

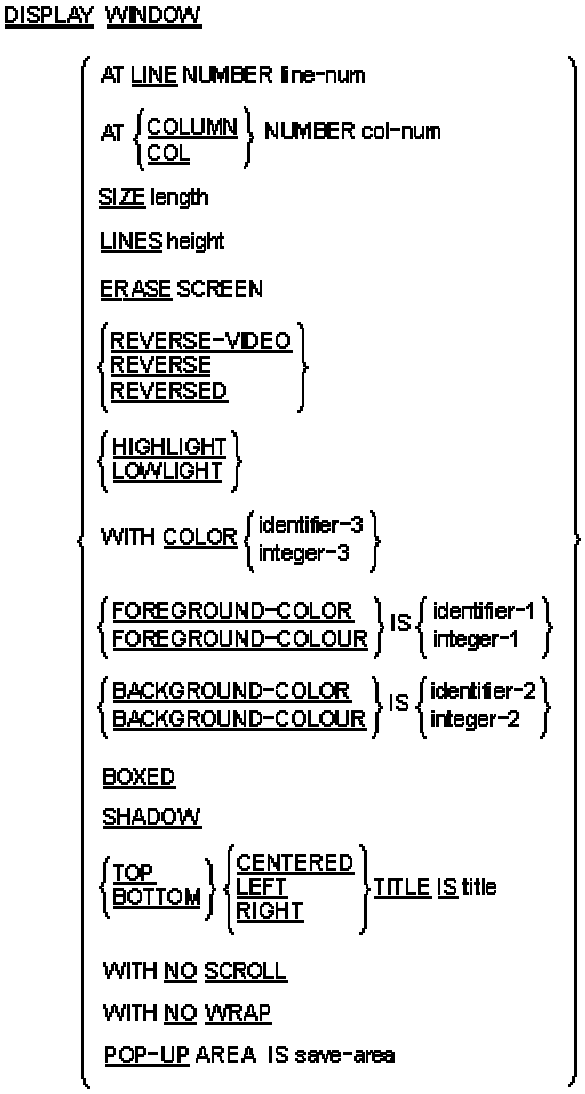
- *d* が閉じられると、*c* が現在のウィンドウになります。
- *b* が閉じられると、*a* が現在のウィンドウになります。
- *e* が閉じられると、*c* が再び現在のウィンドウになります。

8.3.3 DISPLAY 文

ここでは、ウィンドウの作成や線とボックス描画用の 3 つの追加の書き方の DISPLAY 文を提供する COBOL ウィンドウ構文を説明します。 その他の書き方については、『***言語リファレンス***』を参照してください。

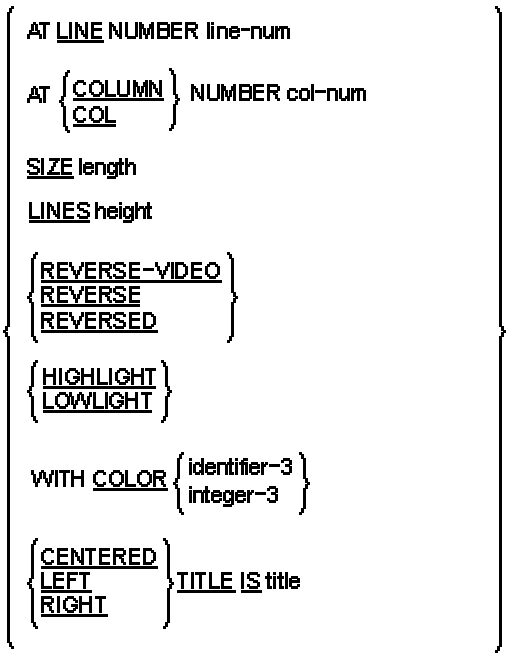
書き方 1

書き方 1


DISPLAY WINDOW
AT LINE NUMBER line-num
AT COLUMN NUMBER col-num
COL
SIZE length
LINES height
ERASE SCREEN
REVERSE-VIDEO
REVERSE
REVERSED
HIGHLIGHT
LOWLIGHT
WITH COLOR identifier-3
integer-3
FOREGROUND-COLOR
FOREGROUND-COLOUR
IS identifier-1
integer-1
BACKGROUND-COLOR
BACKGROUND-COLOUR
IS identifier-2
integer-2
BOXED
SHADOW
TOP
BOTTOM
CENTERED
LEFT
RIGHT
TITLE IS title
WITH NO SCROLL
WITH NO WRAP
POP-UP AREA IS save-area

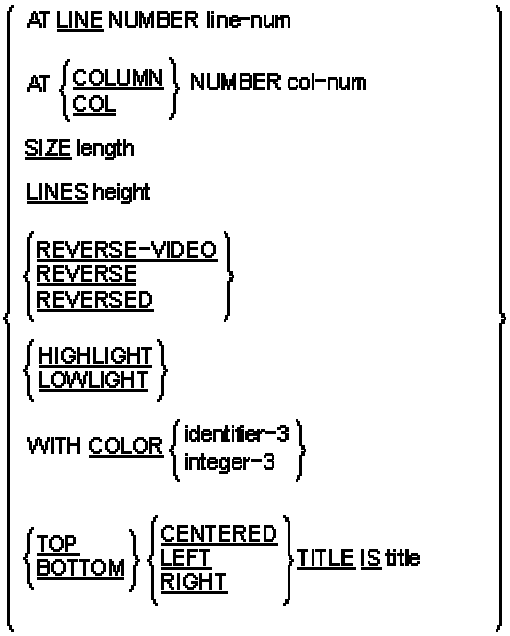
書き方 2

DISPLAY LINE



書き方 3

DISPLAY BOX



構文規則

1. *line-num* は、端末画面上の行位置を指定する、数字定数またはデータ項目です。負以外の整数を指定する必要があります。
2. *col-num* は、端末画面上のカラム位置を指定する、数字定数またはデータ項目です。負以外の整数を指定する必要があります。
3. *length* は、ウィンドウ幅、行幅、または文字位置のボックス幅を指定する、数字定数またはデータ項目です。負以外の整数を指定する必要があります。
4. *height* は、ウィンドウ内の行数を指定する、数字定数またはデータ項目です。負以外の整数を指定する必要があります。
5. *title* は、非数字定数または英数字データ項目です。
6. *save-area* は、PIC X(10) 句で記述される基本データ項目です。
7. COLUMN と COL は等価です。
8. REVERSE、REVERSED、および REVERSE-VIDEO は等価です。
9. COLOR 句は、プリプロセッサ指令 COLOR を使用するときのみサポートされます。これは、既存の Micro Focus 仕様でない構文を追加します。
10. SIZE または LINES 句の 1 つは、DISPLAY 文の書き方 2 に指定する必要があります。
11. *identifier-1*、*identifier-2*、*integer-1*、および *integer-2* は、次のような範囲 0 ～ 7 の値です。

    0　　黒

1	青
2	緑
3	シアン
4	赤
5	マゼンタ
6	茶または黄色
7	白

12. *identifier-3* と *integer-3* は、次の適切な値を加えた値です。

色	前景	背景
黒	1	32
青	2	64
緑	3	96
シアン	4	128
赤	5	160
マゼンタ	6	192
茶	7	224
白	8	256

13.
14. **注**：COLOR 句に使用される前景色の値は、標準の Micro Focus 仕様の FOREGROUND-COLOR および BACKGROUND-COLOR の標準色の値とは異なります。
15.
16. また、ビデオ属性は次の値で指定できます。

反転表示	1024
低輝度	2048
高輝度	4096
下線	8192
点滅	16384

17. COLOR 句が FOREGROUND-COLOR や BACKGROUND-COLOR と同時に使用される場合には、COLOR 句で定義されたカラーは無視されますが、非カラー属性は適切に動作します。

一般規則

すべての書き方

1. LINE と COLUMN 句は、物理画面上の行、カラムを指定する必要があります。
2. カラーシステムの場合は、COLOR や FOREGROUND-COLOR、BACKGROUND-COLOR での指定、他の属性設定 (たとえば、点滅) の両方が使用されます。 モノクロシステムでは、すべてのカラー情報が無視され、他の設定のみが使用されます。
3. すべての属性の組み合わせが、すべてのシステムで使用できるわけではありません。 たとえば、標準 DOS PC では、モノクロモードの設定、REVERSE と UNDERLINE は相互排他的で、これらの属性の 1 つのみ動作します。

書き方 1 (DISPLAY WINDOW)

4. DISPLAY WINDOW 文は、端末ウィンドウを作成し、現在のウィンドウにします。 端末ウィンドウは、画面上の領域です。 ACCEPT や DISPLAY 文 (『*書語リファレンス*』に記述されている他の ACCEPT 書き方 1、2 や 3、DISPLAY 書き方 1、DISPLAY WINDOW/LINE/BOX 文を除く) は、現在のウィンドウのみに影響します。 さらに、すべての ACCEPT や DISPLAY 文 (『*書語リファレンス*』に記述されている他の ACCEPT 書き方 1、2 や 3、DISPLAY 書き方 1、DISPLAY WINDOW/LINE/BOX 文を除く) の行やカラム数は、現在のウィンドウの左上隅から計算されます。 つまり、現在のウィンドウは仮想端末の画面を定義し、物理画面のある領域を占有します。

5. 初期のウィンドウは、全画面に設定されます。
6. 現在のウィンドウを変更する唯一の方法は、他の DISPLAY WINDOW 文や CLOSE WINDOW 文による変更です。

7. LINE NUMBER 句は、ウィンドウの先頭行を設定します。 行番号 1 は画面の先頭行を指します。 行番号は、画面に対する相対位置で、現在のウィンドウに対するものではありません。

8. LINE NUMBER 句が指定されていない場合、0 が指定された場合、または、物理画面外の場合は、画面の先頭行が使用されます。

9. COLUMN NUMBER 句は、ウィンドウの最左のカラムを設定します。 カラム番号 1 は画面の左側を指します。 カラム番号は、画面に対する相対位置で、現在のウィンドウに対するものではありません。

10. COLUMN NUMBER 句が指定されていない場合、0 が指定された場合、または、物理画面外の場合は、カラム番号 1 が使用されます。

11. SIZE 句は、ウィンドウのカラム数を指定します。 このカラム数が画面の右端を越えると、画面の幅は画面外に拡張されます。

12. SIZE 句が指定されていない場合、または、0 が指定された場合には、ウィンドウは画面の右端に拡張されます。

13. LINES 句は、ウィンドウの行数を指定します。この行数が画面の下部を越えると、画面は画面外に拡張されます。

14. LINES 句が指定されていない場合、または、0 が指定された場合には、ウィンドウは画面の下端に拡張されます。

15. ERASE 句が指定されると、ウィンドウの作成時に直ちにクリアされます。それ以外は、ウィンドウの内容は変更されません。ウィンドウをクリアすると、空白文字で埋めます。

16. BOXED 句は、新しいウィンドウの周りにボックスを描画します。ボックスは、ウィンドウの外側に描画されます。画面外のボックスの部分は描画されません。

17. 端末の線描画セットは、ボックスを描くために使用されます。端末に線描画セットがない場合は、同等の ASCII 文字が使用されます。POP-UP 句が指定されると、そのボックスは画面上の他のボックスに重なります。この句が指定されない場合には、ボックス描画は他のボックスに張り付き、交差します。ボックス化された非ポップアップウィンドウがボックス化されたポップアップウィンドウと交差する場合に、そのボックス化されたポップアップウィンドウが最初に作成され、それが閉じられると、2 つのボックスが交差していた部分は再描画されません。つまり、交差している部分の文字は、交差部分がなくなっても、そこに表示されたままです。

18. ERASE 句は、BOXED 句によって暗黙的に含まれます。
19. REVERSED 句は、ウィンドウの前景色と背景色を反転表示します。これは、新しいウィンドウの ACCEPT や DISPLAY 文ごとに影響します。

20. REVERSED 句は ERASE 句によって暗黙的に含まれます。通常、REVERSED 句は、最初に作成したときにウィンドウ全体を反転表示します。

21. SHADOW 句は、ウィンドウに 3 次元的な効果を与え、画面上で浮き上がっているように見せます。
22. 前景や背景の色の値を COLOR フィールドで 0 に設定すると、デフォルトのシステム属性に対応する色が使用されます。
23. TITLE 句は、ウィンドウの境界にタイトルを表示します。TITLE 句は、BOXED 句が指定されている場合のみに効果があります。

24. タイトルは、左上端、上端の中心、右上端、左下端、下端の中心、右下端の 6 つの位置のどれかに表示できます。TOP や BOTTOM が指定されていない場合は、TOP が使用されます。LEFT、CENTERED、または RIGHT が指定されていない場合は、CENTERED が使用されます。

25. NO SCROLL 句は、覚え書きとして扱われます。Windows 用のプリプロセッサは、これを確認するメッセージを表示します。
26. NO SCROLL 句は、単に覚え書きとして扱われます。Windows 用のプリプロセッサは、これを確認するメッセージを表示します。
27. POP-UP AREA 句は、COBOL システムに、新しいウィンドウを作成する前にシステム情報を格納します。この情報は、新しいウィンドウを削除し、基礎となるウィンドウを

復元するために CLOSE WINDOW 文で使用されます。これで、ポップアップウィンドウが使用可能になります。

28. *save-area* データ項目には、システム情報が格納されます。このデータ項目を、格納後に変更しないでください。変更すると、結果が未定義になります。このデータは、CLOSE WINDOW 文で以前のウィンドウを画面に復元し、そのウィンドウを現在のウィンドウに再設定します。

書き方 2 (DISPLAY LINE)

29. DISPLAY LINE 文は、マシンや端末に依存する方法で、水平や垂直の線を描画します。線は、ディスプレイデバイスの最適なモードを使用して描画されます。DISPLAY BOX 文とともに使用すると、画面にフォームを描画できます。DISPLAY LINE 文は、全画面用の ACCEPT や DISPLAY 文の位置付けには影響しません。

30. 線は、画面上で他の線と交差するように表示され、適切な交差用の文字が使用されます。線の末尾が他の直線と交差するように、隅や 3 方向の交差が使用されます。

31. SIZE 句が指定されると、線は水平に描画されます。*length* の値は、線の長さを画面のカラム数で示します。かわりに、LINES 句が使用されると、線は垂直に描画され、*height* の値は画面行数で示されます。

32. 線はラップしたり、スクロールしたりしません。LINES や SIZE 句で描画した線が現在のウィンドウを越えるような場合には、その線はウィンドウの端で切り捨てられます。LINES や SIZE が 0 の場合は、線は描画されません。

33. *line-num* の値は、線の開始行を指定します。*col-num* は、開始カラムを指定します。常に、線は適切に右方向、下方向に描画されます。*line-num* と *col-num* には、現在のウィンドウ内の位置を指定する必要があります。

34. LINE NUMBER や COLUMN NUMBER 句が物理画面の外側を指定している場合、つまり、*line-num* が 0 か 24 (または画面の最大値)、*col-num* が 0 か 80 より大きい場合には、線は描画されません。

35. TITLE 句は、水平線のみに効果があります。TITLE 句が指定されると、*title-string* が線上に表示されます。

36. タイトルは、RIGHT、LEFT、CENTERED 指定に応じて、右端、左端、中央に表示されます。省略時は、CENTERED が使用されます。

37. REVERSE 句は、線の前景と背景を反転します。

書き方 3 (DISPLAY BOX)

38. DISPLAY BOX 文は、マシンや端末に依存する方法で、ボックスを描画します。ボックスは、ディスプレイデバイスの最適なモードを使用して描画されます。ボックスの描画に使用されている線が画面上の他の線と交差する場合は、適切な交差用の文字

が使用されます。DISPLAY BOX 文は、全画面用の ACCEPT や DISPLAY 文の位置付けには影響しません。

39. ボックスの位置は、左上隅で指定されます。ボックスのサイズは、高さと幅で指定されます。

40. LINE NUMBER や COLUMN NUMBER 句が物理画面外を指定すると、ボックスは描画されません。

41. SIZE 句は、ボックスの幅を指定します。LINES 句は、その高さを指定します。SIZE 句が指定されなかったり、0 であったり、ボックスが物理画面やウィンドウの端を越えて拡大されたりした場合には、ボックスは現在のウィンドウの右端まで拡張されます。LINES 句が指定されなかったり、0 であったり、ボックスが物理画面を越えて拡大されたりした場合には、ボックスは現在のウィンドウの下端まで拡張されます。

42. REVERSE 句は、DISPLAY WINDOW 文の場合と同じように動作します。

43. TITLE 句も DISPLAY WINDOW 文の場合と同じように動作します。

8.4 ウィンドウ構文の制約

- この機能は、中間コードの互換性が保証されていませんので、製品リリース時にソースコードを再コンパイルする必要があります。
- ACCEPT や DISPLAY 文をこのウィンドウ構文で使用する場合には、AT LINE NUMBER 構文 (『*言語リファレンス*』を参照) やウィンドウに出現しない項目を含める必要があります。
- COPY REPLACING や REPLACE 文は使用できません。
- ウィンドウ構文は、固定形式の COBOL 原始プログラムのみサポートされます。
- 次の予約語は、ウィンドウ構文に取り入れられるので、それらを利用者語として使用することを避けてください。

  BOX
  BOXED
  CENTERED
  COLOR (COLOR 指令が使用される場合)
  POP-UP
  SCROLL
  SHADOW
  WINDOW
  WRAP

- ウィンドウ構文では、『*言語リファレンス*』に記載されている ACCEPT と DISPLAY 文のみを使用してください。
- ウィンドウ構文を使用する場合は、ANS85 コンパイラ指令が前提になります。この指令は、明示的にも、暗黙的にも設定解除してはいけません。この指令には、構文や

RM、MS、および DG などの他の言語の動作に影響する指令のセットが含まれています。

- 英数字定数は、ウィンドウの文を含む行の末尾を越えて続けてはなりません。
- PROCEDURE DIVISION のスペルミスなどでは、ある種の構文エラーが表示されますが、それ以降のソースの行に対する余分なエラーメッセージが表示されることがあります。
- ウィンドウ構文エラーは、重大なエラーであり、次のように表示されます。

  xnnn-P*******

- カラム 73 は、このカラムが常に空白文字として扱われるので、原始プログラム中に使用できません。
- エラーが発生すると、コンパイラは処理を継続するかどうかを尋ねます。NOERRQ 指令で、この機能を無効にできます。UNIX オペレーティングシステムでは、コンパイル時に、NOERRQ 指令を使用することはできません。

  エラーがない場合や、エラーであっても「続行しますか」に「いいえ」で答えると、コンパイラは 0 のエラー戻りコードを返します。

- 次の文は、単独で 1 行で記述する必要があります。

  DISPLAY WINDOW
  DISPLAY BOX
  DISPLAY LINE
  CLOSE WINDOW
  EXIT PROGRAM

- ウィンドウサブシステムは、最初のウィンドウの文が実行されるときに、自動的に初期化されます。
- アプリケーションがウィンドウ構文と他の ACCEPT/DISPLAY 構文とを切り替えて使用している場合は、ウィンドウ構文による処理を完全に終了した後に、他の ACCEPT/DISPLAY 構文の使用を開始する必要があります。それ以外の場合には、ACCEPT/DISPLAY 文は希望する効果が得られません。

  次の副プログラムをコンパイルして、ウィンドウ構文による処理を明示的に終了するサブルーチンを作成できます。

```
$set preprocess "window1" autoclose
procedure division.
para-1.
    exit program.
```

  他の ACCEPT/DISPLAY 構文に切り替える前に、この副プログラムを呼び出せるようになります。AUTOCLOSE プリプロセッサ指令は、副プログラムを終了する前に EXIT PROGRAM 文にウィンドウ構文による処理を終了させます。ウィンドウサブシス

テムは、他のウィンドウ文が見つかると、再初期化されます。ウィンドウサブシステムの初期化のたびに、背景画面と内容が再表示されます。

- 同一の実行単位で Panels の呼び出しとウィンドウ構文を混在することはできません。
- ウィンドウがアクティブ、またはその実行単位でアクティブであった場合には、DISPLAY SPACES UPON CRT 文はウィンドウを空白文字でクリアしますが、属性は変更しません。
- COPYEXT、COPYLBR、FOLD-COPY-NAME などのコピーファイルに影響する指令は使用できません。
- COBCPY などのコピーファイル処理に影響する環境変数は使用できません。
- 次の構文があります。

  ```
  COPY filename OF library
  ```

  この構文は、サポートされません。

8.5 ウィンドウ構文のエラーメッセージ

次のエラーメッセージがプリプロセッサの処理中に表示されます。.

Unexpected numeric literal:

Unexpected numeric literal:

Unsupported keyword or noiseword:

Unrecognized clause to DISPLAY WINDOW:

Unrecognized clause to DISPLAY WINDOW:

Unrecognized clause to DISPLAY WINDOW:

Unrecognized clause to ACCEPT FROM SCREEN:

This keyword has already been used:

This keyword conflicts with another:

This reserved word is used incorrectly:

Wrongly formed or ordered clause with keyword:

Error during preprocessing - no further details

Unknown COPY file specified

WINDOW1 preprocessor cannot handle free format

SCROLL/WRAP clause processed as comment

編集 / コンパイル / アニメートのループは、エラーが返された後に、原始プログラムの正しくない行に戻ります。

8.6 ウィンドウ構文の補足情報

プログラム内の最初のウィンドウ文が出現すると、画面が再表示されます。これは予期された動作で、プログラムへの影響はありません。

8.7 ウィンドウ構文の例

```
$set preprocess(window1)
 working-storage section.
 78 note-height  value 16.
 78 note-height  value 41.
 78 no-of-chars  value note-height * note-width.
 01 note-window  pic x(10).
 01 dummy    pic x.
 01 note-data value all "- wallpaper ".
   03 note-char pic x occurs no-of-chars.
 screen section.
 01 input-data highlight.
   03 line 4 column 6 value " Accept and Display positions ".
   03 line 5 column 6 value " are relative to the top left ".
   03 line 6 column 6 value " corner of the window. ".
   03    pic x using dummy.
 01 note-screen  pic x(no-of-chars)
              using note-data prompt " " reverse-video.
 procedure division.
* 画面に境界とタイトルを付けて、空白のウィンドウを表示します。
      display window, line 2, column 38, lines note-height,
          size note-width, boxed, erase, reverse
* このウィンドウへの参照を定義し、
* 削除や以前の表示の復元を可能にします。
          pop-up area is note-window
          bottom right title "Press Enter to remove window"
* note-screen の内容でウィンドウを埋めます。
    display note-screen
      display input-data
      accept input-data
      close window note-window.
```

第 9 章：キャラクタインターフェイス用のマウスサポート

多くの場合は、マウスの存在を検出することで文字ベースのアプリケーションのインターフェイスを拡張し、マウスをオプションで使用できます。 COBOL は、マウスのアクセスに対する上位レベルと下位レベルのルーチンを提供します。 マウスの全機能を活用する高度な新しいプログラムを記述できます。 既存のアプリケーションをマウスが使用できるように変更することも容易にできます。

『*既存アプリケーションにマウスサポートを追加*』で、既存アプリケーションにマウスサポートを組み込む方法を説明します。『*名前による呼び出しマウスルーチンの使用方法*』では、マウス機能のすべてを使用する方法を説明します。

9.1 既存アプリケーションにマウスサポートを追加

多くの既存の文字ベース PC COBOL アプリケーションは、COBOL の画面やキーボード管理機能を改善する、拡張 ACCEPT/DISPLAY 構文 (Adis) を使用します。 プログラムに画面節があり、画面の位置やハイライト表示を制御する ACCEPT/DISPLAY 文の拡張を使用しているか、または、x"AF" ルーチンを呼び出したりしていると、それは Adis を使用していることになります。 Adis の詳細については、『*拡張 ACCEPT/DISPLAY 構文*』の章を参照してください。

マウスを使用して、このようなアプリケーションを拡張すると、次のような実行ができます。

- カーソルをすばやく他の項目に移動する。
- マウスボタンを **Enter** キーやファンクションキーとして使用する。
- ポイントアンドピックのメニュー選択をサポートする。

これらのマウスルーチンで、アプリケーションのユーザインターフェイスは、マウスなしでも以前のように動作します。 ただし、マウスを装備している場合は、多くの操作が簡単になり、短縮できます。

既存のアプリケーションにわずかなコードを含めることで、これらの機能を追加できます。 x"AF" ルーチンを特定のファンクションコードで呼び出すことによって、Adis マウスルーチンにアクセスします。これらのファンクションコードで、次のようなことが可能になります。

- マウスドライバの起動。
- マウスの表示や非表示。
- マウスポインタの位置の返却。
- マウスドライバの終了。

番号による呼び出し x"AF" ルーチンを、同じプログラム内で名前による呼び出しと混在することはできません。

admouse.cbl サンプルプログラムに、これらのルーチンと手法を説明しています。このプログラムは、Net Express\base\demo\admouseディレクトリに提供されています。

9.1.1 マウスの初期化と終了

マウスの機能にアクセスする前に、マウスの初期化が必要です。初期化の呼び出しは、Adisにマウスドライバをロードし、マウスポインタを描画するように通知します。アプリケーションがマウスを使用し終わると、終了処理が必要です。終了処理では、マウスドライバを解除し、マウスポインタを削除します。

次のコードは、マウスの初期化と終了の方法を示しています。

```
01  use-mouse-function     pic x comp-x value h"40".
01  terminate-mouse        pic x comp-x value h"00".
01  activate-mouse         pic x comp-x value h"01".
    ...
    call x"AF" using use-mouse-function
                     activate-mouse
```

または

```
    call x"AF" using use-mouse-function
                     terminate-mouse
```

詳細は、次のとおりです。

h"*nn*"	16 進数の定数で、数字フィールドのバイナリ値を 16 進数で表示する Micro Focus 拡張です。
h"40"	10 進数 64 の 1 バイト comp フィールドと等価です。

名前付き定数については、『*言語リファレンス*』を参照してください。

Adis は、マウスが初期化できなかった場合には、*use-mouse-function* に値 255 を返します。

9.1.2 マウスの使用

マウスポインタが表示されると、ユーザは画面上のフィールドをポイントできます。デフォルトの設定では、マウスの第 1 ボタン (通常、左ボタン) を押すと、マウスの指したフィールドの先頭にカーソルが置かれます。マウスポインタが有効な入力フィールド上にない場合、Adis は警告ビープ音を鳴らします。マウスポインタが隠れている場合は、ボタンが押されても無視されます。

9.1.3 ポイントアンドピックメニューの作成

メニューから項目を選択するためにマウスを使用するには、プログラムで、マウスボタンが押されたことを検出し、その時点の画面上のマウスの位置を決定することが必要です。Adis はマウスボタン 1 を検出しますが、他のボタンは無視されます。マウスボタンが押されると、Adis は Adis キー 27 として解釈します。

デフォルトでは、**Enter** (Adis キー 0) キー以外の Adis キーは、ACCEPT 文の実行を終了できません。これを変更するには、次のようなコードを使用します。

次のコードは、マウスで ACCEPT 文の実行を終了させる方法を示しています。

```
 01  set-bit-pairs          pic x comp-x value h"01".
 01  mouse-can-terminate-accept
                            pic x(4) comp-x value h"01321b01".
 01  mouse-cannot-terminate-accept
                            pic x(4) comp-x value h"02321b01".
     ...
     call x"AF" using set-bit-pairs
                      mouse-can-terminate-accept
```

または

```
    call x"AF" using set-bit-pairs
                     mouse-cannot-terminate-accept
```

マウスボタンで ACCEPT 文の実行を終了させるようにすると、2 番目の **Enter** キーが使用できるようになります。この動作は、アプリケーションのダイアログで常に適切であるとは限りません。

ACCEPT 文を終了させたのがマウスボタンであることを判断するために、特殊名段落の CRT-STATUS データ項目を調べる必要があります。CRT-STATUS データ項目は、次のように定義されています。

```
 01  key-pressed.
     05  key-type        pic x.
     05  key1            pic x comp-x.
     05  keyx            redefines key1 pic x.
     05  key2            pic x comp-x.
```

ACCEPT 文が **Enter** キーで終了させられると、*key-type* データ項目は値 0 になります。マウスで ACCEPT 文が終了させられると、*key-type* は 2 となり *key1* は値 27 になります。

例

キータイプ (key-type) を判断するコード例です。

```
     display screen
```

```
     accept screen
     evaluate key-type also key1
       when "0" also any
*         <        Enter キーの処理>
       when "2" also 27
*         <          マウスボタンの処理>
       when "1" also 1
*         <          ファンクションキー 1 の処理>
           ...
     end-evaluate
```

マウスが押されたと判断されると、次の呼び出しでマウスポインタの位置を問い合わせできます。

```
 01  get-mouse-details      pic x comp-x value h"43".
 01  mouse-details.
     05  mouse-x            pic 9(4) comp-x.
     05  mouse-y            pic 9(4) comp-x.
     05  mouse-status       pic 9(4) comp-x.
      ...
     call x"AF" using get-mouse-details
                      mouse-details
```

マウス座標は、画面の左上隅を (0,0) と仮定します。 座標をメニュー選択に変換するには、向き (水平または垂直)、画面上のメニューの相対位置、またはメニュー項目の長さに依存します。 *mouse-status* フィールドは、最後にマウスボタンが押されたときに、どのマウスボタンが押されたかを示します。

9.2 名前による呼び出しマウスルーチンの使用方法

『*既存アプリケーションにマウスサポートを追加*』では、マウスを使用するために Adis の ACCEPT や DISPLAY 文を含んだ既存アプリケーションをすばやく変更する方法を説明しています。 このサポートで、次のことが可能になります。

- マウスの表示と非表示。
- マウスクリック (追加のファンクションキーとして) の検出。
- マウスポインタの位置の決定。

ただし、このサポートは、マウスのすべての機能が使用可能ではありません。 名前による呼び出しルーチンのセットを使用すると、マウス機能のすべてにアクセスできます。

番号による呼び出し x"AF" ルーチンを、同じプログラム内で名前による呼び出しと混在することはできません。

ACCEPT 文に Micro Focus 拡張を使用している場合には、この名前による呼び出しルーチンを使用できません。ただし、上位レベルのルーチンでは可能でないいくつかのマウス機能へのアクセスを提供しています。これには、次のようなものが含まれます。

- マウスの存在とマウスボタン数の検出。
- すべてのマウスボタンの使用。
- マウスポインタ位置のリアルタイムトレース。
- ボタンを押す、ボタンを放す、マウスを移動するなどのすべてのマウスイベントの検出。
- マウスポインタの特性の問い合わせと変更。

ここでは、これらの新しいルーチンとその手法を説明します。

サンプルプログラム **rtsmouse.cbl** では、そのルーチンと手法について説明しています。このプログラムは、**Net Express\base\demo\rtsmouse**ディレクトリに提供されています。次の内容を検討して、プログラムをコンパイルし、実行できます。

9.2.1 マウスの存在とマウスボタンの検出

Adis マウス機能で、初期化と終了処理が必要です。

次のコードは、呼び出しの初期化と終了の方法を示しています。

```
01  mouse-handle         pic x(4) comp-x.
01  mouse-buttons        pic x(2) comp-x.
          ...
    call "CBL_INIT_MOUSE" using mouse-handle
                                mouse-buttons
```

または

```
    call "CBL_TERM_MOUSE" using mouse-handle
```

マウスの初期化呼び出しは、マウスが存在すると、COBOL の特殊レジスタ RETURN-CODE を 0 に設定します。そして、マウスハンドルとマウスボタン数の両方を返します。マウスハンドルは、すべてのマウスルーチンの第 1 パラメータとして必要です。RETURN-CODE の値が 0 以外である場合は、マウスが存在せず、マウスハンドルが無効なことを示します。この無効なマウスハンドルをマウスルーチンに渡すと、マシン環境によってはロックするなどの障害が発生します。

9.2.2 マウスイベントキュー

ユーザがマウスポインタを移動したり、ボタンを押したり、放したりするたびに、マウスイベントが生成されます。マウスイベントは非同期で、独立に、他の動作が行われていても発生します。たとえば、プログラムがファイル更新の途中でも、ユーザはマウスポインタを移動できます。アプリケーションが重要なマウスイベントを取りこぼさないように、イベントを取得するまで

キューに格納されます。キューからイベントが読み込まれると、イベントはキューから取り除かれます。

9.2.2.1 マウスイベントキューの問い合わせ

マウスイベントキューの状態の問い合わせ

次のコードは、マウスイベントキュー内に待機しているイベント数を判断します。

```
01  queued-events       pic x(2) comp-x.
    ...
    call "CBL_GET_MOUSE_STATUS" using mouse-handle
                                      queued-events
```

マウスイベントの読み込み

次の 2 つの方法でキューから読み込みます。

方法	説明
待機読み込み	キューにイベントがある場合は、次のイベントを返します。イベントキューにイベントがない場合には、アプリケーションは、マウスイベントが発生するまで中断され、イベントが発生するとイベントが返されます。
非待機読み込み	キューにイベントがある場合は、次のイベントを返します。キューにイベントがない場合には、*event-data* にすべて 0 を設定して返されます。

次のコードは、マウスイベントをキューから取り込む方法を示します。

```
01  read-nowait             pic x comp-x value h"00".
01  read-wait               pic x comp-x value h"01".
01  event-data.
    05  event-type          pic x(2) comp-x.
    05  event-time          pic x(4) comp-x.
    05  event-row           pic x(2) comp-x.
    05  event-col           pic x(2) comp-x.

    call "CBL_READ_MOUSE_EVENT" using mouse-handle
                                      event-data
                                      read-nowait
```

または

```
    call "CBL_READ_MOUSE_EVENT" using mouse-handle
```

event-data

read-wait

詳細は、次のとおりです。

event-row イベントが発生した時点のマウスポインタの行です。最初の行は行 0 です。

event-col イベントが発生した時点のマウスポインタのカラムです。最初のカラムはカラム 0 です。

event-time 任意の開始時間からの経過時間です。イベント時間は、2 つのマウスイベント間の経過時間を知るのに便利です。たとえば、マウスボタンのシングルクリックとダブルクリックをサポートすることがあります。イベント時間が 0 の場合は、イベントがないことを示します (非待機読み込みが使用されていることを意味します)。

event-type 動作の種類を示します。

各イベントタイプに異なるビットが、このフィールドに設定されます。ビットの組み合わせで、発生したイベントを識別できます。次の表は、イベントの動作と対応するビット位置を示しています。ボタン 1 は、通常マウスの左ボタンで、ボタン 2 が右です。

ビット	値	イベント動作
15-4	16-255	予約済み。
3	8+	ボタン 3 を押した。
2	4	ボタン 2 を押した。
1	2	ボタン 1 を押した。
0	1	マウスが移動した。

同時に複数のイベントが発生できるため、*event-type* の個々のビットを検査する必要があります。詳細については、『*ビットの値を検査*』を参照してください。

9.2.2.2 ビットの値を検査

次のコードは、COBOL でバイト内の単一ビットを検査するビット操作ルーチンの使用例です。

```
78  mouse-moved              value 1.
78  button1-pressed          value 2.
78  button2-pressed          value 4.
78  button3-pressed          value 8.
01  event-data.
    05  event-type           pic x(2) comp-x.
```

```
    05  event-time             pic x(4) comp-x.
    05  event-row              pic x(2) comp-x.
    05  event-co               pic x(2) comp-x.
01  result                     pic x(2) comp-x.
01  button1-status             pic x(8).
        ...
    call "CBL_READ_MOUSE_EVENT" using mouse-handle
                                      event-data
                                      read-nowait
    move button1-pressed to result
    call "CBL_AND" using event-type
                         result by value 2
    if result = button1-pressed
        ...
```

CBL_AND ルーチンは、検査されるビット (この場合は *button1-pressed*) が *event-type* に設定されていない限り、*result* に 0 を設定します。

9.2.2.3 マウスクリックの検査

CBL_READ_MOUSE_EVENT を呼び出した後では、*event-type* の値 0 は 1 つ以上のマウスボタンが放されたことを示しています。

クリックをサポートする (マウスボタンによる選択を示す慣習的な方法で、ボタンを押して放す方法) には、各ボタンの前の状態を保存する必要があります。クリックは、マウスボタンを放すと完了します。

次のコードは、マウスボタンが押されたか、または、放されたかを検査する方法を示しています。

```
       ...
    if result = button1-pressed
        move "pressed" to button1-status
    else
        if button1-status = "pressed"
            move "released" to button1-status
        else
            move space      to button1-status
        end-if
    end-if
       ...
```

ボタンを押した後でマウスポインタを動かせるため、クリックが完了した後にマウスポインタ位置を検査する必要があります。次のコードで、その方法を示します。

```
    ...
  if button1-status = "released"
      evaluate  event-row also event-col
        when 24 also 10 thru 14
            ...
        when 24 also 18 thru 22
            ...
     end-evaluate
  end-if
 ...
```

9.2.2.4 マウスイベントマスク

イベントマスクは、どのような種類のイベントがキューされているかを指定するのに使用されます。このマスクは、CBL_READ_MOUSE_EVENT の呼び出しで返される *event-type* と同じビット設定です。イベントマスクのビットを 0 に設定すると、指定されたイベントのオカレンスがイベントキューに格納されません。デフォルトでは、すべてのイベントがイベントキューに格納されます。

イベントマスクの設定

次のコードは、イベントマスクを設定し検査する方法を示しています。

```
 01  event-mask       pic x(2) comp-x.
     ...
  call "CBL_SET_MOUSE_MASK" using mouse-handle
                                  event-mask
```

または

```
  call "CBL_GET_MOUSE_MASK" using mouse-handle
                                  event-mask
```

CBL_SET_MOUSE_MASK ルーチンは、全画面セッションの実行時のみに使用できます。ウィンドウセッションの実行時では、予測できない結果になります。

ビット値の設定

イベントマスクのビットを設定するときは、他のビット設定に影響しない方法でビット値を変更する必要があります。この方法を、次のコードに示します。

```
 01 source            pic x(2) comp-x.
    ...
  move button1-pressed to source
  call "CBL_OR" using source
                      event-mask by value 2
```

9.2.3 参考

マウス呼び出しの詳細については、『*COBOL システムライブラリルーチン*』の章を参照してください。

第 10 章 : オンラインヘルプの概要

ここでは、オンラインヘルプシステムについて説明します。

オンラインヘルプシステムは、文字ベースのアプリケーションからオンラインヘルプを作成し、画面上に表示する高度なシステムです。このシステムには、利用可能な情報間の拡張ナビゲーション機能も含まれています。

ネイティブな Windows 形式のヘルプシステム、または、Micro Focus 形式のヘルプシステムのどちらを使用するかを選択できます。

10.1 概要

オンラインヘルプシステムは、アプリケーション利用者のために、情報を含んでいるフォーマットされたテキストファイルを表示します。システムには 2 つの部分があります。

- オンラインヘルプビルダ (Ohbld)

  Ohbld を使用して、プログラムのヘルプ情報を含む特別にフォーマットされたテキストファイルを、オンラインヘルプビューアで*トピック*にアクセス可能な圧縮されたオンラインヘルプファイルに変換します (トピックの定義については、『*オンラインヘルプの用語集*』を参照してください)。

- オンラインヘルプビューア (Hyhelp)

  オンラインヘルプビューアは、使用可能なオンラインヘルプファイルからトピックを表示し、ヘルプファイル間の拡張ナビゲーションが可能です。このハイパーテキスト機能が、プログラム名を Hyhelp と呼ぶ理由です。Hyhelp はファンクションキーインターフェイスを使用し、文字ベースのアプリケーションで使用できます。

トピックは、ユーザが選択し、実行できる他のトピックのクロスリファレンスを含んでいます。索引、目次、ブックマーク、および履歴リストは、付加的なナビゲーション機能を提供し、ユーザが必要とする情報にすばやくにアクセスできます。また、ユーザは、情報を自由に参照することもできます。選択されたトピックやその一部は、印刷したり、テキストファイルとして保存したりできます。

トピックは横にスクロールしないで、異なるサイズ (アプリケーションの選択による) のウィンドウに表示できます。

10.2 構成

『*オンラインヘルプビューア (HyHelp)*』の章は、オンラインヘルプビューアの概要、その機能、起動方法、および Net Express のヘルプの見方について説明しています。

『*オンラインヘルプシステム*』の章は、オンラインヘルプビルダとオンラインヘルプビューアの使用方法を説明しています。

『*オンラインヘルプのソースファイル形式*』の章は、オンラインヘルプのソースファイルのテキスト形式、および、使用できるタグや属性の概要について説明しています。

『*オンラインヘルプビルダ用の指令*』の章は、オンラインヘルプビルダの動作を制御する指令について一覧表示し、各指令を説明しています。

『*構成オプションタグ*』の章は、オンラインヘルプビューアの動作を構成するタグについて一覧表示し、各タグを説明しています。

『*オンラインヘルプタグ*』の章は、オンラインヘルプのソースファイルで使用できるタグと属性について説明しています。

『*Ohbld エラーメッセージ*』の章は、オンラインヘルプビルダで出力される致命的なエラー、エラー、および警告メッセージについて一覧表示し、各メッセージを説明しています。

『*オンラインヘルプの用語集*』の章は、聞き慣れない用語の辞書として使用できます。

第 11 章 : オンラインヘルプビューア (HyHelp)

オンラインヘルプは、文字ベースのアプリケーションで使用するために提供されます。 オンラインヘルプビューア (HyHelp) は、アプリケーションで文字ベースのオンラインヘルプを表示するツールです。

11.1 概要

HyHelp は、オンラインでヘルプファイルを表示できます。 HyHelp を使用し、ヘルプファイルにアクセスします。画面に表示、参照、テキストファイルにコピー、または選択した情報の一部を印刷できます。

ここでは HyHelp の機能、起動の方法、およびヘルプの見方を説明します。

11.2 操作方法

HyHelp の呼び出しの詳細については、『*オンラインヘルプシステム*』の章にある『*オンラインヘルプビューアの起動方法*』を参照してください。

11.2.1 HyHelp の使用方法

HyHelp を起動すると、ここの説明が役に立ちます。 HyHelp に関連付けられたユーザインターフェイスや各種の用語を説明します。

HyHelp を起動すると、選択されたトピックのテキストをもった境界付きのパネルが表示されます。図 11-1 は、HyHelp が起動されたときの HyHelp メインメニューを示しています。トピックのタイトルが上の境界に表示されます。

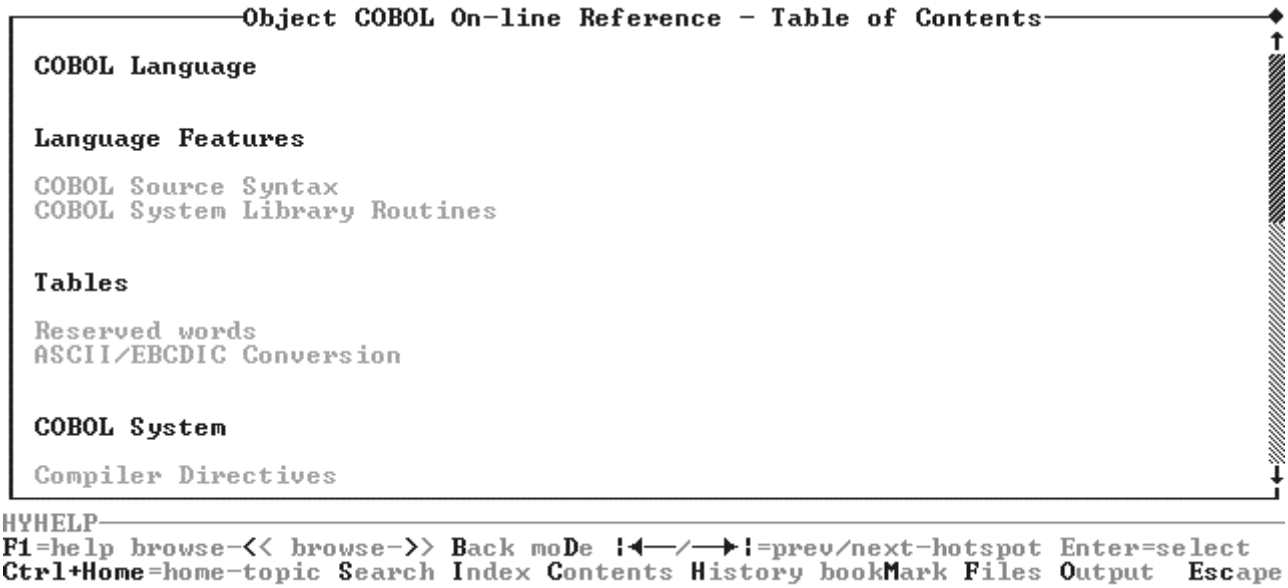


図 11-1：HyHelp メインメニュー

トピックがパネルに一度に表示できない場合は、境界の右側にスクロールバーが表示されます。スクロールバーは、テキスト部分を表示し、トピック内の現在のパネル位置を示します。

↑ キーと ↓ キーを使用して、行を上下に移動したり、**Page Up** キーと **Page Down** キーでパネルごと上下したりできます。

マウスを使用すると、スクロールバー上をクリックして、テキストのパネルを上下できます。また、上下カーソルアイコンをクリックして、行単位でトピックを上下できます。

トピックによっては、凍結行があります。この凍結行は、テキストの残りの部分がスクロールしても、画面の上部に表示されたままです。

HyHelp では、キーボードやマウスを使用して、いろいろな操作ができます。メニュー上に示されたすべての機能は、適切なキーを押すことによってアクティブ化されます。他のいくつかの機能は、カーソル制御キーやマウスを使用して実行できます。 詳細については、後述します。

次の説明では、マウスのクリックやダブルクリックは、関連する箇所でのマウスカーソルの操作を示し、マウスの左ボタンを 1 回クリックしたり、2 回クリックすることを意味しています。マウスポインタは、マウスが接続されているときのみ表示されます。最初に HyHelp を起動すると、パネル境界の右下に表示されます。

11.2.1.1 選択とアクティブ化

HyHelp では、多くのオブジェクトを選択またはアクティブ化できます。オブジェクトを単に選択すると、現在のオブジェクトになります。通常、選択されると、ハイライトされます。選択後のキーストロークは、何のオブジェクトが選択されているかを考慮に入れます。

オブジェクトがアクティブ化すると、そのオブジェクトに関係した処理を実行します。

たとえば、ホットスポット (後述) を選択するとハイライトされ、それを (選択後に **Enter** キーを押して) アクティブ化すると、ホットスポットでクロスリファレンスされているトピックが表示されます。

11.2.1.2 トピックとファイル

HyHelp でアクセス可能な情報は、ヘルプファイルに保持されています。このファイルからのトピックは、使用可能な機能で表示されます。

一般に、各ファイルは自己格納式で、他のファイルをクロスリファレンスしません。ただし、複数のファイルで情報の完全セットを構成することもあります。この場合は、意識することなく 1 つのファイルから他のファイルへと切り替えることができます。あるファイルのトピックが表示されていると、そのファイルが現在のファイルになります。

各ファイルは、トピックのセットを含んでいます。トピックは、通常、ファイルに論理的な順序で格納されています。多くのファイルは、階層的な構造です。一般に、情報を検索する上で、ファイルの構造を理解する必要はありません。ただし、理解していると役に立つことがあります。

ファイルの最初のトピックは、ホームトピックと呼ばれます。ホームトピックは、ファイルに含まれる情報の開始点です。また、ほとんどのファイルは、目次リストと索引を含んでいます。

いくつかのファイルは、リストトピックを含んでいます。これらのトピックには、フルカーソルバーがあります。リストトピックの各行はホットスポットのように、他のトピックへのクロスリファレンスができます。矢印キーやクリックで、カーソルバーを移動して行を選択できます。**Enter** キーを押して、選択したリスト行をアクティブにします。任意の行の上でマウスをダブルクリックすると、その行を選択し、アクティブ化できます。

11.2.1.3 トピック名

トピック名は、最大 30 文字までの ASCII 文字列で、そのトピックを識別するのに使用されます。複数のトピック名で、1 つのトピックを指すこともできます。トピック名は、コマンド行や検索機能からトピックを参照するために使用されます。詳細については、『*オンラインヘルプシステム*』の章を参照してください。

同一のトピック名は、複数のファイル内に存在できます。次のように、情報を表示するときに、希望するトピック名も含むファイルに感嘆符 (!) をトピック名の前に付けることができます。

[*file-name*!] [*topic-name*]

file-name が指定されると、そのファイルのみが検索されます。指定されない場合は、ファイルリストの順序で使用可能なすべてのファイルが検索されます (FILE 構成オプションタグを参照)。 *topic-name* が指定されると、その名前に対応したトピックが検索されます。指定されない場合は、指定されたファイルの最初のトピック (ホームトピック) が表示されます。

たとえば、HyHelp を次のコマンドで起動します。

```
hyhelp test1.hnf!stop
```

HyHelp を起動し、ヘルプファイル **test1.hnf** 内の stop というトピックに関する情報を表示します。

トピック名は、トピックの表示で画面の上部に表示されるトピックタイトルと同じではありません。

11.2.1.4 ホットスポット

トピックには、他のトピックをクロスリファレンスする領域があります。これらの領域は、ホットスポットと呼ばれます。ホットスポットは、通常、その存在を示すために、色が付けられたり、角かっこ (< と >) で囲まれたりします。ホットスポットの上をクリック、または、**Tab** や **Back Tab** キーで、ホットスポットを選択します。選択されたホットスポットは、ハイライトされます。

ホットスポット上でダブルクリックしたり、ホットスポットが選択されたときに **Enter** キーを押したりすると、システムはホットスポットによってトピックのクロスリファレンスを表示します。

トピックの他の語句上でダブルクリックすると、システムはその名前をもつトピックをヘルプファイルで検索します。そのトピックが見つかると、表示されます。

11.2.1.5 ヘルプファイルのナビゲート

ここでは、キーやマウス操作でアクティブにできる付加的な機能を一覧表示します。

キー	説明
Esc	現在のメニューを終了します。
F1=help	現在のメニューのヘルプを表示します。
Enter	選択されたホットスポットやリストの項目をアクティブにします。
↑	パネル内のトピックテキストを 1 行分下に移動したり、リストトピックのカーソルバーを 1 行分上に移動したりします。
↓	パネル内のトピックテキストを 1 行分上に移動したり、リストトピックのカーソルバーを 1 行分下に移動したりします。
Page Up	トピックテキストを 1 パネル分上に動かします。
Page Down	トピックテキストを 1 パネル分下に動かします。
Home	現在のトピックの最初に移動します。
End	現在のトピックの最後に移動します。
Tab	次のホットスポットを選択するか、または、何も選択されていなければ最初のホットスポットが選択されます。
Back Tab	前のホットスポットを選択するか、または、何も選択されていなければ

	最後のホットスポットが選択されます。
Ctrl+Home	現在のヘルプファイルのホームトピックを表示します。

次のようなマウス操作が使用可能です。

操作	説明
スライダの上で垂直スクロールバーをクリック。	トピックテキストを 1 パネル分下に移動します。
スライダの下で垂直スクロールバーをクリック。	トピックテキストを 1 パネル分上に移動します。
↑ キーをクリック。	トピックテキストを 1 行分下に移動します。
↓ キーをクリック。	トピックテキストを 1 行分上に移動します。
テキストパネルの外側でクリック。	履歴メニューを表示します。
ホットスポット / リスト行をクリック。	ホットスポット / リスト行を選択します。
最大化 / 復元アイコン (右上隅) をクリック。	トピックウィンドウを最大化したり、復元したりします。
ホットスポット / リスト行をダブルクリック。	ホットスポット / リスト行をアクティブ化にします。
メインパネル (リストトピックではない) の語句上でダブルクリック。	その名前でトピックを検索します。

11.2.2 HyHelp メニュー

HyHelp メニューは、よく使用する機能にすばやくアクセスできるように設計されています。

ヘルプメニューは、**F1=help** キーを押して、どの HyHelp メニューからでもアクセスできます。これは、呼び出したオプションについてのヘルプ画面を呼び出します。

11.2.2.1 ブックマークメニュー

ブックマークメニューとリストは、HyHelp メインメニューから **B** を押すと表示されます。このメニューを、図 11-2 に示します。

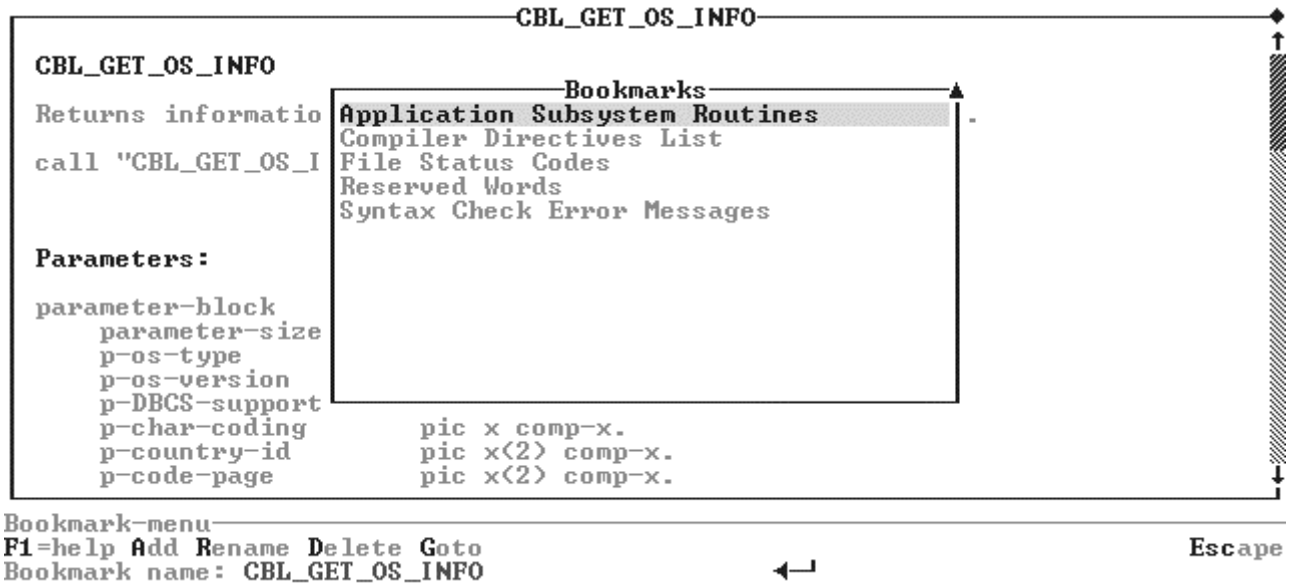


図 11-2：ブックマークメニュー

ブックマークは、あるトピックの特定の箇所を識別するユーザ定義の索引項目です。本の中の特定の箇所にしおりを使用すると同じように、情報の場所をマークするために、このブックマークを使用できます。

ブックマークリストは、ユーザが定義したブックマークのリストです。ブックマークは、索引のようにアルファベット順に並んでいます。各ブックマークは、特定のトピック内の個所を示すためのユーザ定義の名前です。

ブックマークメニューには、次のような操作を実行する機能が表示されます。

- ブックマークの追加
- ブックマークの名前変更
- ブックマークに移動
- ブックマークの削除

11.2.2.2 ファイルメニュー

ファイルメニューは、HyHelp メインメニューから **F** を押すと表示されます。このメニューを、図 11-3 に示します。

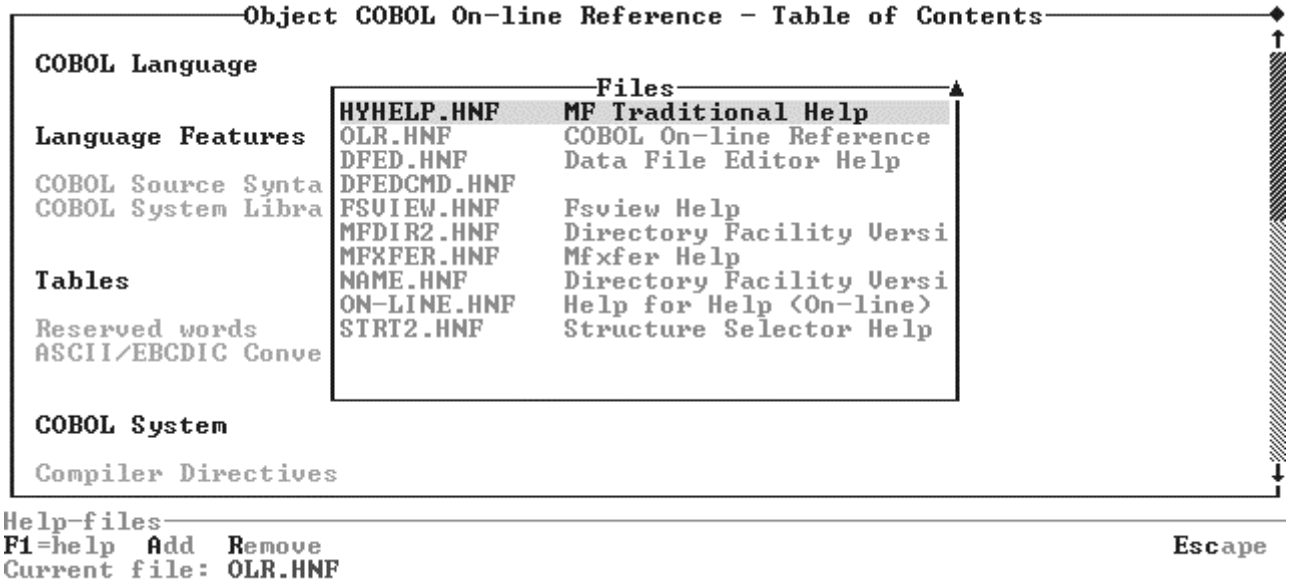


図 11-3：ファイルメニュー

ファイルメニューは、HyHelp で現在使用可能な情報ファイルのリストを表示します。 これらは、トピック名を指定して起動するか、または、ホットスポットではない語句をクリックして検索されたファイルです。

ファイルメニューには、次のような操作を実行する機能が表示されます。

- HyHelp で使用されるファイルを追加
- HyHelp で使用できないようにファイルを削除

11.2.2.3 HyHelp ヘルプメニュー

HyHelp のヘルプメニューは、HyHelp メインメニューから **F1=help** を押すと表示されます。このメニューを、図 11-4 に示します。

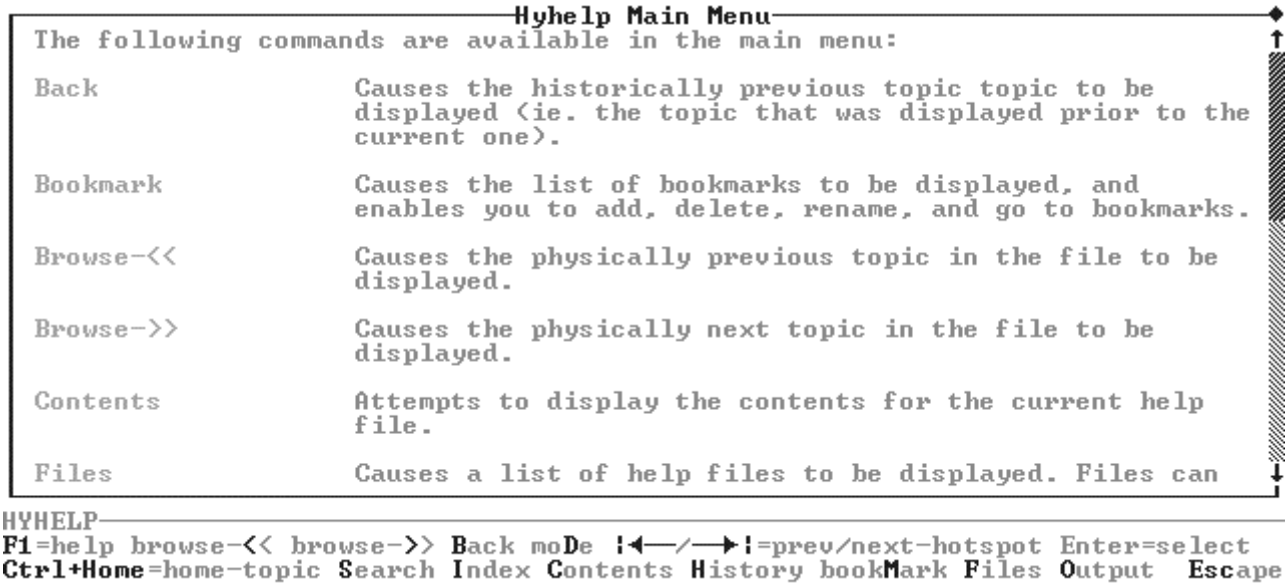


図 11-4：HyHelp ヘルプメニュー

サブメニューや機能を選択するには、適切なファンクションキーや文字を使用します。

11.2.2.4 出力メニュー

出力メニューは、HyHelp メインメニューから O を押すと表示されます。このメニューを、図 11-5 に示します。

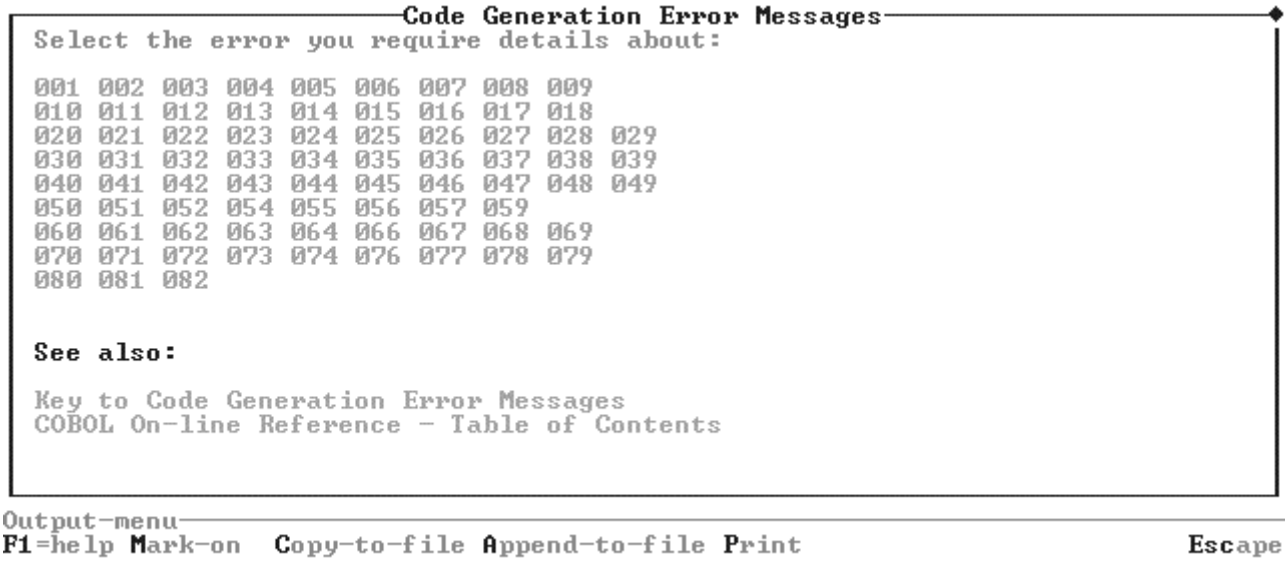


図 11-5：出力メニュー

出力メニューには、次のような操作を実行する機能が表示されます。

- トピック全体または一部をファイルへコピー
- トピック全体または一部を既存のファイルに追加

11.2.3 HyHelp 機能

Adiscf 機能は、メニューからアクセスできます。 次では、HyHelp で使用可能なすべての機能のクイックリファレンスを示します。 また、各機能の詳細を説明します。

11.2.3.1 HyHelp 機能アクセス

次の表では、使用可能な HyHelp 機能をアルファベット順に示し、アクセスするために押すキーを示します。 そのため、キーストロークの順で、その機能にアクセスします。 たとえば、ファイルの削除機能に達するには、HyHelp メインメニューから **F** を押し、続けて **R** を押します。

一部の機能は、矢印キーを使用する必要があります。 使用できる矢印キーがリスト中に示されています。 たとえば、ファイルにコピー機能は、トピック全体をファイルにコピーするか、または、矢印キーを使用して選択したトピックをコピーするかどうかが示されています。

UNIX :
次のリストで、マウス操作の一部の機能は、UNIX の HyHelp では現在使用できません。 そのような機能には、マウスを使用しない代替手段が示されています。

機能	メニューアクセスキー
ホットスポットのアクティブ化	**Tab**、**Enter** または **Back Tab**、**Enter** またはマウスクリック、**Enter** またはマウスのダブルクリック
ブックマークの追加	**M**、**A**
ファイルの追加	**F**、↑、↓、**A**
ファイルに追加	**O**、**M**、↑、↓、**A** または **O**、**A**
後退	**B**
ブックマーク	**M**
後方を参照	<
前方を参照	>
目次	**C**
ファイルにコピー	**O**、**M**、↑、↓、**C** または **O**、**C**
ブックマークの削除	**M**、↑、↓、**D**
エスケープ	**Esc**
ファイル	**F**
ブックマークに移動	**M**、↑、↓、**G**

履歴	H
ホーム	Ctrl+End
索引	I
マーク	O、M
次のホットスポット	Tab
出力	O
前のホットスポット	Back Tab
印刷	O、P
ファイルの削除	F、↑、↓、R
ブックマークの名前変更	M、↑、↓、R
検索	S
モード設定	D

11.2.3.2 HyHelp 機能の説明

ここでは、HyHelp で使用可能な機能を説明します。

ホットスポットのアクティブ化

選択したホットスポットをアクティブにします (**Tab** キー、**Back Tab** キー、またはマウスクリックで選択)。このホットスポットでクロスリファレンスされているトピックが表示されます。

ブックマークの追加

現在のトピックの位置にブックマークを定義します。トピックのタイトルがプロンプトに表示され、ブックマークの名前として使用できることを示します。**Enter** キーを押してこの名前を取り込むか、または、プロンプトに別名を入力し直して **Enter** キーを押します。その名前はブックマークリストに挿入され、HyHelp メインメニューへ戻ります。

ファイルの追加

ヘルプファイルをファイルリストに追加し、HyHelp で使用可能にします。プロンプトでヘルプファイルの名前を入力するか、または、追加したいファイルを検索するために **F2=directory** を選択します。プロンプトに正しいヘルプファイルが表示された後に、**Enter** キーを押します。

パス指定なしでファイル名を指定すると、HyHelp は現在のディレクトリのファイルを検索します。 ファイルが見つからないと、環境変数 COBHNF で定義されたパスのリスト内に指定されているディレクトリで .**hnf** ファイルを検索します。 HyHelp は、Microsoft Advisor 形式のファイル (.**hlp**) を、環境変数 HELPFILES と QH のパスリスト内のディレクトリで検索します。

ファイルが見つかると、リストに追加されます。HyHelp がファイルを見つけられなかったり、ファイルをロードできなかったりした場合は、次のメッセージが表示されます。

```
Failed to add file
```

ファイルに追加

現在のトピック、または、マークされているブロックをテキストファイル **paste.txt** に追加します 。テキストがマークされていない場合は、トピック全体をファイルに追加します。

選択されたテキストは、コピーされる前に 76 文字幅にフォーマットされます。

後退

前に表示されたトピックを表示します。 合間に他のトピックを選択しないで、この機能を繰り返し選択すると、このセッションでこれまで表示されていたトピックが逆順に表示されます。 この場合は、40 トピックまで表示できます。

ブックマーク

『*HyHelp メニュー*』で説明されているブックマークメニューを表示します。 このメニューで使用できる機能の説明については、『ブックマークの追加』、『ブックマークの名前変更』、『ブックマークの削除』、および『ブックマークに移動』を参照してください。

後方を参照

情報ファイルの前のトピックを表示します。参照機能では、オンライン情報ファイル内のトピックをファイルに格納された順序で表示します。また、.**hnf** 形式のファイルにはファイルの作成者が選択した順序で表示できます。これを、参照連鎖と呼びます。 参照連鎖が定義されていない場合は、ファイル内の次 / 前のトピックが表示されます。

この機能でトピックが表示される順序は、完全にファイル構造に依存するので、ここでは詳しく説明しません。

前方を参照

情報ファイルの次のトピックを表示します。参照機能では、オンライン情報ファイル内のトピックをファイルに格納された順序で表示します。また、.hnf 形式のファイルにはファイルの作成者が選択した順序で表示できます。これを、参照連鎖と呼びます。 参照連鎖が定義されていない場合は、ファイル内の次 / 前のトピックが表示されます。

この機能でトピックが表示される順序は、完全にファイル構造に依存するので、ここでは詳しく説明しません。

目次

目次を表示します。目次の例を、図 11-6 に示します。

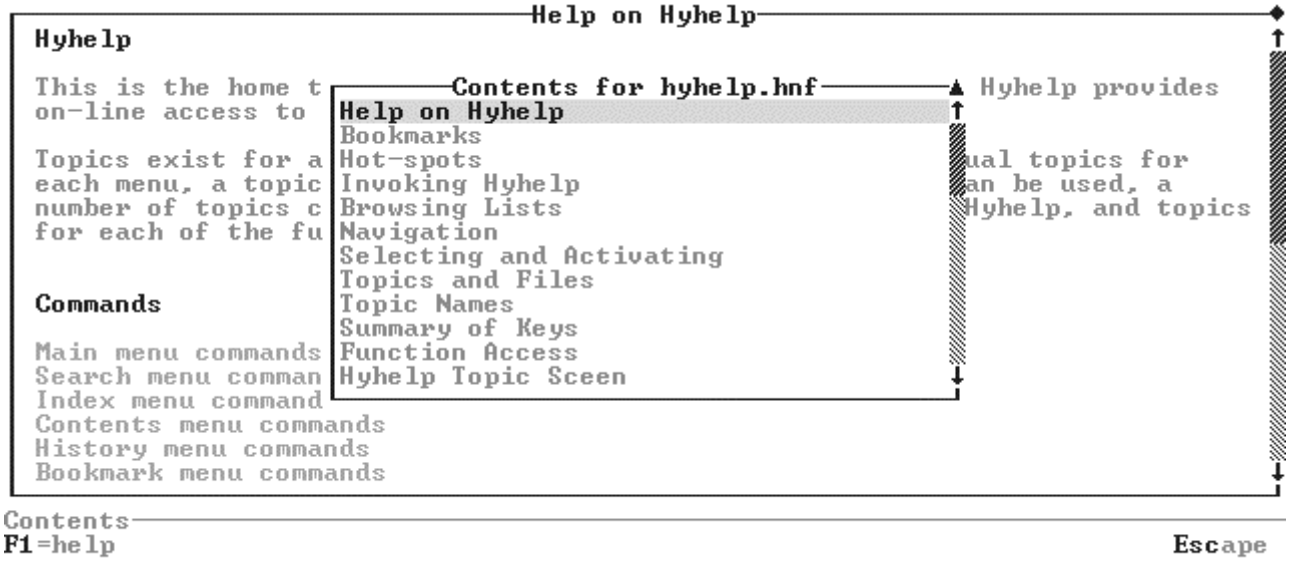


図 11-6：目次画面

.hnf 形式ファイルの一般的な目次が、ポップアップウィンドウにエントリのリストとして表示されます (図 11-6 参照)。エントリは、トピックがファイルに格納された順序です。各エントリは現在のファイルのトピックのタイトルです。 タイトル付きのトピックが、目次リストに表示されます。

エントリを選択するには、↑、↓、Page Up、Page Down、Home、または End キーを使用するか、または、エントリ上でマウスをクリックします。 選択されたエントリはハイライトされます。

ダブルクリックするか、または、選択後に **Enter** キーを押して、エントリをアクティブにします。アクティブになると、目次エントリに関連したトピックが表示されます。

エントリをアクティブにしないで目次リストを終了するには、**Esc** キーを押すか、または、メインパネルの外側でマウスをクリックします。

Net Express オンラインヘルプの目次が表示され、別の選択ができます。

ファイルにコピー

現在のトピック、または、マークされているブロックをテキストファイル **paste.txt** に格納します。マークされているブロックがない場合には、トピック全体がコピーされます。ブロックのマーキングの詳細については、『マーク』機能を参照してください。

選択されたテキストは、コピーされる前に 76 文字幅にフォーマットされます。

ブックマークの削除

ブックマークを削除します。削除するブックマークを選択するには、↑、↓、**Page Up**、**Page Down**、**Home**、または **End** キーを使用するか、または、ブックマーク上でマウスをクリックして、ハイライト表示させます。

D を押します。ブックマークはリストから取り除かれ、削除されます。

エスケープ

HyHelp メインメニューへ戻ります。HyHelp メインメニュー上であれば、HyHelp を終了します。次のプロンプトが表示されます。

```
Exit from HYHELP ?  (Y/N)
```

HyHelp を終了する場合は Y または y を押し、終了をキャンセルする場合は N または n を押します。

ファイル

『*HyHelp メニュー*』で説明されているファイルメニューを表示します。このメニュー上で使用可能な機能については、『ファイルの追加』および『ファイルの削除』を参照してください。

ブックマークに移動

ブックマークに関連付けられたトピックを表示します。トピックを表示するためにブックマークを選択するには、↑、↓、**Page Up**、**Page Down**、**Home**、または **End** キーを使用するか、または、ブックマーク上でマウスをクリックしてハイライト表示させ、**G** を押します。

ブックマークがハイライトされているときに、**Enter** キーを押すか、または、ブックマークリスト内のブックマークをダブルクリックする方法もあります。

履歴

以前に表示したトピックのリストをポップアップウィンドウで表示します (図 11-7 参照)。リストは、最後に表示したトピックを先頭に置いた時間順です。リストの各エントリは、トピックのタイトルです。リストには、最大 40 の最新エントリが表示されます。

エントリを選択するには、↑、↓、**Page Up**、**Page Down**、**Home**、または **End** キーを使用するか、または、エントリ上でマウスをクリックします。選択されたエントリはハイライトされます。

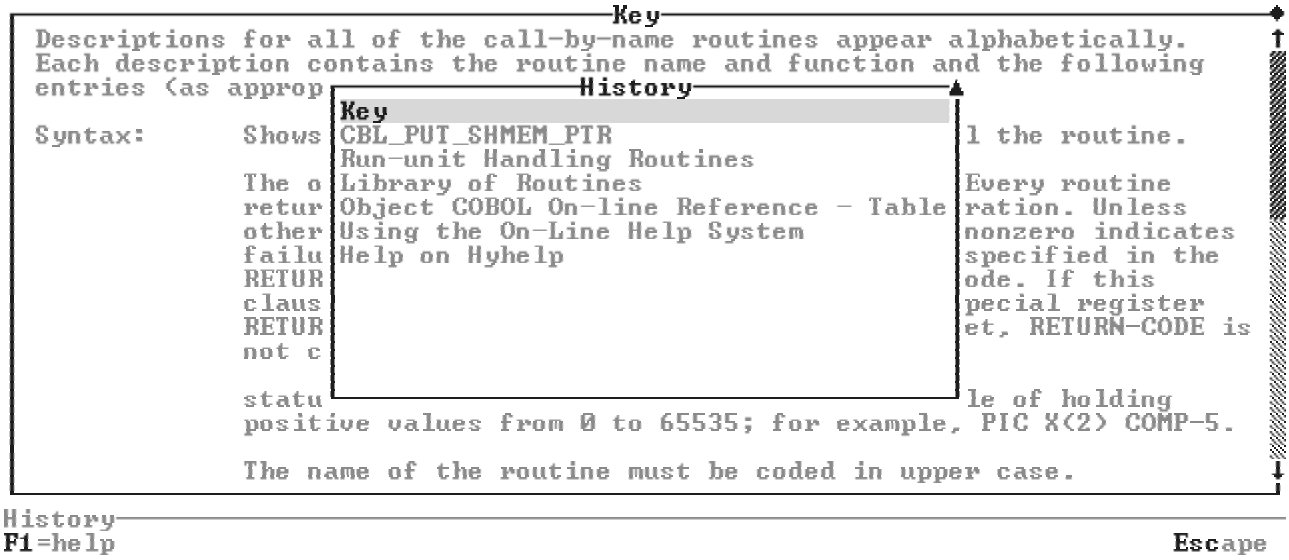


図 11-7：履歴画面

ダブルクリックするか、または、選択後に **Enter** キーを押して、エントリをアクティブにします。アクティブになると、目次エントリに関連したトピックが表示されます。

エントリをアクティブにしないで履歴リストを終了するには、**Esc** キーを押すか、または、メインパネルの外側でマウスをクリックします。

ホーム

現在のヘルプファイルのホームトピックを表示します。

索引

現在のファイルの索引を表示します。一般的な索引を、図 11-8 に示します。

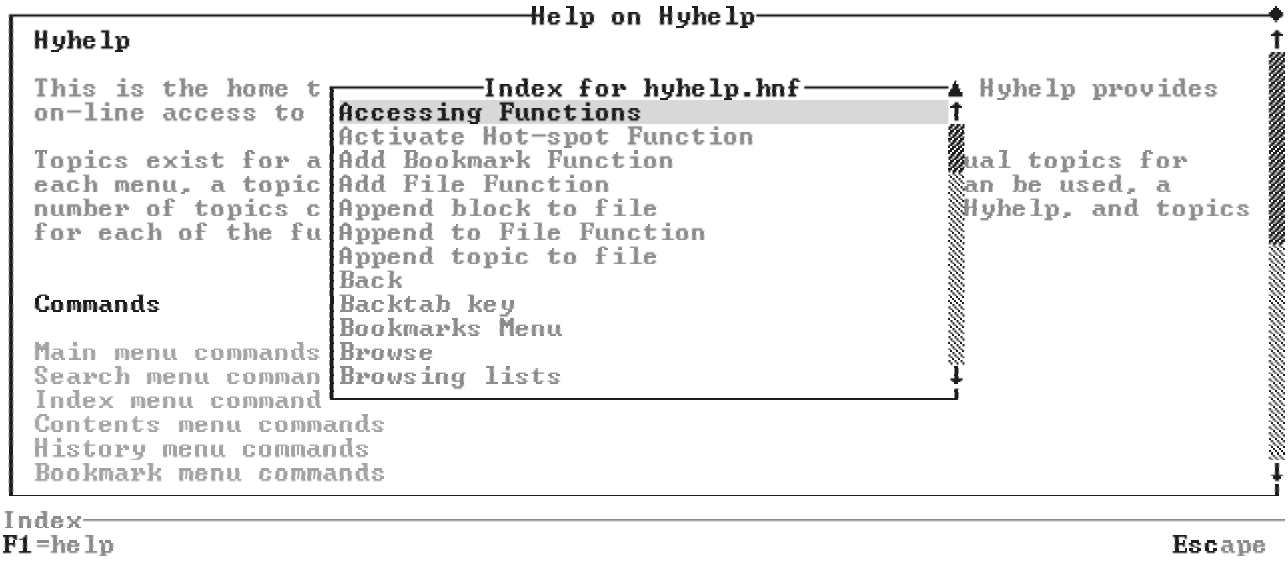


図 11-8：索引画面

.hnf 形式ファイルの一般的な索引が、ポップアップウィンドウにエントリのリストとして表示されます。エントリはアルファベット順で、各エントリは現在のファイルの特定のトピックに関連します。

エントリを選択するには、↑、↓、**Page Up**、**Page Down**、**Home**、または **End** キーを使用するか、または、エントリ上でマウスをクリックします。選択されたエントリはハイライトされます。また、ある文字キーを押すと、その文字で始まる最初のエントリに移動して、エントリを選択できます。

ダブルクリックするか、または、選択後に **Enter** キーを押して、エントリをアクティブにします。アクティブになると、索引エントリに関連したトピックが表示されます。

機能をアクティブにしないで索引リストを終了するには、**Esc** キーを押すか、または、メインパネルの外側でマウスをクリックします。

Net Express オンラインヘルプの索引が表示され、別の選択ができます。

マーク

ファイルやプリンタに出力するブロックをマークします。 そのブロックを開始するには **M** を押し、**↑**、**↓**、**Page Up**、および **Page Down** キーを使用してブロックサイズを指定します。 ブロックが出力したいテキストで選択されると、ファイルにコピー、ファイルに追加、または印刷機能を使用してテキストを出力します。 マーキングをキャンセルするには、再度 **M** を押します。

次のホットスポット

現在のトピックで次のホットスポットを選択します。 ホットスポットが現在選択されていない場合は、最初のホットスポットが選択されます。最後のホットスポットがすでに選択されている場合は、最初のホットスポットが選択されます。

出力

『*HyHelp メニュー*』で説明されている出力メニューを表示します。 このメニューで使用可能な機能については、『マーク』、『ファイルにコピー』、『ファイルに追加』、および『印刷』を参照してください。

前のホットスポット

現在のトピックで前のホットスポットを選択します。 ホットスポットが現在選択されていない場合は、最後のホットスポットが選択されます。最初のホットスポットがすでに選択されている場合は、最後のホットスポットが選択されます。

印刷

現在のトピック、または、マークされているブロックをプリンタに出力します。 テキストがマークされていない場合は、トピック全体が出力されます。

選択されたテキストは、出力される前に 76 文字幅にフォーマットされます。

ファイルの削除

ファイルリストからファイルを削除し、HyHelp で使用できないようにします。 ↑ と ↓ キーを使用して削除するファイルをハイライトし、**R** を押します。 ファイルは、それがリスト中の唯一のファイルでないか、または、現在のファイルでない限り、リストから削除されます。それがリスト中の唯一のファイルでないか、または、現在のファイルでない場合は、次のメッセージが表示されます。

```
Failed to remove file
```

ブックマークの名前変更

既存のブックマークの名前を変更します。 名前を変更するブックマークを選択するには、↑、↓、**Page Up**、**Page Down**、**Home**、または **End** キーを使用するか、または、ブックマーク上でマウスをクリックして、ハイライト表示させます。

R を押して、ブックマークの新しい名前を入力し、**Enter** キーを押します。 選択されたブックマークが名前変更され、新しい名前がリストの新しいアルファベット位置に置かれます。

検索

トピック名を検索します。 プロンプトで検索するトピックの名前の入力が求められ、入力後に **Enter** キーを押します。入力したトピック名が見つからない場合は、次のメッセージが表示されます。

```
Topic "topic-name" not found.
```

ファイルリストに一覧表示されている HyHelp で使用可能なすべてのファイルが、指定された順序で、入力したトピック名で検索されます。

モード設定

カーソルモードとスクロールモードを交互に切り替えます。カーソルモードでは、トピックのウィンドウで矢印キーを使用してカーソルを移動できます。 ホットスポット上にカーソルを移動すると、ホットスポットが自動的に選択されます。 カーソルモードでは、**Ctrl+↑** と **Ctrl+↓** を使用して、トピックをそれぞれ上下にスクロールできます。

スクロールモードでは、トピックを上下スクロールするために、矢印キーを使用します。

第 12 章 : オンラインヘルプシステム

ここでは、Ohbld と Hyhelp の使用方法を説明します。

オンラインヘルプファイルを作成する場合は、次の順序で行います。

1. オンラインヘルプのソースファイルを作成します。
2. オンラインヘルプビルダ (Ohbld) を使用して、ソースファイルをオンラインヘルプファイルに組み込みます。
3. オンラインヘルプビューアでオンラインヘルプファイルを表示します。
4. プログラムにオンラインヘルプビューアの呼び出しを追加します。
5. オンラインヘルプビューアを構成します。
6. オンラインヘルプビューアをそのアプリケーションとともに出荷します。

以後では、これらの手順を説明します。

12.1 ソースファイルの作成

エディタを使用し、表示したい情報を格納するテキストファイルを作成します。1 行は最大 255 文字です。ファイルは、『*オンラインヘルプのソースファイル形式*』の章で説明されている構造に従ってフォーマットされる必要があります。この構造は、HyHelp に対する将来の変更や追加を考慮して設計されており、いくつかの機能が現在の環境によっては表示できない場合があります。ただし、ファイルは、全機能を使用して作成される必要があります。

次に、ソースファイルと情報の構成方法を説明します。

12.1.1 ソースファイルの構造

1 つのオンラインソースファイルの作成に、複数のソースファイルを使用できます。ファイル管理の都合上、または他のシステムとのソースファイルの共有を除けば、複数のソースファイルをもつ必要はありません。最初のソースファイルは、ファイルに付与するタイトルと、ローカルのトピック名をコンテキスト番号に対応させる .define タグで始まります。次にトピックが続きます。各トピックは、.context タグまたは IPF ヘッダーで始まります。この後に、トピックにタイトルを付ける .topic タグと、トピック全体に関連する :i1 索引エントリを付けます。

ファイル内のトピックは、論理的な順序で格納される必要があります。たとえば、ファイルがマニュアルとしてパブリッシュされると、トピックの順序はそのマニュアルに反映されます。これにより、自動的に作成される目次リストは正しく格納されます。

ファイル内の最初のトピックは、ホームトピックです。このホームトピックは、ファイルがトピック名やコンテキスト番号を指定しないでアクセスされると表示されます。ホーム機能でも、いつでも使用可能です。ホームトピックには、ファイル内容を簡単に記述し、ファイルの主要部分に対するクロスリファレンスを提供します。

12.1.2 トピックテキストの作成

トピックは、その情報に容易にアクセスできるように、最適に構造化される必要があります。トピックに何の情報が含まれているかを把握できるように、トピックの始めに十分な情報が含まれ、トピックの最初の表示で明らかにする必要があります。

トピックは、フォーマットされたタグを付けて作成する必要があります。トピックは、段落、固定行のブロック、またはリストで構成されます。リスト内の各エントリは、段落で始まり、段落、固定行のブロック、またはリストと続きます。

ソースのテキストを読みやすくするために、段落テキストの最初の行を段落と同じ行か、または、リスト項目タグに置く必要があります。この部分と、段落の残りのテキストは、トピックの構造を反映するためにインデントされる必要があります。Ohbld が先頭余白を取り除き、HyHelp はトピックウィンドウのサイズに従って再フォーマットします。ただし、ソーステキストに見やすい構造をもたせると、その後の更新が容易になります。ソースを構造化すると、タグの誤りを容易に発見できます。

デフォルトでは、ホットスポットは、自動的にハイライトされます。セクションヘッダーや特別の用語のような他の項目を識別するために、別のハイライト表示を使用することもできます。ただし、ハイライト表示は情報を見にくくするので、多用しないように注意してください。

トピックは、スクロールしなくても見られるような長さである必要があります。

12.1.3 トピック名とコンテキスト番号の定義

ソースファイルで、各トピックはローカルトピック名で識別され、すべてのクロスリファレンス (ハイパーテキストのリンク) がその名前に対して作成されます。各トピックには、.define タグを使用してコンテキスト番号が与えられます。コンテキスト番号は、アプリケーションが (on-line-qry-by-context 動作を使用) 表示する特定のヘルプトピックを問い合わせるときに使用されます。コンテキスト番号が指定されると、ソースファイルを変更しても、(.define 文が変更されない限り) 特定のトピックに割り当てた番号に影響はありません。

ファイルが更新された場合にも、ファイル内のトピックは同じコンテキスト番号を維持することが重要です。ブックマークなどの外部情報は、ファイル名とコンテキスト番号を使用して保存されます。コンテキスト番号が更新の間に変更されると、そのファイルに関連したブックマークはすべて意味がないものになります。

コンテキスト番号を割り当てる場合は、できる限り連続した番号を使用します。オンラインヘルプファイルは、1 から定義された最大数までの各整数値に 3 バイトを割り当てた参照テーブルを保持しています。そのため、500 トピックがあり、コンテキスト番号 1000 をもつと定義すると、少なくとも 1500 バイトがファイルで使用されることになります。

コンテキスト番号は、作成されるファイルの構造を定義するものではありません。コンテキスト番号は、任意に割り当てられます。ファイルの無駄な空間である大きな間隔で番号を任意に割り当てるよりも、すべての番号を連続的に使用することが適切です。

ソースファイル内のクロスリファレンスは、ローカルのトピック名で作成されます。これらは、ファイル空間を節約しクロスリファレンス時の時間を短縮して、コンテキスト番号に変換されます。

外部トピック名ではなく、コンテキスト番号を使用すると、オンラインヘルプファイルは小さくなり、トピックへのアクセスが速くなります。情報は、単一のオンラインヘルプファイルに保持されており、通常、アプリケーションまたはファイル内のクロスリファレンスで直接にアクセスされるので、この方法がほとんどのアプリケーションに適用されます。この一般的な方法は、トピックは外部名ではなく、コンテキスト番号で識別されたり、アクセスされたりします。

外部名は、アプリケーションの外部からトピックを問い合わせる場合のみに使用します。ホームトピックに適用される単一の外部名をアプリケーションのオンラインヘルプファイルに与えると便利です。アプリケーションのヘルプファイルは、アプリケーション名で検索機能を使用して容易にアクセスできます。

12.1.4 文字ベースのオンラインヘルプタグの使用方法

オンラインヘルプのソースファイルは、次のような情報で構成されます。

- オンラインヘルプビルダを制御するビルダ制御タグ
- テキストの一般的な属性を定義する制御タグ
- 特殊なフォーマットを定義するフォーマットタグ

これらのタグは、テキスト表示や、必要であればハイライト表示を定義します。すべてのタグは、コロン (:)、ピリオド (.)、またはドル記号 ($) で始まります。「:」や「$」で始まるタグは、テキスト行のどこからでも開始できます。「.」で始まるタグは、テキスト行の先頭カラムから始まります。ほとんどのタグは、パラメータが続きます。「:」で始まるタグは、ピリオド「.」で終わり、パラメータは通常、ピリオドの前に指定されます。

ソースファイルは、.define、.title、.comment、または .context タグで始まる必要があります。空白行は、固定行のブロックやトピック内でなければ、無視されます。

使用可能なタグの詳細については、『*オンラインヘルプのソースファイル形式*』の章を参照してください。

12.2 文字ベースのオンラインヘルプファイルの組み込み

ソースファイルを作成すると、それを文字ベースのオンラインヘルプファイル (**.hnf**) に組み込むことができます。組み込むためには、オンラインヘルプビルダ (Ohbld) を使用します。

12.2.1 オンラインヘルプビルダ指令の指定

オンラインヘルプのソースファイルをオンラインヘルプファイルに組み込む場合には、オンラインヘルプビルダは、動作を制御するためにデフォルトの設定値を使用します。たとえば、特に指定しない限り、オンラインヘルプファイルの名前は、最初のオンラインヘルプのソースファイル (拡張子 **.hnf**) の名前になり、索引や目次は作成されません。

ただし、COBOL プログラムをコンパイルするときにコンパイラに指令を指定するように、指令を指定してオンラインヘルプビルダのさまざまな動作を変更できます。

12.2.2 オンラインヘルプビルダの実行

オンラインヘルプビルダを実行すると、オンラインヘルプのソースファイルを取り出し、オンラインヘルプファイルを作成します。Net Express のコマンド行やコマンドファイルで、適切な指令を指定して、オンラインヘルプビルダの動作を変更できます。

Ohbld は、フォーマットされたテキストファイルを、HyHelp でアクセスや表示ができる **.hnf** 形式のヘルプファイルに変換します。テキストファイルの形式は、『*オンラインヘルプのソースファイル形式*』の章で説明されています。

Ohbld は指定された順序でソースファイルを処理し、必要であればメッセージを表示します。各トピック、テキストのブロックが作成され、再フォーマットのタグやホットスポット情報が埋め込まれます。各トピックのテキストが処理されると、ブロックは必要であれば圧縮されて、格納されます。すべてのソースファイルが処理されると、必要であれば目次と索引が作成されます。最後に、外部トピック名をコンテキスト番号に、コンテキスト番号を関連トピックに関連付ける参照テーブルがファイルに追加されます。

オンラインヘルプビルダを実行するコマンド行は、次のとおりです。

```
run ohbld {filename     } ...
          {directive    }
          {@command-file}
```

パラメータの詳細は、次のとおりです。

filename — 情報テキストのソースファイル名。ファイル拡張子がない場合は、拡張子 **.txt** が使用されます。複数のファイルを処理する場合は、*filename* にワイルドカード文字「*」と「?」を含めることが可能です。完全パス名を与えることも可能です。

directive — 『*オンラインヘルプビルダ用の指令*』の章に、多くの指令が示されています。すべての指令は「/」で始まり、「NO」をスラッシュと指令の間に挿入すると、その否定となります。たとえば、/NOERRQ は「エラー発生時に停止しない」を意味し、つまり /ERRQ の反対の意味になります。

command-file — ファイル名や指令のリストをもった行順ファイル。コマンド行は入れ子にできません。「アット」文字 (@) は必須です。詳細は、『*Ohbld コマンドファイル*』を参照してください。

指令で上書きされない限り、指定された最初の *filename* が、生成される **.hnf**、**.lst**、および **.msg** ファイルの名前に使用されます。

12.2.3 Ohbld コマンドファイル

オンラインヘルプビルダに多くのファイル名や指令を使用する場合は、Net Express のコマンドプロンプトで毎回入力することは不便です。また、文字列が長すぎて、コマンド行の文字制限内に収まらない可能性があります。

そのため、オンラインヘルプビルダは、コマンドファイルからファイル名や指令を受け付けることができます。コマンドファイルは、ファイル名や指令を含んだ行順ファイルです。Ohbld に、Ohbld コマンド行で「アット」文字 (@) で始まるコマンドファイルを指定し、コマンドファイルを使用するように指示します。たとえば、次のように指示します。

```
run ohbld @myproject.txt
```

myproject.txt には、次の行が含まれます。

```
  File1.txt
  File2.txt
  File3.txt
  /output:myproject.hnf
  /contents
  /index
  /noerrq
```

この例は、次のコマンド行を指定するのと同じです。

```
run ohbld file1.txt file2.txt file3.txt
    /output:myproject.hnf /contents /index /noerrq
```

この例で指定されている Ohbld 指令については、『*オンラインヘルプビルダ用の指令*』の章を参照してください。

コマンド行に、コマンドファイルへの参照に加えて、ファイル名や指令を使用できます。ファイルや指令は、与えられた順序で実行されます。そのため、次のような影響があります。

- コマンドファイルへの参照の前後にファイルを指定して、コマンドファイルのリストの最初や最後にファイルを追加できます。
- コマンドファイルへの参照の後に指令を指定して、コマンドファイルの指令を追加したり、上書きしたりできます。
- コマンドファイルへの参照の前に指令を指定できますが、コマンドファイル内で指定された指令で上書きされる可能性があります。

たとえば、上記のコマンドファイルを使用して、特別のファイル (**preface.txt**) を追加し、エラーメッセージ表示を可能にし (/ERRQ)、エラーメッセージのファイルへの書き込み (/MSG) ができます。これは、次のように入力します。

```
run ohbld preface.txt @myproject.txt /errq /msg
```

12.3 オンラインヘルプビューアでオンラインヘルプファイルを表示

HyHelp は単独で実行でき、他のプログラムからの呼び出しも可能です。起動されると、最初のトピックが表示され、ユーザに管理が渡されます。ユーザは、HyHelp を構成して、使用する色や起動時に使用するヘルプファイルの情報を指定できます。

12.3.1 オンラインヘルプビューアの起動方法

Net Express コマンド行のプロンプトやプログラムから HyHelp を起動できます。

コマンド行

Net Express コマンド行のプロンプトから HyHelp を起動するには、次のコマンドを入力します。

run hyhelp [*filename*!] [*topic-name*]

または

hyhelp [*filename*!] [*topic-name*]

パラメータの詳細は、次のとおりです。

filename 表示するオンラインヘルプファイルの名前。*filename* が指定されていない場合は、COBHNF 環境変数で指定されたオンラインヘルプファイルを含むディレクトリの最初のファイルが使用されます。デフォルトでは、開発環境内のディレクトリで、Hyhelp ヘルプファイル **hyhelp.hnf** が表示されます。

topic-name 表示したいオンラインヘルプファイルのトピック名。

filename が *topic-name* なしに指定される場合は、選択されたオンラインヘルプファイルのホームトピックが表示されます。
topic-name が *filename* なしに指定される場合は、すべてのオンラインヘルプファイルが検索されます。

注：HyHelp の動作を変更する構成ファイルを使用する場合は、コマンド行に「run」を指定する必要があります。コマンド行の「run」を省略すると、HyHelp はデフォルトの構成を使用して実行されます。

12.4 プログラムからのオンラインヘルプビューア呼び出し

Net Express のコマンド行から HyHelp を起動できますが、アプリケーションから起動したい場合もあります。HyHelp は、呼び出しインターフェイスと、アプリケーションにオンラインヘルプを組み込むコピーファイルを含んでいます。

アプリケーションでオンラインヘルプシステムを使用したい場合は、単に HyHelp の呼び出しを追加するのみです。

12.4.1 オンラインヘルプビューアを呼び出すアプリケーションのコーディング

ユーザが特定の項目に関する情報が必要なときに、表示するトピックのコンテキスト番号とそのトピックを含むオンラインヘルプのファイル名を指定する方法で、アプリケーションをコーディングします。また、常に先頭の行に表示するトピック内の位置を設定することもできます。

名前を番号に関連させるためにレベル-78 項目を使用するアプリケーションにコピーファイルを設定することをお薦めします。定義文で使用される同一の名前と番号を、このファイルで使用する必要があります (先行する @ 文字の省略を除いて)。これは、同じトピックを参照するために、同じ名前がアプリケーションを通じて使用されることを意味しています。

12.4.2 呼び出しの定義

COBOL プログラムから標準 COBOL 呼び出しを使用して HyHelp を呼び出せます。

```
call "hyhelp" using on-line-pb
                    on-line-topic
```

パラメータの詳細は、次のとおりです。

on-line-pb	次に示す表示環境に関する固定長の情報を保持するコピーファイル **onl-link.cpy** のパラメータブロック。**onl-link.cpy** を HyHelp を呼び出すプログラムにインクルードする必要があります。
on-line-topic	on-line-qry-by-name 動作に対するトピック名と、空白文字で終了する可変長文字列内のオンラインヘルプファイル名 (または必要な他の情報)。 トピック名は、「!」で終わるファイル名が前に付けられます。ファイル名が渡されない場合は、アクティブなオンラインヘルプファイルが検索されます。 on-line-qry-by-context、on-line-index、または on-line-contents 動作に対しての、on-line-topic は、ファイル名、「!」なしで終了する空白文字を含んでいます。 ファイル名にパスが指定されない場合は、最初に現在のディレクトリが検索され、続けて COBHNF 環境変数で指定されたパスで拡張子によって検索されます。ファイル名に拡張子がない場合は、**.hnf** が仮

定されます。

12.4.3 オンラインヘルプビューアの呼び出し例

```
 78 appl-menu              value 3.
   . . .
copy "onl-link.cpy"
   . . .
   . . .
     move on-line-qry-by-context to on-line-action
     move "myapp.hnf" to on-line-topic
     move appl-menu to on-line-context
     perform call-on-line
   . . .
call-on-line section.
     call "hyhelp" using on-line-pb
                         on-line-topic
     if  return-code not = 0
         display "on-line error:- " on-line-return
     end-if
```

12.4.3.1 onl-link.cpy

次のパラメータ記述は、ファイル **onl-link.cpy** に含まれています。

```
 01 on-line-pb.
       78 on-line-initialize                      value 0.
       78 on-line-qry-by-name                     value 1.
       78 on-line-qry-by-context                  value 2.
       78 on-line-close-down                      value 4.
       78 on-line-index                           value 5.
       78 on-line-contents                        value 6.
       78 on-line-add-file                        value 7.
     03 on-line-action             pic 9(2) comp-x
                                 value on-line-qry-by-name.
     03 on-line-return             pic 9(2) comp-x value 0.
       78 on-line-bad-action                      value 1.
       78 on-line-topic-not-found                 value 2.
       78 on-line-file-not-found                  value 3.
     03 on-line-flags              pic 9(2) comp-x value 0.
       78 on-line-default-window                  value 0.
       78 on-line-window-specified                value 7.
     03 on-line-win-flags          pic 9(2) comp-x
                               value on-line-default-window.
     03 on-line-context-no         pic 9(4) comp-5.
```

```
    03 on-line-ptr                pic 9(4) comp-5.
    03 filler.
        05 on-line-window-x        pic 9(4) comp-5.
        05 on-line-window-y        pic 9(4) comp-5.
        05 on-line-win-width       pic 9(4) comp-5.
        05 on-line-win-height      pic 9(4) comp-5.
    03 on-line-ui-id               pic 9(9) comp-5.
```

on-line-action

HyHelp で実行される動作を指定します。

on-line-initialize

HyHelp を事前ロードし、初期化します。 これを使用しない場合は、最初の HyHelp 呼び出しでロードと初期化が行われます。

on-line-qry-by-name

on-line-topic で与えられたトピック名で選択されたトピックを表示します (オプションでファイル名が先行します)。 ユーザは、HyHelp の全機能を使用可能です。

ファイル名が指定されると、そのファイルが最初に検索され、続けて HyHelp で使用可能なファイルが検索されます。

on-line-qry-by-context

on-line-context-no で指定されたコンテキスト番号で選択されたトピックと、on-line-topic で指定されたファイルが表示されます。 ユーザは、HyHelp の全機能を使用可能です。

on-line-close-down

システムを終了します (すべてのリソースが解放され、すべてのテーブルがクリアされます)。 この後に、CANCEL "HYHELP" 文が続く必要があります。

on-line-index

指定されたファイルの索引をエントリ時に表示します。filename は、後ろには「!」が付かない形式で on-line-topic にある必要があります。

on-line-contents

指定されたファイルの目次をエントリ時に表示します。.hnf ファイルと、h.con の名前をもつトピックを含む Microsoft Advisor 形式のファイルのみで動作します。filename は、「!」を後ろに付けないで on-line-topic で指定する必要があります。

on-line-add-file

指定されたファイルを HyHelp で使用可能なファイルのリストに追加します。

on-line-return

0 以外の場合は、何かのエラーが発生したことを示します。 on-line-return の値は、次のとおりです。

値	意味
on-line-bad-action	動作コードが、定義されたコードではありません。
on-line-topic-not-found	指定されたトピックが見つかりません。
on-line-file-not-found	指定されたファイルが見つかりません。

on-line-flags

次のビット設定 (ビット 0 が LSB) に従って、問い合わせの呼び出し後のリソースの処理を指定します。

ビット	説明
7	設定すると、メニューを表示しないで HyHelp を起動します。
6	未使用。
5	設定すると、HyHelp 画面が終了時に表示されたままになります。
4	設定すると、呼び出しで要求されたトピックが見つからない場合には、HyHelp はアプリケーションに制御を戻しません。
3	設定すると、初期化時にファイルが自動的に開きません。
2	設定すると、ユーザ画面が保存され、終了時に復元されます。
1	設定すると、終了時にすべてのヘルプファイルを閉じます。
0	設定すると、動的に割り当てられたすべてのメモリが、終了時に解放されます。通常の使用で、このビットを設定することをお奨めします。

on-line-win-flags

次のビット設定 (ビット 0 が LSB) に従って、ウィンドウ環境の動作を指定します。

ビット	説明
2	トピックのウィンドウサイズにパラメータが使用されていると設定されます。このビットが設定されていない場合は、トピックを全画面で表示します。

1　　トピックウィンドウ位置にパラメータが使用されていると設定されます。

0　　未使用。

HyHelp が画面より小さいウィンドウで呼び出されると、ウィンドウの右上隅に最大化のアイコンが付けられることに注意してください。このアイコンをクリックすると、全画面にウィンドウが拡大され、右上隅に復元アイコンが付けられます。復元アイコンをクリックすると、ウィンドウは元のサイズに戻ります。これ以外の方法では、ウィンドウサイズを変更できません。

on-line-context-no/on-line-ptr

動作が on-line-qry-by-context である場合は、トピックを検索するために使用されるコンテキスト番号です。on-line-topic で名前が付けられたオンラインヘルプファイル内のこの番号に関連付けられたトピック (感嘆符「!」なしで) が直接指定されます。

また、トピックの開始位置は on-line-ptr に指定できます。これは、常にウィンドウの先頭行に位置する文字のテキストバッファ内オフセットを示す数字です。この値は、試行錯誤して算定します。

1 つの例として、次のように指定します。

- トピックのヘッダーに 14。
- 各テキスト文字 (段落行の先行や後続する空白文字を除く) に 1。
- 各段落に 1。
- 各 :lines と :cgraphic ブロックに 4、ブロック内の各行に 1。
- 任意のテーブルの始めに 2、末尾に 2、テーブルの各項目に 2。

on-line-window-x/y

on-line-win-flags のビット 1 が設定されている場合は、ウィンドウの開始位置 (文字単位で) を指定します。画面の左上隅を 0 とする相対位置です。

on-line-win-width/height

ウィンドウサイズを文字数で指定します。on-line-win-flags のビット 2 が設定されている場合には、この項目は無視されます。

on-line-ui-id

このデータ項目は、HyHelp で使用されていません。

on-line-topic

on-line-topic は、on-line-qry-by-name 動作に対するトピック名を含みます。トピック名は、「!」で終わるファイル名が前に付けられます。*filename* が渡されない場合は、アクティブなオンラインヘルプファイルが検索されます。

on-line-qry-by-context、on-line-index、または on-line-contents の動作では、on-line-topic には、ファイル名と空白文字のみがあります。ファイル名は「!」では終了しません。

ファイル名にパスが指定されない場合は、最初に現行ディレクトリが検索され、続けて COBHNF や QH 環境変数で指定されたパスで拡張子によって検索されます。ファイル名に拡張子がない場合は、**.hnf** が仮定されます。

12.5 オンラインヘルプビューアの構成

HyHelp でデフォルトの設定を使用しない場合は、構成ファイルにパラメータを指定して構成できます。標準の構成ファイルである HyHelp 構成ファイル (**hyhelp.cfg**) を使用します。

hyhelp.cfg は、Net Express に提供されていません。デフォルト設定を変更する場合は、このファイルを作成する必要があります。**hyhelp.cfx** が **\bin** ディレクトリに構成ファイルの例として提供されています。この **.cfx** ファイルを、次の 3 つのどれかの方法で使用します。

- ファイルを **hyhelp.cfg** に名前変更し、COBDIR 環境変数で指定されたディレクトリに含めます。
- ファイルを編集し、**hyhelp.cfg** に名前を変更します。この新しい **.cfg** ファイルを COBDIR 環境変数で指定されたディレクトリに含めます。
- **.cfx** ファイルからオプションをローカルの構成ファイルに含めます。

run コマンドで HyHelp を実行し、構成ファイルの変更を反映させます。

上記のどの方法でも、構成オプションはタグ [HYHELP] に続けて指定する必要があります。

使用可能な構成ファイルのすべてのオプションについては、『*構成オプションタグ*』の章を参照してください。

オンラインヘルプのソースファイルの ASCII テキスト形式については、『*オンラインヘルプのソースファイル形式*』の章を参照してください。

12.6 オンラインヘルプビューアの出荷

HyHelp は次のファイルを使用します。HyHelp を使用するアプリケーションを出荷する場合は、これらのファイルを含める必要があります。

- **hyhintf.gnt** (**bin** ディレクトリ内)
- **hyhelp.hnf** (**docs** ディレクトリ内)

- **hyhelp.lbr** (**bin** ディレクトリ内)
- **hyhelp.cfg** (**bin** ディレクトリ内)
- **helpname.lbr** (**bin** ディレクトリ内)
- **name.lbr** (**bin** ディレクトリ内)
- **utils.lbr** (**bin** ディレクトリ内)

マイクロソフト形式の .hlp ファイルへのアクセスが必要である場合は、HyHelp には次の追加ファイルが必要です。

- **mshelp.dll**
- **mshif.dll**
- **mshif.exe**

第 13 章 : オンラインヘルプのソースファイル形式

ここでは、オンラインヘルプソースファイルの ASCII テキスト形式について説明します。

オンラインヘルプソースファイルは、標準の ASCII テキストファイルです。トピックに分割するために必要なテキストが含まれます。タイトルなどのトピックの外観を定義するために、タグが使用されます。表示時にどのようにフォーマットするかを示すために、追加のタグがテキスト内で使用されます。

多くの場合には、トピックは、水平スクロールを必要としないで、トピックのウィンドウ幅に合うように、表示時に動的に自動フォーマットされます。この動的なフォーマットは、フレックステキストと呼ばれます。いくつかのフォーマットタグはこの機能を禁止しているので、使用する際には注意が必要です。

オンラインヘルプシステムのタグは、IBM IPF システムや Microsoft Advisor との互換性を保つように設計されています。ただし、他のタグに優先して、オンラインヘルプシステムのタグを使用することをお奨めします。

ビルダは、Microsoft Advisor 形式のソースファイルも処理するため、フレックステキスト処理を確実に有効にするために、最初のテキスト行を定義する段落やリストタグを使用する必要があります。

ソースファイルに含まれるタグには、次の 4 つの種類があります。

- ビルダ制御タグ

  オンラインヘルプビルダの制御を行います。

- 制御タグ

  テキストの一般的な属性を定義します。

- フォーマットタグ

  特殊なフォーマットを定義します。

- テキスト行と属性

  これらのタグは、テキスト表示や、必要であればハイライト表示を定義します。

すべてのタグには、コロン (:)、ピリオド (.)、またはドル記号 ($) が前に付けられます。「:」や「$」で始まるタグは、テキスト行のどこからでも開始できます。「.」で始まるタグは、テキスト行

の先頭カラムから始まります。 すべてのタグには、パラメータが続きます。「:」で始まるタグは、ピリオド「.」で終わり、パラメータは通常、ピリオドの前に指定されます。

ソースファイルは、.define、.title、.comment、または .context タグで始まる必要があります。空白行は、固定行のトピックやブロック内でない限り、無視されます。

13.1 IBM IPF システムとの互換性

Ohbld は、IPF V2.00 で指定されている情報表示機能 (IPF) のタグや記号を認識します。次のタグがオンラインヘルプシステムで完全に実現され、『*オンラインヘルプタグ*』の章で説明されています。

.*	:fn	:note	:rm
:caution	:hp1 〜 :hp9	:nt	:sl
:cgraphic	.im	:ol	:title
:color	:li	:p	:ul
:dl	:lines	:parm1	:userdoc
:fig	:lm	:pd	:warning
:figcap	:lp	:pt	:xmp

次の IPF タグは、このオンラインヘルプシステムで部分的に実現されており、一部が『*オンラインヘルプタグ*』の章で説明されています。

:artlink
:artwork
:font
:h1 〜 :h6
:i1 および :i2
:link

次の IPF タグは、現在オンラインヘルプシステムで実現されていません。ただし、使用されていても Ohbld により認識されます。

:acviewport
:ctrl
:ctrldef
:ddf
:docprof
:fn
:hide
:icmd
:isyn
:pbutton
:table

IPF の詳細については、IBM OS/2 2.0 テクニカルライブラリの『*情報表示機能の手引きと解説書*』を参照してください。

Ohbld は、オンラインヘルプファイルの特殊文字を表示するために使用される IPF 記号を認識します。各記号は、アンパサンド (&) で始まり、ピリオド (.) で終わります。 記号は、大文字と小文字を区別し、大文字と小文字とでは異なる記号になります。

次の表は、IPF 記号の例で、表示される文字、名前、および IPF 記号を示しています。

文字	名前	IPF 記号
&	アンパサンド	&amp
@	アット文字	&atsign
$	ドル記号	&dollar
>	右向きの矢印	&rahead

IPF 記号の詳細については、IBM OS/2 2.0 テクニカルライブラリの『*情報表示機能の手引きと解説書*』を参照してください。

13.2 ビルダ制御タグ

ビルダ制御タグは、ソーステキストのある部分を取り込んだり、除外したりするために使用されます。次に、これらのタグの構文と機能の概要を示します。

13.2.1 構文の概要

- 注釈の定義。

  .comment *comment-text*

- ソーステキストのコピー。

  .im *filename*

- 条件付きビルド。
- $if *condition*.
- *true-text*
- [$else. *false-text*]
  $end.

- マクロの定義。
- $defmacro *macro-name*.
- *macro-text* [*macro-param* ...]

$edefmacro.

- マクロ内での条件付きビルド。
- $ifmdef *macro-param*.
- *true-text*
- $elsem.
- *false-text*

$endm.

- マクロの使用。

  $macro *macro-name* [macro-param=[']*value*['] ...].

13.3 制御タグ

制御タグは、テキストの属性を定義しますが、テキストが表示されるときの外観には関係しません。次に、アルファベット順にこれらのタグの構文と機能の概要を示します。

13.3.1 構文の概要

- 参照連鎖

  .browse #*chain-id* @*topic-name* @*topic-name*

- コマンドトピック
- .command
- :exec
- :call

:shell

- 目次

  .context *topic-name*

- コンテキスト番号の定義

  .define @*topic-name* @*context-no*

- トピックの終了

  .end

- 凍結行

  .freeze nn

- ヘッダー

  :h1, :h2, :h3

- 索引エントリ
- :i1. *index-text*
- :i2. *index-text*
  ..index *index-text*

- ホットスポットのリンク
- :link.
  :elink.

- クロスリファレンスのリスト

  .list

- ポップアップウィンドウ

  .popup

- サブヘッダー

  :h4, :h5, :h6

- ファイルのタイトル

  .title *title-text*

- トピックのタイトル
- .heading *text*
  .topic *text*

- ユーザドキュメント
- :userdoc.
  :eusrdoc.

13.4 フォーマットタグ

フォーマットタグは、テキストが表示されるときの外観を定義します。フォーマットタグは「:」で始まり、任意のカラムから始められますが、最初のカラムから始めることをお奨めします。次に、使用可能なフォーマットタグをアルファベット順に表示し、その概要を説明します。

13.4.1 構文の概要

- 警告
- :caution.
  :ecaution.

- 色

  :color [*fc*] [*bc*]

- 例
- :xmp.
  :exmp.

- 図
- :fig.
- :figcap
  :efig

- 固定行ブロック
- :cgraphic.
- :ecgraphic.
-
- :lines.
  :elines.

- 脚注
- :fn [*id=*].
  :efn.

- リスト部

  :lp.

- 余白
- :lm margin=*n*.
  :rm margin=*n*.

- 備考

  :note [*heading-text=*] *text*
  または
  :nt [*heading-text=*] *text*
  :ent.

- 段落
- :p.*text*
- テキストリスト

- 定義リスト
- :dl. [compact] [tsize=*nn*] [break=*xxxx*].
- :dthd. *term-hdg*
- :ddhd. *definition-hdg*
- :dt. *term*
- :dd. *definition-text*
- :dt. *term*
- :dd. *definition-text*

  :edl.

- 順序付きリスト
- :ol [compact].
- :li. *text*

  :eol.

- パラメータリスト
- :parml [compact] [tsize=*nn*] [break=*xxxx*].
- :pt. *term*
- :pd. *definition-text*

  :eparml.

- 単純リスト
- :sl [compact].
- :li. *text*

  :esl.

- 順序なしリスト
- :ul [compact].
- :li. *text*

  :eul.

- テキストリストは、段落、固定行ブロック、さらにリストを含むことができます。
- 警告
- :warning [*heading-text*] *text*

  :ewarning.

次に、フォーマットタグをアルファベット順で説明します。ただし、テキストリストタグはリストの末尾にまとめてられています。

13.4.2 単純、順序付き、順序なしリストの使用方法

リストタグは、いろいろな種類のリストを作成するために使用されます。

リストは :sl, :ul or :ol タグで始まります。単一パラメータ compact が指定できます。各項目間に空白行のないリストを作成します。

リストの各項目は、**:li.** タグで始まります。 項目のテキストが同じ行でタグの後に続き、ソーステキストの構造が見えるように、インデントされます。

項目には、:p. タグで始まる追加の段落、:lines や :cgraphic タグで始まる固定行のブロックを含むことができます。 項目には、他のいろいろな種別のリストを含むこともできます。

リストは、リストの終了タグ **:esl.**、**:eul.**、または **:eol** で終了します。

13.4.3 定義リストとパラメータリストの使用方法

定義リストとパラメータリストは、いくつかの説明の段落が続くテキストを用語として提示するために使用されます。 これらのリストは 2 つのカラムがあり、最初のカラムは用語、2 番目のカラムはこれらの定義です。 用語の特徴に従って、最初のカラムの幅を指定します。

それに続く段落は、インデントできます。また、用語を頭に置いて、同じ行に適切なインデントで続けるか、または、次の行から続けることを選択できます。

定義リストは、ブレークパラメータのデフォルト設定、および、パラメータリストが用語や定義のヘッダーを含まないことを除けば、パラメータリストと同じです。

13.4.3.1 定義リスト

定義リストは用語とその値を 2 カラムで示します。デフォルトでは、説明は用語と同じ行から始まります。

定義リストの指定は、**:dl** で始めます。

13.4.3.2 パラメータリスト

パラメータリストは、パラメータのリストとその意味や値を示すために使用されます。 リストは定義リストと同じように表示されます。デフォルトでは、定義リストは 2 番目のカラム (定義テキスト) の始まりが常に最初のカラム (用語) の 1 行下に表示されます。 ヘッダータグは含まれません。

パラメータリストの指定は、**:parml** で始めます。

13.4.4 リストパラメータ

compact	リスト中の項目が空白行で区切られないように指定します。
tsize=*nn*	用語用の領域を文字数で指定します。 定義段落は、この値によってインデントされます。 デフォルトは 10 です。
break=*xxxx*	用語と定義間の間隔を指定します。ここで、*xxxx* は、次のどれかになります。 none 常に、用語の後に定義の最初の行が続きます。 用語が

tsize より長いと、行は 1 行空けられて続きます。

`fit` 用語が tsize より短いと、同じ行に説明の最初の行を置きます。短くない場合には、説明は次の行から始まります。

`all` 常に、説明を次の行から始めます。

デフォルト値は、定義リストに対して none で、パラメータリストには `all` です。

使用可能なオンラインヘルプシステムのタグについては、『*オンラインヘルプタグ*』の章を参照してください。

13.5 テキスト行と属性

トピックの本体は、テキスト行で構成されます。テキスト行は、ピリオド以外の文字で始まります。次のフラグは、トピックのテキスト属性を変更するために埋め込まれます。

テキスト	効果
\a	ホットスポットの開始 (表示)
\b	太字
\i	斜体
\u	下線
\p	平文
\v	ホットスポットのクロスリファレンスの区切り文字 (非表示)
:hp1.	斜体、:ehp1. を最後に付ける
:hp2.	太字、:ehp2. を最後に付ける
:hp3.	太字と斜体、:ehp3. を最後に付ける
:hp4.	赤、:ehp4. を最後に付ける
:hp5.	下線、:ehp5. を最後に付ける
:hp6.	下線と斜体、:ehp6. を最後に付ける
:hp7.	下線と太字、:ehp7. を最後に付ける
:hp8.	青、:ehp8. を最後に付ける
:hp9.	マゼンタ、:ehp9. を最後に付ける

太字、斜体、および下線は、切り替え可能で、組み合わせることも可能です。\p は現在の属性をリセットし、平文に戻します。\a は、構成ファイルの *color-hotspot* パラメータで定義された属性で表示ホットスポットのテキストを表示します (詳細については、『*オンラインヘルプシステム*』の章にある『*オンラインヘルプビューアの構成*』を参照してください)。

円記号 (\) をトピック内で使用する場合は、円記号の組を使用します。 そのため、
c:\\cobol\\demo は、画面上に c:\cobol\demo として表示されます。

注: 特殊文字 (\、:、.、および $) をテキストの一部に使用する場合は、「\」を前に付けます。

例

```
.context @file-menu
:p. 現在のドライブを確認してください。:hp1.
    c:\\cobol\\demo:ehp1.
```

これは、ドライブが斜体として定義された属性で、正しく表示されます。

13.5.1 ホットスポットと非表示テキスト

ホットスポットは、関連するトピック名をもったテキスト行内の表示テキストです。 ホットスポットがオンラインヘルプビューアでアクティブになると、クロスリファレンスのトピックが表示されます。

デフォルトでは、ホットスポットは自動的にハイライトされます。 構成ファイルでホットスポットの属性を定義する場合は、構成済み属性のホットスポットを \a フラグを使用して表示します。 それ以外の場合は、現在のテキスト属性で表示されます。

クロスリファレンスは、\v タグで区切られた非表示テキストを使用して埋め込まれます。 デフォルトでは、クロスリファレンスに対するホットスポットは、最初の空白文字が含まれていない最初の \v タグに先行する文字列です。 \a (アンカー) フラグは、これを上書きするために使用されます。 \a フラグは、クロスリファレンスに対するホットスポットが \a フラグと最初の \v タグの間の文字列であることを示します。 また、:link や :elink. を使用してホットスポットを定義することもできます。

クロスリファレンスは、1 つのトピック名です。 参照されたトピックが同じファイルにあると、トピックへ直接ポインタが作られるので、ローカル参照を使用することをお奨めします。

定義したトピック名に加えて、次のトピック名をクロスリファレンスとして使用できます。

トピック名	説明
!B	後退 - ホットスポットが選択されたときに、以前に表示されたトピックを表示します。
!C	目次 - ホットスポットが選択されたときに、目次リストが表示されます。

!I 索引 - ホットスポットが選択されたときに、索引が表示されます。

コンテキスト番号 (.define タグで以前に定義されている) が使用される場合は、(コロンで区切られた) コンテキスト番号を続けることができます。これは、トピックが最初に表示されるときに、先頭行の参照されているトピックの箇所を示します。この番号を自動的に定義する方法はありません。そのため、試行錯誤で必要な箇所に対するオフセットを作成する必要があります。

そのかわりに、:h4, :h5 や :h6 タグにクロスリファレンスを使用し、トピックが最初に表示されるときに、先頭行に表示されるトピックを指定できます。

例

次の例で、「ハイライト」という語句と「入れ子にされたメニュー」の指定は、それぞれトピック @hilite および @submenua にオフセット 123 でクロスリファレンスしているホットスポットです。

```
.context @menu
:p. これらの項目の指定によって、ハイライト\v@hilite\v
と\a 入れ子にされたメニュー\v@submenua:123\v について検索できます。
```

13.5.2 Microsoft Advisor 形式のソースファイル

Microsoft Advisor ヘルプファイルの作成に使用されるコマンドのサブセットが、このシステムで認識されます。これにより、Microsoft Advisor システムで使用されるソースファイルが、**.hnf** ファイルを作成するために使用できます。この形式で作成されたトピックは、固定行のブロックとして扱われます。そのため、ウィンドウサイズが変更されたときに、再フォーマットされません。

2 つのシステム間でソースファイルを共有しない場合は、ソースファイル作成時に、前述のフォーマットタグを使用する必要があります。

Microsoft Advisor 形式のファイルは、文字ベースの環境、つまり、Hyhelp の GUI エミュレーションモードでのみサポートされます。

第 14 章 : オンラインヘルプビルダ用の指令

ここでは、オンラインヘルプビルダ (Ohbld) の動作を制御するために指定される指令について説明します。 各項目で、指令が設定されたときの効果について説明します。 別な方法が指定されていない場合には、その指令を無効にすると逆の効果が得られます。 指令を明示的に設定しない場合のデフォルト設定も説明されています。

指令設定の詳細については、『*オンラインヘルプシステム*』の章にある『*オンラインヘルプビルダの実行*』を参照してください。

/AUTOHOTSPOT

デフォルト

/AUTOHOTSPOT

ホットスポットテキストにデフォルトの属性を使用するかどうかを指定します。 このデフォルト属性は、構成ファイルで COLOR-HOTSPOT を指定して変更できます。 オンラインヘルプビルダは、ソースファイルに指定されているホットスポットの属性を無視し、この指令のデフォルト設定でエラーメッセージを出力します。

この指令を /NOAUTOHOTSPOT に設定すると、ソースファイルに指定されている、ホットスポットのテキスト属性が表示されるようにします。

/CASE

デフォルト

/NOCASE

作成したオンラインヘルプファイル内のトピック名を、大文字小文字を区別するものと見なして指定します。 デフォルトでは、トピック名は大文字小文字を区別しません。

/COMPRESS[:*compress-type*]

/C

デフォルト

/COMPRESS:1

トピックを圧縮するかどうかを指定します。

パラメータ

***compress-type* は、次の値のどちらかです。**

0 - 圧縮なし
1 - 圧縮

/CONTENTS

デフォルト

/NOCONTENTS

オンラインヘルプファイルに定義されているすべてのトピックタイトルのリストを含んだトピックを作成します。このトピックの ID はファイルヘッダーに格納され、オンラインヘルプビューアの目次機能で使用されます。

複数レベルの目次を作成するために、ヘッダータグを使用できます。ヘッダータグの詳細については、『*オンラインヘルプのソースファイル形式*』の章にある『*制御タグ*』を参照してください。

/CONTENTSTITLE:"*contents-title*"

デフォルト :

/CONTENTSTITLE:"*hnf-filename* の目次"

ヘルプファイルの目次リストのタイトルバーに表示されるテキストを指定します。この指令の効果を得るには、/CONTENTS または /CONTENTSTOPIC を指定する必要があります。

/CONTENTSTOPIC

デフォルト

/NOCONTENTSTOPIC

目次をリストボックスではなく、トピックとして表示します。

目次が 1,000 件以上のエントリを含む場合は、/CONTENTS ではなく /CONTENTSTOPIC を使用します。

/CPY:*filename*

デフォルト

/NOCPY

パラメータ

filename　　コピーファイルに付ける名前。

オンラインヘルプのソースファイルの各ローカルトピックにレベル-78 のエントリを含む COBOL コピーファイルを作成します。この **.cpy** ファイルを使用し、トピック名を検索するよりも、COBOL プログラムから直接トピックにアクセスできます。

/CPY がファイル名なしで指定される場合には、コピーファイルの名前は **.hnf** ファイルと同じ名前で、拡張子 **.cpy** が付けられます。

例

ファイル **fred.txt** には、次が含まれます。

```
.context @one
.topic トピック 1
:p.これは、トピック 1 です。
.context @two
.topic トピック 2
:p.これは、トピック 2 です。
.context @three
.topic トピック 3
:p.これは、トピック 3 です。
```

次のオンラインヘルプビルダのコマンド行を指定します。

```
run ohbld fred /cpy:fred.cpy
```

次を含むファイル **fred.cpy** が作成されます。

```
78  78-ctxt-one       value 1.
78  78-ctxt-two       value 2.
78  78-ctxt-three     value 3.
```

注: 30 文字以上のコンテキスト名は **.cpy** ファイルでは切り捨てられます。

/DEFINE:*filename*

デフォルト

/NODEFINE

オンラインヘルプのソースファイルの各ローカルトピックに .define エントリを含む **.def** ファイルを作成します。この **.def** ファイルを使用し、既存のコンテキスト番号を維持するために、同一ソースファイルの次のビルドへ入力します。

たとえば、古いブックマークのファイルが古いコンテキスト番号に依存したり、COBOL プログラムがトピックをコンテキスト番号で直接アクセスしていたりする場合に、この指令が役に立ちます。

/DEFINE がファイル名なしで指定されると、**.def** ファイルは **.hnf** ファイルと同じ名前で、拡張子 **.def** が付けられます。

例

ファイル **fred.txt** には、次が含まれます。

```
.context @one
.topic トピック 1
:p.これは、トピック 1 です。
.context @two
.topic トピック 2
:p.これは、トピック 2 です。
.context @three
.topic トピック 3
:p.これは、トピック 3 です。
```

次のオンラインヘルプビルダのコマンド行を指定します。

```
run ohbld fred /define:fred.def
```

次を含むファイル **fred.def** が作成されます。

```
.define @one 1
.define @two 2
.define @three 3
```

/ERRQ

デフォルト

/ERRQ

デフォルトでは、オンラインヘルプビルダは、オンラインヘルプのソースファイルにエラーを発見するたびに一時停止します。そして、継続、中断、最後までズームするかを尋ねられます。/NOERRQ が指定される場合には、オンラインヘルプビルダはエラー時に停止しません。

/EXTERNAL

デフォルト

/NOEXTERNAL

ビルドされるオンラインヘルプファイル内で定義されていないトピックを参照するすべてのクロスリファレンスリストを作成します。参照リストは、/LIST の設定によって、ファイルに書き込まれたり、表示されたりします。

/IGNORE:*boolean-variable [,boolean-variable]*

デフォルト

/NOIGNORE

その値が偽であるようなブール変数名のリストを指定します。この指令は、条件付きビルドで使用されます。これらの変数はテキスト内で $if タグで使用され、条件付きでマークされたテキストのブロックをビルドから除外します。

パラメータ

boolean-variable	真か偽の値をもつ、名前または識別子。/IGNORE は、このパラメータを偽に設定します。

例

ファイル **fred.txt** には、次が含まれます。

```
.context @topic1
.topic トピック 1
```

```
:p. $if cond-1. cond-1 が真の場合は、このテキストをインクルードします。
$else.
:p. cond-1 が偽の場合は、このテキストをインクルードします。
$end.
```

次のオンラインヘルプビルダのコマンド行を指定します。

```
run ohbld fred /ignore:cond-1
```

$if と $else タグの間のテキストがビルドから除外され、$else と $end タグの間のテキストがインクルードされます。

/IGNORE 指令を省略すると、ビルドの結果に影響はしません。ただし、オンラインヘルプビルダは、宣言されていない変数を偽と見なし、警告メッセージを出力します。

その他の参照項目

『/INCLUDE』

/INCLUDE:*boolean-variable{,boolean-variable}*

デフォルト

/NOINCLUDE

その値が真であるようなブール変数名のリストを指定します。この指令は、条件付きビルドで使用されます。 これらの変数はテキスト内で $if タグで使用され、条件付きでマークされたテキストのブロックをビルドにインクルードします。

パラメータ

boolean-variable	真か偽の値をもつ、名前または識別子。/INCLUDE はこのパラメータを真に設定します。

例

ファイル **fred.txt** には、次が含まれます。

```
.context @topic1
.topic トピック 1
:p. $if cond-1. cond-1 が真の場合は、このテキストをインクルードします。
$else.
:p. cond-1 が偽の場合は、このテキストをインクルードします。
```

$end.

次のオンラインヘルプビルダのコマンド行を指定します。

```
run ohbld fred /include:cond-1
```

$if と $else タグの間のテキストがビルドにインクルードされ、$else と $end タグの間のテキストが除外されます。

/INCLUDE 指令は、/IGNORE 指令と連動して使用され、より正確な条件付きビルドを可能にします。次のコマンド行があります。

```
run ohbld fred /include:cond-1 /ignore:cond-2
```

これは、最初にマークされたテキストのブロックをインクルードし、2 番目のテキストのブロックを除外します。

その他の参照項目

『/IGNORE』

/INDEX

デフォルト

/NOINDEX

ヘルプファイルに定義されているすべての索引を含む索引トピックを生成します。索引トピックのコンテキスト番号は、ファイルヘッダーに格納され、オンラインヘルプビューアの索引機能で使用されます。索引機能を選択すると、索引がリストボックスに表示されます。索引からエントリを選択すると、そのトピックが表示されます。

/INDEXTITLE:"*index-title*"

デフォルト

/INDEXTITLE:"*hnf-filename* の索引"

ヘルプファイルの索引リストのタイトルバーに表示されるテキストを指定します。この指令の効果を得るには、/INDEX または /INDEXTOPIC を指定する必要があります。

/INDEXTOPIC

デフォルト

/NOINDEXTOPIC

索引をリストボックスではなく、トピックとして表示します。

索引が 1,000 件以上のエントリを含む場合は、/INDEX ではなく /INDEXTOPIC を使用します。

/LIST

デフォルト

/NOLIST

デフォルトでは、オンラインヘルプビルダからのすべての出力が、画面に表示されます。
/LIST を指定すると、オンラインヘルプファイルと同じ名前の概要のリストファイルが .lst の拡張子を付けて作成され、現在のディレクトリに格納されます。

/LISTTEXT

デフォルト

/NOLISTTEXT

ソーステキストをビルドし、画面上に表示します。 /LIST が指定されると、この情報も指定された .lst ファイルに書き込まれます。

/LOCAL

/L

デフォルト

/NOLOCAL

すべてのローカルトピック名のリストを作成します。リストは、/LIST の設定によって、ファイルに書き込まれたり、表示されたりします。

/MAXNAME:*context-length*

デフォルト

/MAXNAME:32

コンテキスト名の最大長を指定します。*context-length* は、1 ～ 32 です。 オンラインヘルプビルダは、指定した値より長いコンテキスト名を検出すると、警告メッセージを表示します。

後で構文チェックを行うコピーファイルを /CPY 指令で作成する場合は、22 以下の値を指定する必要があります。 これは、オンラインヘルプビルダがコピーファイルに書き込むときに 8 文字 (「78-ctxt-」の長さ) を各コンテキスト名の前に付けて、COBOL のデータ名の最大長が 30 文字であるためです。

/MSG

/M

デフォルト

/NOMSG

最初に読み込まれたオンラインヘルプのソースファイルと同じ名前のメッセージファイルが .msg の拡張子を付けて作成され、現在のディレクトリに格納されます。 このファイルは、オンラインヘルプのソーステキストのエラーを検索するために使用されます。

/OUTPUT:*filename*

デフォルト

なし。 最初のオンラインヘルプのソースファイルの名前を、拡張子 .hnf を付けて使用します。

ビルドされるオンラインヘルプファイルに使用する名前を指定します。 拡張子 .hnf を使用する必要があります。

/PASS2

デフォルト

/NOPASS2

デフォルトでは、オンラインヘルプビルダはオンラインヘルプファイルをビルドするために 1 パス (でのビルド) をとります。 /PASS2 は、オンラインヘルプビルダにオンラインヘルプファイルを 2 パス (でのビルド) で読むように指示します。

サブヘッダー、:h1、:h2、および :h3 タグ内の RES パラメータに対する前方参照を正しく解決するために、/PASS2 が必要です。

/SOUND

デフォルト

/SOUND

デフォルトでは、オンラインヘルプビルダは、オンラインヘルプのビルド時にエラーが発生すると、システムのベルを鳴らします。 オンラインヘルプ のソースファイルに多数のエラーがある場合は、/NOSOUND を指定してシステムのベルを止めることができます。

/TOPIC

/T

デフォルト

/NOTOPIC

ビルドされるオンラインヘルプファイルに定義されているトピック名のリストを作成します。リストは、/LIST 指令の設定によって、ファイルに書き込まれたり、表示されたります。

/VERSION:*number*

デフォルト

/VERSION:6

パラメータ

number　　使用するバージョン。

作成されるオンラインヘルプファイルのバージョンを指定します。 /VERSION:*n* は、オンラインヘルプビルダに、**.hnf** ファイルのバージョン *n* をビルドするように指示します。 この指令に有効なパラメータは、5 と 6 のみです。

オンラインヘルプビューアのバージョンがオンラインヘルプビルダよりも古いときに、この指令を使用します。

/WARNING

デフォルト

/WARNING

ビルド時に発生した警告メッセージを、プレフィックス OHB*nnn*W をつけて表示します。ここで *nnn* はメッセージ番号です。 /NOWARNING を指定すると、警告メッセージが抑止されます。

これらのエラーのリストは、『*Ohbld エラーメッセージ*』の章を参照してください。

第 15 章 : 構成オプションタグ

ここでは、オンラインヘルプビューアの動作を構成する場合に使用できる構成オプションタグを一覧表示します。指定するタグの詳細については、『*オンラインヘルプシステム*』の章にある『*オンラインヘルプビューアの構成*』を参照してください。

構成オプションは、構成ファイル内で [HYHELP] タグに続けて記述します。

ファイル、またはファイル内のセクションは、次の形式である必要があります。

```
[HYHELP]
option
option
...
```

パラメータの詳細は、次のとおりです。

option　　次に一覧表示されている 1 つ以上の構成オプションを示します。

[ ] (角かっこ) は必須です。

COLOR *font foreground* **ON** *background*
COLOUR *font foreground* **ON** *background*

font を色を付けて定義します。

パラメータ

font　　構成するフォントタイプ。たとえば、平文、太字、斜体。
foreground　　このフォントの前景色。
background　　このフォントの背景色。

font に使用できる値は、次のとおりです。

フォント	説明
p	平文
u	下線
i	斜体

b	太字
bi	太字と斜体
bu	太字と下線
iu	斜体と下線
bui	太字、斜体、および下線

デフォルト

フォントタイプのデフォルト色は、次のとおりです。

フォントタイプ	デフォルト色
p	sys-att-11
u	黒地に緑
i	黒地にシアン
b	sys-att-12
bi	黒地に明るいシアン
bu	sys-att-05
iu	黒地にマゼンタ
bui	黒地に明るいマゼンタ

foreground と *background* に使用できる値は、次のとおりです。

```
black
blue
brown
cyan
green
magenta
red
yellow
dark-grey
light-blue
light-cyan
light-green
light-grey
light-magenta
light-red
```

例

```
color iu brown on red
```

斜体と下線が付けられたテキストは、赤地に茶色で表示されます。

COLOR-*object foreground* ON *background*
COLOUR-*object foreground* ON *background*

object を色を付けて定義します。

COLOR-*object system-attribute*

パラメータ

object	色付けするインターフェイス領域。
foreground	この領域の前景色。
background	この領域の背景色。

デフォルト

インターフェイス領域のデフォルト色は、次のとおりです。

エントリ	デフォルト
topic-border	sys-att-04
topic-cursor	sys-att-02
cursor	sys-att-02
hotspot	斜体に定義された属性
block	sys-att-02
popup-border	sys-att-04
menu-text	sys-att-03
menu-message	sys-att-04
keytops	sys-att-04
popup-text	sys-att-11
popup-cursor	sys-att-02

object に使用できる値は、次のとおりです。

オブジェクト	説明
topic-border	トピックパネルの境界。
topic-cursor	トピックウィンドウのカーソル。
cursor	topic-cursor と同じ。
hotspot	非ハイライト表示ホットスポット。
block	マークされたテキスト。
popup-border	ポップアップリストボックスの境界。
menu-text	メニュー項目。
menu-message	メニューメッセージ。
keytops	メニュー項目のキートップ。
popup-text	ポップアップリストの項目。
popup-cursor	ポップアップリストのカーソル。

foreground と *background* に使用できる値は、次のとおりです。

black	dark-gray
blue	light-blue
brown	light-cyan
cyan	light-green
green	light-gray
magenta	light-magenta
red	light-red
yellow	

例

```
color-hotspot white on cyan
```

ホットスポットは、シアンの背景に白で表示されます。

FILE *filename*

起動時にオンラインヘルプビューアで使用されるオンラインヘルプファイルを指定します。このリスト内のファイルの順序は、ファイルリスト中のファイルの順序を決定します。

パラメータ

filename	起動時に使用可能になるファイル名。

FILE と MFPATH 構成パラメータで指定されたファイルは、構成ファイルに指定された順序でアクセスされます。オンラインヘルプファイルが見つからない場合は、COBHNF 環境変数で指定されたディレクトリが検索されます。 オンラインヘルプファイルがこの段階で見つからないと、現在のディレクトリのオンラインヘルプファイルが使用されます。

FILE オプションでは 1 つのファイルしか指定できませんが、複数の FILE オプションを含めることができます。 オンラインヘルプビューアは、最大 32 ファイルを使用できます。

MFPATH *path*

Micro Focus 形式のオンラインヘルプファイルを検索するパスを指定します。

パラメータ

path	オンラインヘルプファイルが明示的なパス情報をもたずに指定された場合に使用するパス。*path* は、パスリストを含んだ環境変数 ($ が先頭に付けられます) です。

MONO *font monochrome-attribute*

オンラインヘルプビューアで使用されるフォントのモノクロ属性を定義します。

パラメータ

font	構成するフォントタイプ。 たとえば、平文、太字、斜体。
monochrome-attributes	このフォントタイプに割り当てるモノクロ属性。

デフォルト

構成ファイルに定義された属性に依存します。

font に使用できる値は、次のとおりです。

フォント	説明
p	平文

u	下線
i	斜体
b	太字
bi	太字と斜体
bu	太字と下線
iu	斜体と下線
bui	太字、斜体、および下線

例

```
mono bu underline reverse-video
```

太字と下線が付けられたテキストが、モノクロ画面上の反転表示内に下線付きで表示されます。

MONO-*object monochrome-attribute*

オンラインヘルプビューアのインターフェイス領域でのモノクロ属性を定義します。

パラメータ

object	構成するインターフェイス領域。
monochrome-attributes	このインターフェイス部分に割り当てるモノクロ属性。

デフォルト

構成ファイルに定義された属性に依存します。

object に使用できる値は、次のとおりです。

オブジェクト	説明
topic-border	トピックパネルの境界。
topic-cursor	トピックウィンドウのカーソル。
cursor	topic-cursor と同じ。
hotspot	非ハイライト表示ホットスポット。
block	マークされたテキスト。

popup-border	ポップアップリストの境界。 たとえば、目次リスト。
menu-text	メニュー項目。
menu-message	メニューメッセージ。
keytops	メニュー項目のキートップ。
popup-text	ポップアップリストの項目。
popup-cursor	ポップアップリストのカーソル。

例

mono-hotspot highlight underline

ホットスポットは、モノクロ画面上にハイライトされ、下線が付けられて表示されます。

PASTE-FILE *filename*

パラメータ

filename	出力するファイル。

デフォルト

PASTE-FILE .\paste.txt

オンラインヘルプビューアの「ファイルにコピー」や「ファイルに追加」機能を使用して出力されたヘルプのページに対するファイル名を指定します。

PRINT-DEVICE *device-name*

パラメータ

device-name	印刷する装置の名前。

デフォルト

PRINT-DEVICE :lst

オンラインヘルプビューアから印刷機能が選択された場合に、行が送られる装置やファイルを指定します。

第 16 章 : オンラインヘルプタグ

ここでは、使用可能なオンラインヘルプシステムのタグについて説明します。タグはアルファベット順に説明されています。

次のような種類のタグが使用できます。

- オンラインヘルプビルダを制御するビルダ制御タグ
- テキストの一般的な属性を定義する制御タグ
- 特殊なフォーマットを定義するフォーマットタグ

また、次のような制御も可能です。

- ハイライト表示などのテキストや属性
- トピック間にリンクを提供するホットスポットや非表示テキスト

16.1 オンラインヘルプビルダタグ構文の指針

オンラインヘルプビルダのタグには、次の規則が適用されます。:

- ピリオド (.) で始まるタグは、行の先頭カラムから始まる必要があります。
- コロン (:) やドル記号 ($) で始まるタグは、行のどこからでも始められます。

  コロン (:) で始まるタグは、ピリオド (.) で終わり、通常、パラメータはピリオドの前に出現します。

- すべてのタグには、パラメータが続きます。
- パラメータ *topic-name* が続くタグは、この topic-name は COBOL の一意名と同じ構文規則に従います。詳細については、『***言語リファレンス***』を参照してください。

16.2 ビルダ制御タグ

ビルダ制御タグは、ソーステキストのある部分をインクルードしたり、除外したりするために使用されます。次に、これらのタグの構文と機能の概要を示します。

タグ	説明
.	注釈の定義。
$if	条件付きビルド。

$else	条件付きビルド。
$end	条件付きビルド。
$defmacro	マクロの定義。
$edefmacro	マクロの定義。
$ifmdef	マクロ内での条件付きビルド。
$elsem	マクロ内での条件付きビルド。
$endm	マクロ内での条件付きビルド。
$macro	マクロの使用。
.comment	注釈の定義。
.im	ソーステキストのコピー

16.3 制御タグ

制御タグは、テキストの属性を定義しますが、テキストが表示されるときの外観には関係しません。次に、アルファベット順にこれらのタグの構文と機能の概要を示します。

タグ	説明
.browse	参照連鎖
.command	コマンドトピック
:exec	
:call	
:shell	
.context	目次
.define	コンテキスト番号の定義
.end	トピックの終了
.freeze	凍結行
:h1	ヘッダー
:h2	
:h3	
:h4	サブヘッダー
:h5	
:h6	
.heading	トピックのタイトル
:i1.	索引エントリ

:i2.

..index

:link　　ホットスポットのリンク

:elink

.list　　クロスリファレンスのリスト

.title　　ファイルのタイトル

.topic

:userdoc　　ユーザドキュメント

:eusrdoc

16.4 フォーマットタグ

フォーマットタグは、テキストが表示されるときの外観を定義します。フォーマットタグは「:」で始まり、任意のカラムから始められますが、最初のカラムから始めることをお奨めします。次に、アルファベット順にこれらのタグの構文と機能の概要を示します。

タグ	説明
:caution	警告
:ecaution	
:cgraphic	固定行ブロック
:ecgraphic	
:color	色
:dl	定義リスト
:dthd	
:ddhd	
:dt	
:dd	
:dt	
:dd	
:edl	
:fig	図
:efig	
:figcap	図のキャプション
:fn	脚注

:efn
:lines
:elines
:lp　　リスト
:note　　備考
:nt
:ent
:ol　　順序付きリスト
:li
:eol
:p　　段落
:parml　　パラメータリスト
:pt
:pd
:eparml
:sl　　単純リスト
:li
:esl
:ul　　順序なしリスト
:li
:eul
:xmp　　例
:exmp
:warning　　警告
:ewarning

*

構文

. comment-text*

パラメータ

comment-text　　オンラインヘルプビルダで無視されるテキスト行。

.* タグで、オンラインヘルプのソースファイルに、オンラインヘルプビルダで無視される注釈や注記のテキストを含めることができます。

備考

このタグで始まる行は、テキストファイルのどこにでも指定できます。このタグと次に続くテキストの間に、空白文字は必要ありません。

.* タグを他のタグの中に (つまり、タグ開始のコロンとタグ終了のピリオドの間) に記述できません。また、.* タグを、他のタグとそれに付随するテキストや属性の間に記述できません。

このタグは、最初のカラムから始まる必要があります。

例

```
.*このテキストは、ソースファイルのコンパイル時に無視されます。
```

---

browse

構文

.browse #*chain-id* @*topic-name* @*topic-name*

パラメータ

chain-id	参照連鎖を識別します。1 トピックに 1 つのみの連鎖が許されているので、このタグは #1 で指定する必要があります。
topic-name	ローカルのトピック名、もしくは他のトピック番号。最初の *topic-name* は、連鎖の現在のトピックに先行するトピックを識別します。この値が @0 の場合には、現在のトピックは連鎖の最初のトピックです。 同様に、2 番目の *topic-name* は、連鎖で現在のトピックに続くトピックを識別します。この値が @0 の場合には、現在のトピックは連鎖の最後のトピックです。 特殊な場合の @0 を除いて、トピック名は、番号または数字で始まる文字列であってはなりません。

.browse タグは、トピックへの参照先やトピックからの参照先を作者が制御できるようにします。

備考

.browse タグを使用して、オンラインヘルプのソースファイルに現れる順序でトピックを参照するデフォルトの動作を変更できます。

このタグは、最初のカラムから始まる必要があります。

例

次のトピックは、参照連鎖の最初のトピックです。

```
.context @main-menu
.browse #1 @0 @submenu
```

このトピックが表示されると、オンラインヘルプビューアの「前方を参照」機能が @submenu トピックを表示します。テキストファイル内でこのトピックの前後のトピックに関係なく、「後方を参照」機能は無効で、常に「前方を参照」機能で @submenu トピックを表示します。

---

caution

構文:

```
:caution [text="caution-heading"].
caution-text
...
:ecaution.
```

パラメータ

caution-heading	注意のヘッダーを変更します。ヘッダーを指定しない場合は、テキスト「CAUTION:」が画面上に表示されます。
caution-text	警告メッセージのテキスト。

:caution タグは、警告メッセージを表示します。

備考

ヘッダーと警告テキストの間に、空白行が 1 行挿入されます。:warning を参照してください。

例

```
:caution.
これは、「CAUTION:」のヘッダーをもつ警告メッセージです。
:ecaution.
:caution text="Extreme Caution".
これは、「厳重な注意 :」のヘッダーをもつ警告メッセージです。
:ecaution.
```

---

cgraphic

構文

```
:cgraphic.
text/graphics
...
:ecgraphic.
```

備考

:cgraphic タグは :lines タグと似ていますが、テキストは常にモノスペースフォントで表示されます。 これにより、表示時にきれいに揃える文字の図を作成できます。

例

```
:cgraphic.
これらの行をインデントと間隔を維持し、
ラップしないできれいに揃えます。
+------------------+ +------------------+
|                  | |                  |
|    ブロック A    | |   ブロック B     |
|                  | |                  |
+------------------+ +------------------+
          |                    |
          +--------------------+
                     |
:ecgraphic.
```

---

color

構文

```
color [fc=foreground-text-color]
[bc=background-text-color].
```

パラメータ

foreground-text-color　　テキストの色。

background-text-color　　テキストの背景色。

:color タグは、テキストや背景の色を変更します。

備考

画面の色は変更されません。このタグで設定された色は、別のタグを指定するか、または、ヘッダー定義があるまで、効果が継続します。次の値を使用できます。

- default
- blue
- cyan
- green
- neutral
- red
- yellow

システムカラーに戻すには、fc=default と bc=default を使用します。

例

```
:color fc=red bc=neutral.
:p. このテキスト行は、中間色の背景に赤で表示されます。
:color fc=blue bc=red.
:p. このテキスト行は、赤の背景に青で表示されます。
```

---

command

.command タグは、オンラインヘルプビューア内から、他のプログラム、オペレーティングシステムのコマンド、またはシェルの実行を可能にします。

備考

このタグは、.context、:h1、:h2、または :h3 タグの直後に指定する必要があります。このコマンドトピックの構文は、通常のトピックとは異なります。

オンラインヘルプビューアでコマンドトピックがアクセスされると、指定されたコマンドシーケンスが実行され、最後に表示したトピックが表示されます。.browse タグを使用して、ファイルを参照するときに、コマンドトピックをバイパスできます。ホームトピックは、コマンドトピックであってはなりません。

構文

.command タグには、次のタグのどれかが直後に続きます。

タグ	説明
.topic	目次リストのエントリを挿入します。目次リストからこのエントリを選択すると、そのコマンドを実行します。トピックが **.context** タグで始まる場合のみに、このタグを使用します。

..index, :i1, :i2	索引エントリを挿入します。
:call	COBOL を呼び出します。:call タグに続くテキストは、RUN コマンド入力時のものとまったく同じです。
:exec	オペレーティングシステムの命令や実行可能ファイルを実行します。このタグに続くパラメータは、Net Express コマンドプロンプト入力時のものとまったく同じです。
:shell	現在のオペレーティングシステムのシェルを起動します。

任意の数のタグが、コマンドトピック内で指定できます。タグは、指定された順序で実行されます。

例

```
.context @test
.command
:exec dir /p
:exec list ohbld.doc
:shell
```

このトピックにアクセスすると、**dir/p** と **list ohbld.doc** が現在のオペレーティングシステム内で実行され、Net Express コマンドプロンプトに戻ります。

Net Express コマンドプロンプトから、次を入力します。

```
exit
```

これで、前のトピックに戻ります。

---

comment

構文

.comment *comment-text*

パラメータ

comment-text	オンラインヘルプビルダで無視されるテキスト行。

.comment タグで、オンラインヘルプのソースファイルに、オンラインヘルプビルダで無視される注釈や注記のテキストを含めることができます。

備考

このタグで始まる行は、テキストファイルのどこにでも指定できます。

このタグは、最初のカラムから始まる必要があります。

例

```
.comment   このテキストは、無視されます。
```

---

context

構文

```
.context topic-name
```

パラメータ

topic-name	トピックの名前。

備考

各トピックは、複数 (最大 32) の .context タグが先行します。*topic-name* は、トピックを検索するために使用できるトピック名です。トピック名は 30 文字に制限され、空白文字以外の任意の文字が含まれます。複数のトピック名をトピックに関連付けることができます。1 つのトピックに関連付けされるすべての .context タグは、物理的にともに出現する必要があります。

このタグは、最初のカラムから始まる必要があります。

ローカルトピック名は、先頭の「アット」文字 (@) で識別されます。ローカルトピック名は、単一のオンラインヘルプファイルを構成するオンラインヘルプソースファイルからのみ参照できます。ローカルトピック名は、ファイルに格納されません。そのかわりに、すべてのローカルトピック名への参照は、ビルド時に解決されます。このような参照は、使用領域を節約でき、アクセスが速くなります。ローカルトピック名を数字で始めることはできません。

すべてのトピック名は、外部ソースからアクセスが必要なものを除き、ローカルである必要があります。

:h1、*:h2*、*:h3* と *:h4*、*:h5*、*:h6* を参照してください。

例

```
.context @entry-field
.context entry-field
```

---

define

構文

.define *@topic-name @context-no*

パラメータ

topic-name	ローカルトピック名。
context-no	そのトピックに割り当てられた 1 〜 65,535 のコンテキスト番号。

.define タグは、トピックにコンテキスト番号を予約するために使用されます。

備考

たとえば、トピックが COBOL プログラムから直接参照される場合など、トピックのコンテキスト番号を知っている必要がある場合に、この機能は役に立ちます。 すべての .define タグは、最初のトピックの前に出現する必要があります。

このタグは、最初のカラムから始まる必要があります。

.define タグがトピックに指定されない場合には、オンラインヘルプビルダは、自動的にコンテキスト番号をトピックに割り当てます。 ただし、コンテキスト番号は、ファイルがビルドされるたびに変更されます。

トピックは複数のコンテキスト番号をもてますが、コンテキスト番号は 1 つのトピックのみに関連付けられます。 コンテキスト番号は、完全に任意で、ファイルの構造には関連がありません。

例

```
.define @main-menu @13
```

---

dl

構文

```
:dl [compact] [tsize=nn] [break=xxxx].
:dthd. term-hdg
:ddhd. definition-hdg
:dt. term
:dd. definition-text
:dt. term
:dd. definition-text
......
:dt. term
:dd. definition-text
```

:edl.

パラメータ

nn	用語に割り当てられる文字数。定義段落は、この値によってインデントされます。デフォルトは 10 です。
xxxx	用語とその定義の間隔。
term-hdg	用語の上に表示されるヘッダー (先頭カラム)。
definition-hdg	定義の上に表示されるヘッダー (2 番目のカラム)。
term	用語として表示されるテキスト。
definition-text	定義として表示されるテキスト。

備考

リスト内の項目は、用語と定義で構成されます。 他のタグを使用して、用語と定義のカラムのヘッダーをハイライトできます。

xxxx に使用可能な値は、次のとおりです。

none	定義の最初の行は、常に用語と同じ行に表示されます。 用語が *nn* より長いと、1 行空けられて定義が続きます。 これは、デフォルト設定です。
fit	用語が *nn* より短いと、定義の最初の行が用語と同じ行に続きます。 用語が *nn* より短くない場合には、定義の最初の行は次の行から始まります。
all	定義の最初の行は、常に用語に続く行に表示されます。

compact を使用して、リスト内の項目が空白行で区切られないように指定できます。

2 つのカラムのヘッダーは、:dthd と :ddhd タグで定義されます。 用語は :dt. タグで始まり、説明は :dd. タグで始まります。説明には、追加の段落、ブロック、他のリストを続けることができます。リストを終了するには、edl. タグを使用します。

$defmacro

構文

```
$defmacro macro-name.
macro-text [macro-param ... ]
$edefmacro.
```

パラメータ

macro-name　　定義するマクロ名。

macro-text　　マクロに含まれるテキスト。マクロにパラメータを、パーセント記号 (%) で囲んで渡すことができます。

macro-param　　マクロで受け取られるパラメータ。

同じテキストを何度も繰り返したり、あるパターンに適合させたりする場合に、マクロを使用します。マクロ定義に基本パターンを指定すると、マクロを呼び出して、異なるパラメータで同じ基本パターンを使用できます。

例

```
$defmacro topic.
:h1 id=%id%.%title%
:p.%text%

$defmacro macro-1.
:p. このテキストは、macro-1 の結果です。パラメータはありません。
$edefmacro.

$defmacro macro-2.
:p. このテキストは、macro-2 の結果です。パラメータは、%a%、"%b%"、%c% です。
$edefmacro.
```

マクロの定義後の使用方法については、『$macro』タグを参照してください。

---

end

構文

```
.end
```

.end タグは、不要な空白行の表示を無効にして、トピックを強制終了します。

備考

ディスプレイの最終行の下へのカーソル移動を防止し、リストトピックを終了するためには、このタグを使用する必要があります。

このタグは、最初のカラムから始まる必要があります。

---

fig

構文

```
:fig.
text
...
:efig.
```

パラメータ

text　　プロポーショナルフォントで表示するテキスト。

:fig タグは、プロポーショナルフォントを使用して、テキストを入力したものと同じ形式で表示するために使用されます。

備考

テキストサイズがウィンドウサイズより大きい場合には、テキストはクリップされます。テキストがプロポーショナルフォントで表示されるので、それぞれの文字や数字はきれいに並べることができません。このタグは、文字の図を表示するには適切ではありません。:cgraphic. と :xmp タグを参照してください。

このタグ内に図のキャプションを指定できます (:figcap 参照)。

例

```
:fig.
カラム 1            カラム 2
------------------------------
カラム 1 テキスト  カラム 2 テキスト
カラム 1 テキスト  カラム 2 テキスト
カラム 1 テキスト  カラム 2 テキスト
カラム 1 テキスト  カラム 2 テキスト
:efig.
```

---

figcap

構文

```
:figcap. text
```

パラメータ

text　　キャプション。

:figcap タグは、図のタイトルを指定するために使用されます。

備考

このタグは、:fig タグの直後か、または、:efig タグの直前に置く必要があります。タイトル用のテキストを、このタグと同じ行か、または、次の行に置くことができます。 タイトルのテキストは、コロン (:) やセミコロン (;) を含んではなりません。:fig タグを参照してください。

例

```
:fig.
カラム 1            カラム 2
------------------------------
カラム 1 テキスト  カラム 2 テキスト
カラム 1 テキスト  カラム 2 テキスト
カラム 1 テキスト  カラム 2 テキスト
カラム 1 テキスト  カラム 2 テキスト
:figcap. 図のタイトル
:efig.
```

---

fn

構文

```
:fn{id=topic-name}.
popup-text
...
:efn.
```

パラメータ

topic-name	脚注のトピック名。
popup-text	表示されるテキスト。

:fn タグは、ユーザが脚注に対するハイパーテキストリンクを選択するときに、ポップアップウィンドウに表示できる脚注を指定するために使用されます。

備考

脚注は、段落、リスト、ハイライト指定、およびアートワークに付けることができます。 脚注に索引エントリを入れることはできません。1 つの脚注は他の脚注が始まる前に終わる必要があります。

このタグは :link タグ (:link タグを参照) とともに使用します。

例

```
:fn id=footnote1.
この脚注は、footnote1 の refid をもつ :link タグ
を使用して、指定されたハイパーテキストリンクでトピックの
先行情報を得ることができます。
:efn.
```

---

freeze

構文

```
.freeze nn
```

パラメータ

nn　　ウィンドウの先頭で凍結され、スクロールされないトピックの行数。

.freeze タグは、常に固定位置に表示されるテキストを指定するために使用されます。

備考

行は、固定行ブロックの一部でなければなりません。

.freeze タグは、トピック内のテキストの前に出現する必要があります。

このタグは、最初のカラムから始まる必要があります。

例

```
.context @main
.freeze 3
<Key>    <Overview>
-------------------------------------------------------------
この行は、このトピック用テキストの最初の行です。
この行はスクロールされます。ただし、この行の上 3 行はスクロールされません。
```

---

h1、h2、h3

構文

heading-tag {id=*topic-name*}
{name=*topic-name*}
[res=*context-no*]
[local]
[global].

heading-text

パラメータ

heading-tag	目次リスト (トピックの結果には影響しない) のトピックエントリを指定する、次の :h1、:h2 や :h3 のどれかを指定します。 :h1 レベル 1 のヘッダーで、目次リストでインデントされません。 :h2 レベル 2 のヘッダーで、4 文字インデントされ、:h1 や他の :h2 ヘッダーが続く必要があります。 :h3 レベル 3 のヘッダーで、8 文字インデントされ、:h2 や他の :h3 ヘッダーが続く必要があります。
topic-name	トピック名に対するトピック名。各トピックに、少なくとも 1 つの *topic-name* を指定する必要があります。 *id=topic-name* の指定は、*name=topic-name* の指定と同じです。
context-no	トピックに予約されたコンテキスト番号。コンテキスト番号を予約するには、/PASS2 指令を指定する必要があります。または、オンラインヘルプビルダが 2 番目のパス (でのビルド) を必要としない .define タグも使用することができます。
heading-text	トピックのタイトルや目次リストのエントリとして現れるテキスト。
local	すべてのトピック名がローカルであることを示します。これは、デフォルトですが、指定することもできます。
global	すべてのトピック名が外部であることを示します。 すべての指定されたコンテキスト名がローカルと外部の両方である場合は、local と global の両方を指定できます。

ヘッダータグは、トピックに対するいろいろなレベルのヘッダーを指定するために使用されます。

備考

ヘッダータグ :h1、:h2、および :h3 は、基本のトピックヘッダーです。ヘッダータグは、新しいトピックを始めるために .context と .topic タグのかわりに使用できます。

例

```
:h1 id=help-menu id=help-option local global. ヘルプメニュー
```

次の例は、上記の例と同じ効果です。

```
.context @help-menu
.context help-menu
.context @help-option
```

---

h4、h5、h6

構文

subheading-tag{id=*topic-name*}
{name=*topic-name*}
[local]
[global].
[*subheading-text*]

パラメータ

subheading-tag	目次リストのトピックエントリのサブレベルを指定する、次の :h4、:h5 や :h6 のどれかを指定します。 :h4 レベル 1 のサブヘッダー。 :h5 レベル 2 のサブヘッダーで、:h4 や他の :h5 レベルのサブヘッダーが続く必要があります。 :h6 レベル 3 のサブヘッダーで、:h5 や他の :h6 レベルのサブヘッダーが続く必要があります。 これらのタグは、トピックを始めるものではなく、どこからでもアクセスできるトピック内のヘッダーを指定するものです。
topic-name	サブヘッダーに対するトピック名。 *topic-name* は、何回でも指定できます。 id=*topic-name* の指定は、name=*topic-name* の指定と同じです。
local	すべてのトピック名がローカルであることを示します。これは、デフォルトですが、指定することもできます。
global	すべてのトピック名が外部であることを示します。
subheading-text	トピックのタイトルとして現れるテキスト。太字で表示され、空白行が続きます。

備考

ヘッダータグ :h4、:h5、および :h6 は、トピック内のサブヘッダーです。これらのタグは、新しいトピックを始めるものではないので、目次リストに入力されません。

:h1、:h2 や :h3 タグでのクロスリファレンスと同様に、これらのタグにクロスリファレンスできます。

例

```
:h1 id=on-line-topic-1 local global. オンラインヘルプシステムの使用方法
    System
...
:h4 id=file-menu. ファイルメニュー
...
:h4 id=options-menu. オプションメニュー
...
:h5 id=option-search. 検索機能
...
:h1 id=on-line-topic2 local global. オンラインリファレンスの
    使用方法

    \a ファイルメニュー\@vfile-menu\v を参照してください。
...
    \a 検索機能\v@option-search\v を参照してください。
```

---

heading

構文

.heading *text*

パラメータ

text　　トピックウィンドウの境界に表示されるヘッダー。

指定されたテキストが、トピックウィンドウの上端の境界の中央に表示されます。

備考

.heading は .topic タグと同じ効果ですが、text は目次リストに出現しません。

このタグは、最初のカラムから始まる必要があります。

例

```
.heading メインメニュー
```

---

i1、i2

構文

:i1[id=*i1-id-name*].*index-text*
:i2[refid=*i1-id-name*].*index-text*

パラメータ

:i1	**1 次索引エントリ。**
:i2	**2 次索引エントリ。このエントリは、i1-id-name に対する索引エントリの下でインデントされます。**
i1-id-name	**1 次索引エントリの名前で、最大 30 文字の長さです。**
index-text	**索引に表示するテキスト。**

備考

索引タグは、索引がビルド時に /INDEX 指令を使用して作成されたときに、トピックに対する索引エントリを指定するために使用されます。:i2 タグを使用して索引エントリに対するサブエントリはいつでも作成でき、1 次索引エントリと同時に指定する必要はありません。

索引タグは、トピックのどこにでも使用できます。トピック内での索引エントリの位置が、トピック番号にしたがって記録されます。索引エントリがオンラインヘルプビューアから選択されると、トピックは適切な位置に整列されます。

この結果、索引は :i1 エントリで最初にアルファベット順にソートされ、次に各 1 次索引エントリに対して :i2 エントリでソートされます。

例

```
.context @main-menu
:i1 id=help-menu. ヘルプメニューオプション
...
:i2 refid=help-menu. 索引
...
:i2 refid=help-menu. ヘルプの使用方法
...
:i2 refid=help-menu. 一般的なヘルプ
```

この結果、次の索引が作成されます。

ヘルプメニューオプション
一般的なヘルプ
索引
ヘルプの使用方法

$if、$else、$end

構文

$if *condition*.

true-text
[$else. *false-text*]
$end.

パラメータ

condition	ブール変数を含む式。*condition* は、ブール演算子 and、or、および not を含むことができます。
true-text	*condition* が真の場合に、ビルドされるソース。
false-text	*condition* が偽の場合に、ビルドされるソース。*false-text* には、さらに条件付きビルドタグを含められます。

備考

condition で使用されるブール変数は、/INCLUDE と /IGNORE 指令で定義されます。条件付きビルドタグを、どのようなレベルでも入れ子にすることはできません。

例

```
.context @cond-build
$if cond-1.
$if cond-2.
:p. 両方の条件は真です。
$else. :p. cond-1 のみが真です。
$end.
$else.
$if cond-2.
:p. cond-2 のみが真です。
$else.
:p. どちらの条件も偽です。
$end.
$end.
```

$ifmdef、$elsem、$endm

構文

```
$defmacro macro-name.
...
$ifmdef macro-param.
true-text
$elsem.
false-text
$endm.
```

```
...
$edefmacro.
```

パラメータ

macro-name	条件付きマクロテキストを含むマクロ名。
macro-param	検査する値のパラメータ。
true-text	*macro-param* が *macro-name* に渡される場合にマクロに含まれるテキスト。
false-text	*macro-param* が *macro-name* に渡されない場合にマクロに含まれるテキスト。

備考

$ifmdef、$elsem、および $endm 構造は、$defmacro と $edefmacro タグ内のみで使用できます。

例

```
$defmacro macro-1.
...
$ifmdef param-1.
:p. パラメータの値は、%param-1%です。
$elsem.
:p. このマクロへの指定されたパラメータはありません。
$endm.
...
$edefmacro.
```

---

im

構文

.im *filename*

パラメータ

filename　ビルドに組み入れられるファイル。

オンラインヘルプのソースファイルに他のファイルを埋め込みます。

備考

.im タグは COBOL の COPY 文のように動作します。

このタグは、最初のカラムから始まる必要があります。

例

```
.context @main-menu
:p. メインメニューについてのテキストは、
他のファイルからソーステキストをインクルードします。
.im e:\help\menus\filemenu.txt
.im e:\help\menus\loadmenu.txt
.im e:\help\menus\savemenu.txt
```

---

index

構文

```
..index index-text
```

パラメータ

index-text	索引に入力されるテキスト。

備考

..index タグは 1 レベル以上の索引を使用できません。 /INDEX 指令が指定されると、指定されたテキストが索引に入力されます。 *:i1*、*:i2* も参照してください。

例

```
.context main-menu
..index メインメニュー
..index メイン、メニュー
:p. これはメインメニューです。
```

---

lines

構文

```
:lines
text
...
:elines
```

パラメータ

text　　表示したいテキスト。

:lines タグで任意の行のセットをマークして、表示時に再フォーマットしないようにするために使用されます。

備考

標準フォントを使用した固定行ブロックを形成します。

これらの行は、標準フォントを使用します。そのため、プロポーショナルフォントを標準フォントとするシステム上では、行がプロポーショナルフォントで表示され、行の並びは維持されません (*:cgraphic* を参照してください)。

例

```
:lines. 次行のテキストは、
        前行上にラップされません。
        これらの行のインデントは維持されます。

ただし、これは                                  1234567890
これと揃いません                - OTTFFSSENT
:elines.
```

---

link

構文

```
:link reftype=hd
[refid=topic-name]
[res=context-no]
[database="filename"]
.hotspot-text
:elink.
```

パラメータ

topic-name	参照されるトピックのトピック名。*topic-name* にサブヘッダーを指定すると、サブヘッダーがアクセスできるようになります。
context-no	参照されるトピックのトコンテキスト番号。*topic-name* または *context-no* をトピックへのリンクに指定できますが、両方を使用することはできません。
"filename"	クロスリファレンスされるトピックを含む、別のオンラインヘルプファイルの名前。これが指定されない場合は、トピックが現在のオンラインヘルプファイルに存在すると仮定されます。引用符 (") でデータベー

ス名を囲む必要があります。

hotspot-text ハイパーテキストリンクとしてトピック内で表示される :elink. タグで区切られた総テキスト。このテキストを選択すると、クロスリファレンスされたトピックが表示されます。

:link タグは、追加情報へのリンクを指定します。リンクをトピックや脚注に指定できます (:fn を参照)。

例

```
.context @main-menu
:p. メインメニューについてのテキスト。
...
.context @secondary-menu
:p. 二次メニューについてのテキスト。
...
.context @file-menu
:p. ファイルメニューについてのテキスト。
:link reftype=hd refid=main-menu. メインメニューについてのテキストを
  参照するために、これを選択してください。:elink.
:p. :link reftype=hd refid=footnote1.脚注の使用方法:elink.
についての追加情報は、
脚注で記述されています。
```

---

list

構文

```
.list
```

.list タグは、トピックがクロスリファレンスリストにあることを示すために使用されます。

備考

トピックは固定行トピックで、フォーマットタグは使用できません。各行は、他のトピックへのクロスリファレンスを形成します。ホットスポットを通常の方法で、各行の始めや行の最初の語句に指定することで、クロスリファレンスされたトピックが識別されます。

このタグは、最初のカラムから始まる必要があります。

トピックが表示されると、最初の行が全体的にハイライトされます。これは選択バーで、必要な行へ移動できます。**Enter** キーを押して、トピックを選択します。

---

lm

構文

```
:lm margin=n.
```

:lm タグは、左余白の幅を *n* 文字に設定するために使用されます。左余白は、テキストウィンドウの左端から *n* 文字に設定されます。デフォルトは 0 です。

例

```
:lm margin=2.
```

---

lp

構文

```
:lp. list-text
```

パラメータ

list-text	リストの順序を中断しないリスト内のテキスト。テキストは、現在のリスト項目の左余白から始まります。

備考

:lp タグは、たとえば、リストの順序を中断しない項目の説明のように、リストへテキストを追加するために使用されます。

例

```
:ol.
:li. 項目 1
:li. 項目 2
:lp. 項目 2 の説明
:li. 項目 3
 :eol.
```

$macro

構文

```
$macro macro-name [macro-param=[']value['] ... ].
```

パラメータ

macro-name	定義済みマクロの名前。
macro-param	マクロ *macro-name* のパラメータ名。
value	*macro-param* を *macro-name* に渡すための値。これには、他のマクロへの呼び出しを含めることができます。

備考

マクロを呼び出すと、オンラインヘルプビルダは、指定したマクロパラメータを追加して、そのマクロで定義されたテキストをオンラインヘルプのソースファイルにインクルードします。

例

マクロの使用方法に関する次の例は、前述の $defmacro で説明されたマクロの定義を使用しています。

```
$macro topic id=t1 title='トピック 1'
    text='これは最初のトピックです。'.

$macro topic id=t2 title='トピック 2'
    text='これは 2 番目の
トピックです。 $macro macro-1. $macro
macro-2 a=1 b="xyz" c=2.'.
```

これらのマクロを使用すると、オンラインヘルプのソースファイルに、次を入力する場合と同じ効果があります。

```
:h1 id=t1.トピック 1
:p. これは最初のトピックです。

:h1 id=t2.トピック 2
:p. これは 2 番目のトピックです。
:p. このテキストは、macro-1 の結果です。パラメータはありません。
:p. このテキストは、macro-2 の結果です。パラメータは、1、xyz、2 です。
```

---

note

構文

```
:note [text="note-heading"].
note-text
```

パラメータ

note-heading　　備考のヘッダーに使用されるテキスト。これを指定しない場合は、デフォルトの Note になります。

note-text　　単一段落のテキスト。

:note タグは、テキストに単一段落の備考を始めるために使用されます。

備考

備考のヘッダーは、テキストの左余白に置かれ、備考テキストが同じ行に続きます。備考が次の行へラップする場合は、備考のヘッダーに並びます。リスト内で備考を始めると、テキストはリスト項目の余白に並びます。備考は、ソースファイルの次のタグで終了します。

例

```
:note text="New Note Title:".
この備考のタイトルは「新しい備考タイトル：」と
これに続く備考の残りです。
:p. このタグまたは他のタグで、備考を終了します。
```

---

nt

構文:

```
:nt [text="note-heading"].
note-text
...
:ent.
```

パラメータ

note-heading　　備考のヘッダーに使用されるテキスト。これを指定しない場合は、デフォルトの Note になります。

note-text　　備考のヘッダーの下に含めるテキスト。

:nt タグは、テキストに複数段落の備考を指定するために使用されます。

備考

備考のヘッダーは、テキストの左余白に置かれ、備考テキストが同じ行に続きます。続くテキスト行は、最初の行の始まりに並び、備考のヘッダーの右に並びます。新しい段落を始めるために、備考内に :p タグを使用できます。段落やリスト内で、:nt タグを使用できます。他の備考を始める前に、備考を終了する必要があります。

例

```
:nt text="Long Note:".
この備考のタイトルは「長い備考：」と
これに続く備考の残りです。
:p.これは、備考の 2 番目の段落です。
:p.これは、備考の 3 番目の段落です。
:ent.
```

---

ol

構文

```
:ol [compact].
:li. text
...
:eol.
```

パラメータ

text　　リスト項目。

順序付きリストは、各項目に番号が付けられていることを除けば、順序なしリストと同じです。入れ子にされたリストがある場合は、順序付きリストのレベル 1、レベル 3、レベル 5 などは、文字で 2、4 などと番号が前に置かれます。

compact を使用して、リスト内の項目が空白行で区切られないように指定できます。

---

p

構文

```
:p. text
```

パラメータ

text　　段落テキスト。

:p. タグは、新しい段落の始まりをマークするために使用されます。

備考

すべての段落は、このタグで始まります。このタグと次のフォーマットタグまたはトピックの末尾の間のすべてのテキスト行は、段落の一部として扱われます。 段落は 1 つの単位として扱われ、ソースファイルからの行情報は無視されます。 段落が表示されると、空白行が続き

ます。他のフォーマットタグが直後に続く :p. タグは、空の段落と見なされ、単に継続の空白行として表示されます。

段落テキストを、同じ行でタグの直後に続けられます。タグとテキストの間の空白文字は、テキスト行の先頭にある空白文字と同様に無視されます。

備考:.context タグに続く最初のテキスト行に :p. タグが先行していない場合は、:lines. が最初のタグであるように、固定行のトピックが仮定されます。これらのトピックは、トピックウィンドウに合うように再フォーマットされません。このようなトピックでのフォーマットタグの使用は認められていません。

例

```
:p. これは、語句と各行の間に
単一の空白文字で入力された
通常の段落です。
:p. これは、異なるインデントをもついくつかの行を
        繰り返すテキストの通常ではない段落です。
                ただし、この段落は、画面上にフォーマットされたときに、
    すべての語句の間が単一の空白文字をもつ
 1 つの段落となります。
```

---

parml

構文

```
:parml [compact] [tsize=nn] [break=xxxx].
:pt. term
:pd. definition-text
:pt. term
:pd. definition-text
...
:pt. term
:pd. definition-text
:eparml.
```

パラメータ

nn	用語に割り当てられる文字数。定義段落は、この値によってインデントされます。デフォルトは 10 です。
xxxx	用語とその定義の間隔。

term　　用語として表示されるテキスト。
definition-text　　定義として表示されるテキスト。

xxxx に使用可能な値は、次のとおりです。

none　　定義の最初の行は、常に用語と同じ行に表示されます。用語が *nn* より長いと、1 行空けられて定義が続きます。
fit　　用語が *nn* より短いと、定義の最初の行が用語と同じ行に続きます。用語が *nn* より短くない場合には、定義の最初の行は次の行から始まります。
all　　定義の最初の行は、常に用語に続く行に表示されます。これは、デフォルト設定です。

compact を使用して、リスト内の項目が空白行で区切られないように指定できます。

備考

リスト内の項目は、2 つの部分で構成され、それぞれ独自のタグが付きます。用語は :pt. タグで始まり、説明は :pd. タグで始まります。説明には、追加の段落、ブロック、他のリストを続けることができます。リストを終了するには、:eparml. タグを使用します。

---

popup

構文

```
.popup
```

.popup タグは、他のウィンドウの一番上に表示されるポップアップウィンドウであることを示します。ポップアップウィンドウは、スタックされません。

このタグは、最初のカラムから始まる必要があります。

ポップアップウィンドウを参照できません。また、ポップアップウィンドウにホットスポットを使用することはできません。ポップアップウィンドウから使用できるトピックは、ポップアップウィンドウを呼び出したトピックのみです。ポップアップウィンドウはオンラインプログラムでのみ使用可能で、段落テキストに定義できません。

例

```
.context @main-menu
:p. メインメニューについてのテキスト。
\a 二次メニュー用のポップアップ\v@secondary-menu\v
...
```

```
.context @secondary-menu
.popup
:p. 二次メニューについてのテキスト。 これは、ポップアップ内に表示されます。
```

---

rm

構文

```
:rm margin=n.
```

:rm タグは、右余白の幅を *n* 文字に設定するために使用されます。右余白は、テキストウィンドウの右端から *n* 文字に設定されます。デフォルトは 0 です。

例

```
:rm margin=5.
```

---

sl

構文

```
:sl [compact].
:li. text
...
:esl.
```

パラメータ

text　　リスト項目。

備考

単純リストは、行のリストです。リストの行は、ウィンドウ幅が変更されると再フォーマットされます。単純リストは、常に現在の左余白で始まります。単純リスト内の単純リストは、4 文字インデントされます。

---

title

構文

```
.title title-text
```

パラメータ

title-text オンラインヘルプファイルのタイトル。タイトルは最大 32 文字の長さです。

.title タグは、オンラインヘルプファイルにタイトルを与えるために使用されます。

備考

タイトルは、最初のトピックが始まる前に出現し、最後のタイトルはファイルのタイトルになります。 このタイトルは、オンラインヘルプビューアでファイル名とともに表示されます。

このタグは、最初のカラムから始まる必要があります。

例

`.title アプリケーションヘルプファイル`

---

topic

構文

`.topic` *topic-text*

パラメータ

topic-text トピックのタイトル。タイトルは最大 32 文字の長さです。

.topic タグは、トピックのタイトルを指定するために使用されます。

備考

タイトルは、オンラインヘルプビルダのトピックウィンドウの上部境界の中心に表示されます。 /CONTENTS 指令が指定されていると、目次リストが追加されます。

このタグは、最初のカラムから始まる必要があります。

.topic タグが指定されていない場合は、最初の .context タグに使用されているトピック名が使用されます。

例

`.topic メインメニューの説明`

---

ul

構文

```
:ul [compact].
:li. text
...
:eul.
```

パラメータ

text　　リスト項目。

備考

順序なし (または、箇条書き) リストは、複数の段落を構成する項目のリストです。各項目の最初の行が、他の順序なしリスト内に埋め込まれていない場合 (この場合は、「-」で始まります) は、「o」で始まります。 箇条書きは、1 文字インデントされ、現在のインデントから 4 文字インデントされます。

compact を使用して、リスト内の項目が空白行で区切られないように指定できます。

---

userdoc

構文

```
:userdoc.
...
:euserdoc.
```

パラメータ

なし

:userdoc タグは、ビルドされるソースファイルを識別するために使用されます。

備考

このタグは、ソースファイルの最初のタグである必要があります。 オンラインヘルプビルダに、その後に続くタグ付きテキストをコンパイルするように指示します。タグ付きテキストの終わりは、ソースファイルの最後のタグである :euserdoc タグで識別されます。

例

```
:userdoc.
.
.
```

.
:euserdoc.

---

warning

構文:

```
:warning [text="warning-heading"].
warning-text
...
:ewarning.
```

パラメータ

warning-heading	警告のヘッダーを変更します。ヘッダーを指定しない場合は、テキスト「Warning:」が画面上に表示されます。
warning-text	警告として表示されるテキスト。

:warning タグは、警告メッセージを表示します。

備考

テキストは、メッセージと同じ行に表示されます。:caution を参照してください。

例

```
:warning.
これは、「Warning:」のヘッダーをもつ警告メッセージです。
:ewarning.
:warning text="Error Condition".
これは、「誤り条件 :」のヘッダーをもつ
エラーの可能性がある警告メッセージです。
:ewarning.
```

---

xmp

構文

```
:xmp.
text
...
:exmp.
```

パラメータ

text　　モノスペースフォントで表示するテキスト。

:xmp タグは、入力されたとおりに、モノスペースフォントでテキストを表示するために使用されます。

備考

このタグは、プログラムの例や、テキストを単一文字または数字を揃えたいときに使用されます。テキストは、ウィンドウの左余白から 2 文字インデントされて始まります。テキストサイズがウィンドウサイズより大きい場合には、テキストはクリップされます。:cgraphic. と :fig タグを参照してください。

例

```
:xmp.
$set ans85
 identification division

 program-id. new-program

 environment division
:exmp.
:xmp.
カラム番号        カラム説明
-----------------------------------
1                 説明 1
2                 説明 2
3                 説明 3
4                 説明 4
5                 説明 5
:exmp.
```

第 17 章 : Ohbld エラーメッセージ

ここでは、オンラインヘルプビルダで出力されるエラーメッセージを説明します。

オンラインヘルプビルダが出力するエラーメッセージには、次のような重大度レベルがあります。

- 致命的 - 回復不能なエラーを示します。 ヘルプファイルのビルドが停止します。
- エラー - オンラインヘルプビルダが理解できないタグまたは指令を示します。 ヘルプファイルのビルドは継続しますが、エラーの構文は無視されます。
- 警告 - タグや指令が、構文的には正しくても、何らかのエラーがあることを示します。

重大度レベルは、エラーメッセージ番号の最後の文字で示されます。

WARNING 指令で、警告の通知を無効にできます。

オンラインヘルプビルダが完了すると、各カテゴリごとのエラーの合計数が出力されます。

OHB001F Illegal command line

- 指定されたオンラインヘルプビルダコマンド行が正しくありません。 ファイル指定の誤りや正しくないパラメータが指定された可能性があります。

OHB002F Failed to open source file *source-filename*

- ソースファイルを開けませんでした。通常、指定されたファイルが見つからなかったためです。

OHB003E Unknown Ohbld directive

- オンラインヘルプビルダコマンド行で不明な指令の指定を試行しました。

OHB005E Failed to open /DEFINE file *def-filename*

- /DEFINE 指令で指定したファイルを開けませんでした。ディスク領域の不足や指定された名前のファイルは存在するが、他のプロセスでロックされているか、または、読み取り専用になっている可能性があります。

OHB006E Failed to open /CPY file *cpy-filename*

- /CPY 指令で指定したファイルを開けませんでした。ディスク領域の不足や指定された名前のファイルは存在するが、他のプロセスでロックされているか、または、読み取り専用になっている可能性があります。

OHB007E Failed to open /LIST file *list-filename*

- /LIST 指令で指定したファイルを開けませんでした。ディスク領域の不足や指定された名前のファイルは存在するが、他のプロセスでロックされているか、または、読み取り専用になっている可能性があります。

OHB011E Heading tag or .context expected

- ソーステキストの最初のタグは、.title、.define、.comment、またはヘッダーのどれかである必要があります。 ソーステキストの最初のタグは、これらのどれかでもありません。

OHB012W .title has no text

- ソースファイルに、.title タグに続くタイトルのテキストがありません。 結果は、.title タグを指定しない場合と同じです。

OHB013W More than one .title specified. The first one will be used

- ソースファイルに、複数の .title タグが含まれています。 最初の .title タグのみが使用されます。

OHB016E Unknown parameter

- タグが無効なパラメータで指定されています。 このエラーのタグの例は、:h1、:h2、:dl.、および :sl が考えられます。

OHB017E This tag is only valid at the start of the source

- .define や .title タグが、最初のトピックの開始後に使用されました。タグをソースの先頭に移動するか、または、トピックをこのタグの後から始まるように移動します。

OHB020W Duplicate .define - ignored

- 同一のトピック名を 2 つの define コマンドで使用しました。

OHB021E Local context name expected after .define

- .define コマンドの最初のパラメータは、ローカルトピック名である必要があります。 つまり、@ 文字が先頭に必要です。

OHB022W .define context name too long. Truncated to 32 characters

- .define コマンドに指定されるローカルトピック名が長すぎます。32 文字に切り捨てられました。

OHB023E .define name has no context number

- .define コマンドにコンテキスト番号が指定されていません。

OHB030E Context name *context-name* is already defined

- 複数の外部トピック名が指定されています。このような使用方法は許されていません。

OHB031E Context no. reserved for *topic-name* is used by another topic

- 2 つの異なるトピックに同一のコンテキスト番号が使用されています。このような使用方法は許されていません。

OHB032W Topic has more than one context number

- トピックが複数のローカルトピック名をもち、それらのうちの 1 つ以上が定義される前に参照されています。複数のコンテキスト番号が同一のトピックを指しているため、参照テーブルで無駄な領域を消費します。

  これを回避するには、各トピックに 1 つのローカル名を使用するか、トピックが現れる前にトピックに定義された最初のローカル名を参照するか、または、使用前に .define タグを使用して同一のコンテキスト番号を付けます。

OHB034E Topic has too many context names

- 1 つのトピックは、最大 32 のコンテキスト番号が許されています。32 を越えるコンテキスト番号は無視されます。

OHB035E Topic has no context number

- :h*n* タグでトピックを定義しましたが、*topic-name* の値を指定していません。

OHB036W Heading level is wrong

- ヘッダータグを正しく指定していません。:h2 タグには、:h1 タグまたは他の :h2 タグが続く必要があります。:h3 タグには、:h2 タグまたは他の :h3 タグが続く必要があります。

OHB040E Hotspot has no text

- ホットスポットにテキストを定義していません。このエラーは、\v タグの直前に \a タグがあると発生します。

OHB042E Illegal reference

- 不正なホットスポット参照を指定されています。

  ホットスポットの末尾に区切りの \v タグがあるかを確認し、2 つの \v タグの間に参照があるかを確認します。

OHB043E *source-filename* line *line-no*: local context *context-name* undefined

- ローカルのトピック名がクロスリファレンスとして使用されていますが、.context コマンドに出現していません。

OHB050W Context name truncated to 32 characters

- トピック名が長すぎます。32 文字に切り捨てられました。

OHB051E In a single pass, context numbers may only be reserved at start

- ヘッダータグの RES オプションは、/PASS2 指令がオンラインヘルプビルダコマンド行で指定されたときのみに使用できます。

OHB063W .topic has no text

- テキストを含まないトピックが見つかりました。このような使用方法は許されていません。

OHB064E Topic is too large

- オンラインファイルに格納されているトピックのテキストは、制御文字を含めて 32 KB 以下である必要があります。このエラーが発生した場合は、トピックサイズの減少化を考慮する必要があります。

OHB065F End of source, no topics found

- オンラインヘルプビルダは指定されたソースファイルの末尾に達しましたが、トピックが見つかりませんでした。
- 正しいソースファイルを指定したかどうか確認してください。

OHB066W Title text is too long. Truncated to 80 characters

- ファイルのタイトルが長すぎます。80 文字に切り捨てられました。

OHB067W Topic size exceeds 32K. It will not be compressed

- .hnf ファイル内のトピックが、圧縮プログラム CBLDC002 で制限されている 32 KB を越えています。トピックがこれより大きいと、圧縮されずに格納されます。これは、オンラインヘルプファイルのサイズのみに影響しますが、トピックの表示には影響しません。

OHB070E Illegal browse chain id

- 参照連鎖に指定できる値は、#1 のみです。

OHB071E .browse local context expected

- browse コマンドの両トピック名は、ローカルトピック名または番号でなければなりません。

OHB090E .command must immediately follow context

- .command タグは、.context、:h1、:h2、または :h3 タグの直後に指定する必要があります。

OHB091E Tag is illegal in a command topic

- コマンドトピックで許されているタグは、:call、:i1、:i2、..index、:exec、および :shell のみです。

OHB093E Tag is only allowed in a command topic

- :call、:exec、および :shell タグは、コマンドトピック内のみで使用できます。

OHB095E Command has no text

- :call や :exec タグをパラメータなしで指定しました。このような使用方法は許されていません。

OHB102E ..index has no text

- ..index タグが、テキストなしで指定されています。タグは無視されました。

OHB111W Attribute found in hotspot text

- テキスト属性は、ホットスポットには許されていません。

OHB113W :ehp tag is incorrect

- 閉じる :ehp?. タグが、関連する開く :hp?. タグなしで見つかりました。 たとえば、ハイライトテキストの開始を :hp1. で指定し、:ehp2 でハイライト表示を解除した場合です。

OHB132W List item is not inside a list

- 先行する start-of-list タグがないリスト項目が見つかりました。:p. タグと見なされます。

OHB136E End of list tag is invalid

- start-of-list タグのない end-of-list が見つかりました。タグは無視されました。

OHB137E End of list tag expected

- end-of-list タグのない start-of-list タグが見つかりました。

OHB138W definition tag expected

- 定義タグ (:dd. や :pd.) ではなく、用語タグ (:dt. や :pt.) が見つかりました。

OHB139W defn list term expected

- 用語タグ (:dt. や :pt.) ではなく、定義タグ (:dd. や :pd.) が見つかりました。

OHB140W Line truncated to 76 characters

- 固定行ブロックの 1 行が長すぎます。76 文字に切り捨てられました。

OHB142W :ecgraphic or :elines expected

- :cgraphic. タグに :ecgraphic. タグがないか、または、:lines. タグに :elines. タグがありません。

OHB150W Boolean variable *variable-name* is undefined

- オンラインヘルプビルダコマンド行で、条件式に /IGNORE や /INCLUDE で設定されていないブール変数を使用しています。

OHB153E Unknown boolean operator

- 条件式で許されているブール演算子は、AND、NOT、および OR のみです。

OHB155E Boolean variable expected

- $if タグの後に、条件が見つかりませんでした。

OHB156E $if at line *line-no* has no $end.

- $end のない $if タグが見つかりました。

OHB160W Full stop ('.') expected after tag

- 「.」のないタグが見つかりました。

OHB161E .br only allowed in text

- .br タグは、段落テキスト内のみ使用できます。

OHB170E :i2 refid *index-entry* is not defined in an :i1 entry

- :i2. タグで指定された *i1-id-name* が :i1. 索引エントリで定義されていません。

OHB171W Index text too long. Truncated to 80 characters

- 索引エントリに定義されたテキストが長すぎます。80 文字に切り捨てられました。

OHB172W Too many index items

OHB200W Failed to open text file

OHB201W Artlink expected

OHB202W Invalid item in artlink file

OHB203W Artwork missing

OHB204W Unknown tag

OHB205W Invalid token

OHB206W Too many browse chains

OHB212F Files limit exceeded

OHB213F Boolean limit exceeded

OHB214F i1 tag limit exceeded

OHB215F i2 tag limit exceeded

OHB216F Macro limit exceeded

OHB217F Globals limit exceeded

OHB218F Restable limit exceeded

OHB219F Topic control limit exceeded

OHB220F Topic text limit exceeded

- トピックに対するテキストが長すぎます。ヘッダータグ (:h1、:h2、など) が正しく記述されていません。

OHB221F HNF file size limit exceeded

OHB222F Local topic limit exceeded

OHB223F im limit exceeded

OHB224F Browse limit exceeded

OHB225F Topic start limit exceeded

OHB226W Line length limit exceeded

第 18 章 : オンラインヘルプの用語集

Net Express のヘルプでは、新しい用語がいくつか使用されています。ここでは、それらの用語の意味を定義します。

ブックマーク

ブックマークは、ユーザ定義の索引エントリです。トピックの任意の場所にブックマークを設定して、名前を付けます。この名前がアルファベット順にブックマークのリストに格納されます。ブックマークは、Hyhelp とオンラインヘルプの実行の間保持されます。アルファベット順のリストから定義されたブックマークにアクセスすると、直ちにそれに付随するトピックが表示されます。頻繁にアクセスするトピックにブックマークを付けると便利です。また、他のトピックを参照している間に、現在の場所に一時的にマークを付ける手段としても使用できます。

参照連鎖

参照連鎖は、オンラインヘルプファイルの作者が互いに連鎖させたトピックのシーケンスです。ユーザは、連鎖を前後に参照できます。この機能を使用して、類似した題目のトピックを互いに接続できます。参照連鎖が定義されない場合は、オンラインヘルプファイル全体が参照連鎖になり、各トピックが物理的に前後のトピックに連鎖されます。

文字の図

トピックを生成するときに、標準の文字を使用して図を作成できます。これらは文字の図と呼ばれます。これらの図の垂直揃えを維持するために、行は :cgraphic および :ecgraphic タグを使用してブロック化されます。アクセスと表示プログラムは、これらの行に対してモノスペースのフォントを指定することで、文字の図のブロックが表示されたときに、垂直揃えが維持されるように保証します。

コンテキスト番号

各トピックは、そのトピックを参照する一意なコンテキスト番号をもっています。コンテキスト番号は、ファイルの各トピックのアドレスを定義する参照テーブルの索引として使用されます。

コンテキスト番号を選択し、.define タグを使用してそれを定義できます。コンテキスト番号は完全に任意のもので、ファイルの構造を反映したものではありません。ソースファイルのトピックの順序が、認識されている唯一の構造です。

複数のコンテキスト番号が同じトピックを示すこともできます。

クロスリファレンスリスト

クロスリファレンスリストは、特殊な固定行のトピックで、各行は他のトピックへのクロスリファレンスを含んでいます。そのようなトピックは、.list タグを使用して定義されます。フォーマット

タグを含むことはできません。クロスリファレンスリストが表示されると、全幅のカーソルが表示されます。このカーソルを使用して、クロスリファレンスしたい行を選択できます。

この行の最初の語句は、ホットスポットとして定義されていない限り、クロスリファレンス名として使用されます。ホットスポットとして定義されている場合には、ホットスポットのクロスリファレンスが使用されます。

外部トピック名

外部トピック名は、外部からトピックがアクセスされるときに使用されます。これらの名前はオンラインヘルプファイルのツリー構造に保持されます。トピック名が、検索や、アクセスや表示システムへのエントリ時に使用された場合は、その名前をツリーで検索します。

固定行ブロック

トピック内の行ブロックは、固定行として扱われます。固定行ブロックは、フォーマットされず、記述されたままに表示されます。垂直揃えの必要がない場合は、そのようなブロックは :lines と :elines タグを使用して定義されます。垂直揃えを維持する場合は、ブロックを文字の図として定義する必要があります。

フレックステキスト

フレックステキストトピックは、水平スクロールを必要としないで、トピックウィンドウ幅に合うように、表示時に動的にフォーマットされます。固定行ブロックは、ウィンドウの右端を越えて拡張できます。

履歴リスト

トピックを表示すると、各トピックのタイトルが履歴リストに記録されます。このリストにアクセスすると、以前に表示したトピックを識別して再表示できます。履歴リストには、現在のセッションで表示された最新の 40 項目が保持されます。

ホームトピック

ホームトピックは、オンラインヘルプファイルを作成するために使用された最初のソースファイル内の最初のトピックです。Hyhelp やオンラインヘルプがファイル名で指定されて、トピック名が指定されていない場合に、ホームトピックが表示されます。さらに、ホーム機能を使用すると、ホームトピックが表示されます。

ホームトピックには、オンラインヘルプファイルに格納されている非常に短い情報の概要が常に含まれ、ユーザは、ホットスポットによってファイルの各部分を表示できます。

ホットスポット

ホットスポットは、トピック内のある領域です。アクティブになると、他のトピックを表示します。ホットスポットは、ハイライト表示を使用して識別されます。

IPF

IPF は、OS/2 Presentation Manager で使用されるヘルプ機能です。 オンラインヘルプシステムでは、IPF タグはコロン (:) で始まり、ピリオド (.) で終わります。

リスト

リストは、特殊な方法で表示されるトピック内の構造です。 各リストは、項目のセット構成され、各項目の始まりは適切なタグによって識別されます。各リストの最初と最後もタグで識別されます。 リスト項目に、さらにリストを含めることができます。 これらの埋め込みリストの表示は、埋め込みの深さを反映します。

単純リスト、順序付きリスト、順序なしリスト、定義リスト、およびパラメータリストなど、いくつかの種類のリストがあります。

ローカルトピック名

ローカルトピック名は、単一のオンラインヘルプファイルからのみ参照できるトピック名です。 ローカルトピック名は、先頭の「アット」文字 (@) で識別されますが、IPF タグには適用されません。 Ohbld は、すべてのローカルトピック名をコンテキスト番号に変換します。 ローカルトピック名を数字で始めることはできません。

オンラインヘルプファイル

オンラインヘルプファイルは、拡張子 **.hnf** をもつファイルで、ソース情報ファイルから Ohbld で作成されます。

タグ

タグは、ソースファイルに埋め込まれる制御語句で、ファイルやトピックが作成される方法を定義します。 一部のタグは、テキストで構造を定義するために使用され、テキストは表示ウィンドウで動的に再フォーマットされます。

トピック

オンラインヘルプは、トピックで構成されます。トピックは、ウィンドウに表示される単一のエンティティです。 スクロールしたり、ページングしたりすることで、すべてのトピックを表示できます。 題目は、クロスリファレンスするためにホットスポットを使用して、いくつかのトピックを含むことができます。Net Express ヘルプは、この方法を使用します。たとえば、題目 PERFORM はいくつかのクロスリファレンストピックで構成されます。

トピック名

すべてのトピックには、少なくとも 1 つの名前が付けられます。 ローカル名は、内部的にトピックを参照するために使用されます。 これらは、コンテキスト番号に変換されます。 外部トピ

ック名は、オンラインヘルプファイルのツリー構造に保持されています。 トピック名が、検索や、アクセスや表示システムへのエントリで使用された場合は、その名前をツリーで検索します。

付録 A : チュートリアル - ACCEPT/DISPLAY

このチュートリアルでは、Net Express を使用した文字ベースのプログラミングを紹介します。**Adsamp.cbl** プログラムでのユーザ向けの文字画面の設定方法と入力方法について説明します。

ここでは、拡張 ACCEPT/DISPLAY 構文 (Adis) を使用して COBOL テキストウィンドウにデータエントリ画面を表示する方法を学びます。これで、ユーザが入力したデータを取り込むことができるようになります。これは、ACCEPT を終了させるのに使用できるキーが、ユーザごとに特定されるということを意味しています。

次の項目を実行するには、そのチュートリアルの手順に従ってください。

1. ADSAMP プログラムの準備
2. ACCEPT を終了するキーの指定
3. データエントリ画面の表示
4. データの取り込み
5. ユーザ入力の問い合わせ

A.1 手順 1 - ADSAMP プログラムの準備

1. [**スタート**]、[**プログラム**]、[**Micro Focus Net Express**] から [Net Express] アイコンをクリックし、 Net Express を起動します。**Adsamp** プロジェクトをロードするために、Net Express で [**ファイル**]、[**開く**] をクリックし **demo** ディレクトリを見つけます。**Adsamp.cbl** をダブルクリックして、プログラムを開きます。
2.  をクリックして、**Adsamp.cbl** をコンパイルします。完了すると、「リビルドの完了」というメッセージが出力領域に表示されます。
3. [**アニメート**] メニューで、[**アニメート開始**] をクリックし、Adsamp のアニメートを開始します。
4. 表示されるダイアログボックスで、[**OK**] をクリックします。プログラムの最初の行がハイライトされ、いつでも開始できる状態になります。

A.2 手順 2 - ACCEPT を終了するキーの指定

ACCEPT/DISPLAY モジュール (X"AF") は、すべてのデータが入力されたときに ACCEPT を終了させるために押すキーを指定できるようにします。このモジュールは、他のキーを無効にすることもできます。

1.  をクリックして、最初の文を実行します。

Set-bit-pairs は現時点で ACCEPT/DISPLAY モジュールに実行させたい機能で、有効にしたいファンクションキーを設定します。

Enable-esc-and-f1 は、Esc と F1 の 2 つのファンクションキーのみを有効にする設定を行う集団項目です。2 番目の基本パラメータは値「1」に設定され、ユーザファンクションキーであることを示しています。

2.  をクリックして、2 番目の文を実行します。

   ここでは、異なる 2 番目のパラメータで同じ機能を使用しました。 **Disable-all-other-user-keys** は、Esc と F1 以外のすべてのファンクションキーを無効にする集団項目です。2 番目の基本パラメータは、ここでも値「1」です。

3.  をクリックして、3 番目の文を実行します。

   この文で、「**/**」キーを有効にし、F1 キーではなく「/h」を入力して、ヘルプを表示できるようにします。

   これは、ファンクションキーというよりは、むしろデータキーです。2 番目の基本パラメータは「3」になります。 チュートリアルの最終で、これがどのように動作するか確認できます。

A.3 手順 3 - データエントリ画面の表示

1. 最初に、  を 2 回クリックしてカーソル位置を設定します。カーソルを最初のフィールドの開始位置に設定することもできます。

   カーソル位置フィールドは、**特殊名段落**の **Cursor is** 句で参照されます。

2.  を 1 回クリックして、プログラムに移動し、再び **display** 文を実行します。

   データエントリ画面は、COBOL テキストウィンドウに表示されます。この段階では、まだデータを入力できません。

A.4 手順 4 - データの取り込み

1.  をクリックして、**accept** 文を実行します。
2. データエントリ画面でフィールドにデータを入力します。
3. **F1** を押します。

   このファンクションキーを特別に有効にしたので、プログラムに戻ります。Esc キー、または「/」キーを押すと同じ動作をしますが、他のファンクションキーまたはデータキーでは ACCEPT を終了しません。

4. [**キータイプ**] を右クリックし、表示されるポップアップメニューで [**"キータイプ"の検索**] をクリックします。

   key-status パラメータは、**特殊名段落**の **crt status** 句で参照され、ACCEPT が終了させられた方法の詳細が含まれています。

5. Adsamp ウィンドウ内の任意の個所を右クリックし、表示されるポップアップメニューで [**戻る**] をクリックします。

A.5 手順 5 - ユーザ入力の問い合わせ

1.  をクリックして、**evaluate** 文を実行します。

   条件「1」は真です。これは、ユーザファンクションキー用のコードです (**Enter** キーを押して ACCEPT を終了すると真となる条件「0」に注意してください)。

2.  を再びクリックして、**key-code-1** をダブルクリックします。

   ファンクションキー **F1** 用のコードである値 1 が表示されます。この段階で、プログラムはどのキーを押すと ACCEPT が終了するか正確に把握しています。

3. そして再び、  を再度押します。

   F1 キーをクリックするという動作は、プログラムにヘルプ画面を呼び出すように指示することです。

4.  をクリックして **display-help-screen** 節に移動します。次に、COBOL テキストウィンドウで、画面本体を表示します。

   ACCEPT/DISPLAY モジュールへの次の呼び出しは、以前に使用した set-bit-pairs 呼び出しとは異なる機能です。 今回は、以前のようにデータを取り込むのではなくて、ユーザから単一文字を取得します。

5.  をクリックして、呼び出しを実行します。

   どのキーを押しても、ヘルプ画面からプログラムへ戻れます。

Adsamp プログラムに慣れてきた場合は、いろいろなキーを使用可能にする **set-bit-pairs** 呼び出しを試行してください。 ここで、**evaluate key-type** 文のブレークポイントを設定します。ループで、毎回異なるキーを押し、これが **key-status** の値に与える影響を確認します。

付録 B : 使用例とサンプルプログラム

ここでは、ここまでに説明した内容を具体的に示す使用例とサンプルプログラムを記載しています。コードのみを見るのではなく、以前の章を参照しながらコードを見てください。

B.1 拡張 ACCEPT/DISPLAY

```
working-storage section.
01 a-screen-text.
          03 cust-name-text              pic x(14) value "Customer name".
          03 filler                              pic x(20).
          03 cust-number-text    pic x(16) value "Customer amount".
01 a-screen-data redefines a-screen-text.
          03 filler                              pic x(14).
          03 customer-name               pic x(20).
          03 filler                              pic x(16).
          03 customer-amount     pic z9.9.
01 ws-customer-amount    pic 99v9.
procedure division.
run-start.
          move zero to customer-amount
          display a-screen-text at line 12 column 1
          accept a-screen-data at line 12 column 1
          move customer-amount to ws-customer-amount
          perform until ws-customer-amount not=zero
        display "Customer amount は、ゼロ以外である必要があります"
                at line 25 column 1 with bell
        display customer-amount at line 12 column 51
                with reverse-video blink
        accept a-screen-data at line 12 column 1
        move customer-amount to ws-customer-amount
    end-perform
    stop run.
```

B.2 ANSI ACCEPT/DISPLAY

```
 working-storage section.
 01 a-field   pic 9999.
 procedure division.
 run-start.
     accept a-field
     display "A-Field=" a-field
```

```
     stop run.
```

B.3 CALL 文

```
call x"AF" using set-bit-pairs
                 parameter-block
```

パラメータは、次のように定義されています。

```
01 set-bit-pairs                  pic 9(2) comp-x value 1.
01 parameter-block.
   03 bit-pair-setting   pic 9(2) comp-x.
   03 bit-map-section      pic x value "2".
   03 bit-pair-number            pic 9(2) comp-x.
   03 filler                                    pic 9(2) comp-x value 1.
```

bit-pair-setting フィールドと *bit-pair-number* フィールドに設定される値は、実行したい関数によって異なります。 これらのパラメータに必要な値については、それぞれ説明されています。

すべての x"AF" 呼び出しでエラーが発生する場合には、*set-bit-pairs* が呼び出しからの戻り値として 255 に設定されます。

B.4 CRT Status 句の構文

```
special-names
        crt status is key-status
```

ここで、*key-status* は、次の定義をもつプログラムの作業場所節にある 3 バイトデータ項目です。

```
 01 key-status.
                 03 key-type              pic x.
                 03 key-code-1          pic 9(2) comp-x.
                 03 key-code-2          pic 9(2) comp-x.
```

ACCEPT 文が実行されると、ACCPET が終了した方法を示すように *key-status* が設定されます。一般に、*key-status* の各項目は、次のように使用します。

key-type　　ACCEPT が終了する方法を指定します。戻り値は次のようになります。
　　"0"　ACCEPT の正常な終了。

"1" ユーザファンクションキーによる終了。

"2" 拡張 ACCEPT/DISPLAY 構文キーによる終了。

"3" 8 ビットデータキーによる終了。

"4" 16 ビットデータキーによる終了。

"5" シフトキーによる終了。

"6" ロックキーによる終了。

"9" エラー。

key-code-1 ACCEPT 操作を終了させたキーの番号を示します。この番号の正確に意味は、key-type の返された値によって変わります。

key-code-2 key-type と key-code-1 が 0 の場合には、key-code-2 は ACCEPT 操作を終了させたキーに対するそのままのキーボードコードを含んでいます。単一キー以外のキーストロークのシーケンスが 1 つの機能を実行するように構成されている場合では、最初のキーストロークのコードのみが返されます。
key-type が 4 の場合は、key-code-2 には ACCPET 操作を終了させた文字の 2 番目のバイトが含まれています。
それ以外の場合は、key-code-2 の内容は未定義です。

B.5 ライブラリルーチンを使用したキャラクタユーザインターフェイスの作成

この例では、テキストと属性の 80 バイトの文字列を画面に書き込みます。テキストは画面の一番上の行に表示されます。

```
working-storage section.
01 screen-position.
    03 screen-row       pic 9(2) comp-x value 0.
    03 screen-col       pic 9(2) comp-x value 0.
01 string-length        pic 9(4) comp-x value 80.
01 character-buffer     pic x(80).
01 attribute-buffer     pic x(80).
procedure division.
    move all "x" to character-buffer
    move all x"70" to attribute-buffer
    call "CBL_WRITE_SCR_CHATTRS" using screen-position
                                       character-buffer
                                       attribute-buffer
                                       string-length
```

B.6 MODE IS BLOCK 句

次の集団項目があります。

```
 01 display-item.
  03 display-item-1     pic x(20).
  03 filler             pic x(35).
  03 display-item-2     pic 9(10).
  03 filler             pic x(105).
  03 display-item-3     pic z(4)9.
```

次の文を実行します。

```
display display-item at 0101 mode is block.
```

display-item が、次のように定義された基本項目のように扱われます。

```
01 display-item          pic x(175).
```

この結果、FILLER 項目の内容も表示されます。

B.7 画面節を使用したキャラクタユーザインターフェイスの作成

ボタンをクリックして Net Express を起動し、デモンストレーションプロジェクトをロードします。

```
      $set ans85

*****************************************************************
* Copyright Micro Focus International Limited 2000.  All   *
* Rights reserved.                                              *
*
* このデモンストレーションプログラムは、Micro Focus           *
* 製品のユーザ向けに提供されています。アプリケーションの一    *
* 部として変更し、配布する場合には、ここで、Micro Focus       *
* 社に原著作権が存在することを正しく記述すれば使用する        *
* ことができます。                                            *
*
*****************************************************************
```

```
*************************************************************
*                                                           *
* ADSAMP.CBL                                                *
*                                                           *
* このプログラムでは、Adiscf を使用してデフォルトの構成が     *
* 選択されていることを仮定しています。                        *            *
*************************************************************

 special-names.
     cursor is cursor-position
     crt status is key-status.

 data division.
 working-storage section.

************************************************
* x"AF" 呼び出しに使用するパラメータ
************************************************

 01 set-bit-pairs                pic 9(2) comp-x value 1.
 01 get-single-character          pic 9(2) comp-x value 26.

 01 enable-esc-and-f1.
     03 filler                    pic 9(2) comp-x value 1.
     03 filler                    pic x value "1".
     03 filler                    pic 9(2) comp-x value 0.
     03 filler                    pic 9(2) comp-x value 2.

 01 disable-all-other-user-keys.
     03 filler          pic 9(2) comp-x value 0.
     03 filler                   pic x value "1".
     03 filler          pic 9(2) comp-x value 2.
     03 filler          pic 9(2) comp-x value 126.


 01 enable-slash-key.
     03 filler                    pic 9(2) comp-x value 1.
     03 filler                    pic x value "3".
     03 filler                    pic x value "/".
     03 filler                    pic 9(2) comp-x value 1.
************************************************
* ACCEPT の終了後に返される状態コード
************************************************
 01 key-status.
     03 key-type                  pic x.
     03 key-code-1                pic 9(2) comp-x.
```

```
     03 key-code-1-x            redefines key-code-1 pic x.
     03 key-code-2                pic 9(2) comp-x.

************************************************
* Cursor-Position は、ACCEPT の終了時に、
* カーソル位置を含んでいる ADIS によって
* 返されます。
************************************************
 01 cursor-position.
     03 cursor-row               pic 99.
     03 cursor-column            pic 99.

************************************************
* プログラムが使用する作業領域
************************************************
 01 work-areas.
     03 wa-name                  pic x(30).
     03 wa-address-line-1        pic x(40).
     03 wa-address-line-2        pic x(40).
     03 wa-address-line-3        pic x(40).
     03 wa-address-line-4        pic x(40).
     03 wa-age                   pic 999 value 0.

 01 exit-flag                    pic 9(2) comp-x value 0.


************************************************
* 画面節
************************************************
 screen section.

 01 main-screen.
     03 blank screen.
     03 line 2 column 27
         value "Typical Data Entry Screen".
     03 line 3 column 27
         value "------------------------".
     03 line 5 column 1 value "name     [".
     03 pic x(30) using wa-name highlight prompt " ".
     03 value "]".
     03 line 7 column 1 value "address  [".
     03 pic x(40) using wa-address-line-1
                                highlight prompt " ".
     03 value "]".
     03 line 8 column 1 value "         [".
     03 pic x(40) using wa-address-line-2
```

```
                                    highlight prompt " ".
     03 value "]".
     03 line 9 column 1 value "         [".
     03 pic x(40) using wa-address-line-3
                                    highlight prompt " ".
     03 value "]".
     03 line 10 column 1 value "         [".
     03 pic x(40) using wa-address-line-4
                                    highlight prompt " ".
     03 value "]".
     03 line 12 column 1 value "age      [".
     03 pic zz9 using wa-age         highlight prompt " ".
     03 value "]".
     03 line 20 column 1 value
        "-----------------------------------------------
-       "---------------------------------------------".
     03 line 21 column 1 value "f1" highlight.
     03 value "=/help".
     03 column 75 value "esc" highlight.
     03 value "ape".

 01 help-screen.
     03 blank screen.
     03 line 1 column 34 value "help screen".
     03 line + 1 column 34 value "-----------".
     03 line 4 value "escape" highlight.
     03 value "     leave this program.".
     03 line 6 column 1 value "f1 or /h" highlight.
     03 value "   obtains this screen.".
     03 line 8 column 1
         value "use cursor keys to move around ".
     03 value "the fields on the screen".
     03 value "enter will".
     03 line + 1 column 1 value "accept the data ".
     03 value " present new blank form to fill in.".
     03 line 24 column 25
         value "press any key to continue ...".

************************************************
* 手続き部
************************************************

 procedure division.
 entry-point section.

* 最初に、必要なキーが有効になっていることを確認します。
```

```
* Escape と F1 キーを有効にします。

     call x"AF" using set-bit-pairs
                      enable-esc-and-f1

* 他のユーザファンクションキーをすべて無効にします。
     call x"AF" using set-bit-pairs
                     disable-all-other-user-keys

* ファンクションキーとして動作するように「/」を設定し、
* ACCEPT 操作を終了します。

     call x"AF" using set-bit-pairs
                     enable-slash-key

* ACCEPT の終了時に、カーソル位置が返されることを確認します。
* ACCEPT が終了しました。 行 1、カラム 1 に設定されました。
* カーソルの初期位置が確実に最初のフィールドの
* 開始位置になります。

     move 1 to cursor-row
     move 1 to cursor-column

* Esc キーが押されるまでループします。

     perform until exit-flag = 1
         display main-screen
         accept main-screen
         evaluate key-type
          when "0"

* ACCEPT 操作が正常に終了しました。
* Enter キーが押されました。 この場合、単純に Work 領域が空白になります。
* 最初のフィールドから再開します。

             initialize work-areas
             move 1 to cursor-row
             move 1 to cursor-column

          when "1"

* ユーザファンクションキーが押されました。他のキーが
* 無効であるため、Esc または F1 キーのどちらかです。

             if key-code-1 = 0
```

```
* Esc キーが押されました。プログラムを終了します。

                move 1 to exit-flag
            else

* F1 キーが押されました。ヘルプ画面が表示されます。
                perform display-help-screen
            end-if

          when "3"

* データキーによって ACCEPT 操作が終了しました。他のキーでこの処理
* が有効になっていないため、"/" になるはずです。
* ここで次の文字を取得して、
* 「H」 または 「h」 のどちらが押されたかを確認します。

            call x"AF" using get-single-character
                             key-status
            if key-type = "3" and
              (key-code-1-x = "h" or
               key-code-1-x = "H")
                perform display-help-screen
            end-if

         end-evaluate
     end-perform
     stop run.

 display-help-screen section.

* ヘルプ画面を表示して、
* キー入力を待ちます。

     display help-screen
     call x"AF" using get-single-character
                      key-status.
```

B.8 プログラムマッピングの変更

次のコードでは、**Back Space** キー (キー番号 14) の動作をカーソルを左に移動する (機能 3) に変更し、**Tab** キー (キー番号 8) で **Tab** 機能 (機能 8) を実行するように変更します。

```
* Back Space キーのマッピングを変更
     move 14 to adis-key-number
```

```
    move 3 to adis-map-byte
    call x"AF" using set-map-byte
                     adis-key-mapping
* Tab キーのマッピングを変更
    move 8 to adis-key-number
    move 8 to adis-map-byte
    call x"AF" using set-map-byte
                     adis-key-mapping
```

B.9 ACCPET を終了する Adis 構成

次は、拡張 ACCEPT/DISPLAY 構文で ACCEPT 操作を終了する構成方法の例です。

```
    accept data-item at 0101
    if key-type="0"
        if key-code-1=48
            display "Terminated by return key"
        else
            display "最後のフィールドへの自動スキップで終了されました"
        end-if
    end-if.
```

B.10 Adis キーの削除

次のプログラムは、**Tab** と **Back Tab** キーをファンクションキーとして動作するように設定します。また、← と → キーを、カーソルがフィールドから出る場合に、ファンクションキーとして動作するように設定します。

```
* Tab (キー 8) と Back Tab (キー 9) キーを次のファンクションキーとして
* 動作するように設定します。
    move 1 to adis-key-setting
    move 8 to first-adis-key
    move 2 to number-of-adis-keys
    call x"AF" using set-bit-pairs
                     adis-key-control
* カーソルがフィールドから出る場合にのみ、
* ← キー (キー 3) と → キー (キー 4) がファンクションキー
* として動作するように設定します。
    move 3 to adis-key-setting
    move 3 to first-adis-key
    move 2 to number-of-adis-keys
    call x"AF" using set-bit-pairs
                     adis-key-control
```

```
     accept data-item at 0101
     if key-type="2"
         evaluate key-code-1
          when 3
             display "左カーソルにより、カーソルがフィールドから出ました
-            "
          when 4
             display "右カーソルにより、カーソルがフィールドから出ました
            "
          when 8
             display "Tab キーが押されました"
          when 9
             display "Back Tab キーが押されました"
         end-evaluate
     end-if.
```

B.11 ファンクションキーとして動作するデータキー設定の検出

```
* 文字 "A"～"Z" を、
* ACCEPT 操作を終了するように設定します。
     move 1 to data-key-setting
     move "A" to first-data-key
     move 26 to number-of-data-keys
     call x"AF" using set-bit-pairs
                      data-key-control
     accept data-item at 0101
     if key-type="3"
         evaluate key-code-1
          when 65
             display "A pressed"
          when 66
             display "B pressed"
          when 90
             display "Z pressed"
         end-evaluate
     end-if.
```

B.12 ユーザファンクションキーの検出

次のコードは、どのファンクションキーが押されたのかを検出します。有効になっているキーが **Esc**、**F1**、および **F10** キーのみであると仮定します。

```
    accept data-item at 0101
    if key-type="1"
        evaluate key-code-1
         when 0
            display "escape was pressed"
         when 1
            display "F1 was pressed"
         when 10
            display "F10 was pressed"
        end-evaluate
    end-if.
```

B.13 ファンクションキーの使用方法 - サンプルプログラム

次は、ファンクションキーを使用するプログラムの記述方法を示した例です。**Esc** キーが使用可能ですが、他のファンクションキーはファンクションキーを押すか、または、オプションの最初の文字の次にスラッシュ (/) を押すことによって選択できることを仮定します。

```
$set ans85
*************************************************
* このプログラムでは、Adiscf を使用してデフォルトの構成が
* 選択されていることを仮定しています。
*************************************************

 special-names.
     cursor is cursor-position
     crt status is key-status.
 data division.
 working-storage section.
*************************************************
* x"AF" 呼び出しに使用するパラメータ
*************************************************

 01 set-bit-pairs          pic 9(2) comp-x value 1.
 01 get-single-character    pic 9(2) comp-x value 26.

 01 enable-esc-and-f1.
    03 filler     pic 9(2) comp-x value 1.
    03 filler     pic x value "1".
    03 filler     pic 9(2) comp-x value 0.
    03 filler     pic 9(2) comp-x value 2.
 01 disable-all-other-user-keys.
    03 filler     pic 9(2) comp-x value 0.
    03 filler     pic x value "1".
```

```
    03 filler     pic 9(2) comp-x value 2.
    03 filler     pic 9(2) comp-x value 126.

 01 enable-slash-key.
    03 filler     pic 9(2) comp-x value 1.
    03 filler     pic x value "3".
    03 filler     pic x value "/".
    03 filler     pic 9(2) comp-x value 1.
************************************************
* ACCEPT の終了後に返される状態コード
************************************************
 01 key-status.
    03 key-type            pic x.
    03 key-code-1          pic 9(2) comp-x.
    03 key-code-1-x   redefines key-code-1 pic x.
    03 key-code-2          pic 9(2) comp-x.

************************************************
* Cursor-Position は、ACCEPT の終了時に、
* カーソル位置を含んでいる ADIS によって
* 返されます。
************************************************
 01 cursor-position.
    03 cursor-row          pic 99.
    03 cursor-column       pic 99.

************************************************
* プログラムが使用する作業領域
************************************************
 01 work-areas.
    03 wa-name             pic x(30).
    03 wa-address-line-1   pic x(40).
    03 wa-address-line-2   pic x(40).
    03 wa-address-line-3   pic x(40).
    03 wa-address-line-4   pic x(40).
    03 wa-age              pic 999 value 0.

 01 exit-flag              pic 9(2) comp-x value 0.
************************************************
* 画面節
************************************************
 screen section.

 01 main-screen.
    03 blank screen.
    03 line 2 column 27
```

```
          value "Typical Data Entry Screen".
       03 line 3 column 27
          value "------------------------".
       03 line 5 column 1 value "name     [".
       03 pic x(30) using wa-name         highlightprompt " ".
       03 value "]".
       03 line 7 column 1 value "address  [".
       03 pic x(40) using wa-address-line-1
                                          highlight prompt " ".
       03 value "]".
       03 line 8 column 1 value "         [".
       03 pic x(40) using wa-address-line-2
                                          highlight prompt " ".
       03 value "]".
       03 line 9 column 1 value "         [".
       03 pic x(40) using wa-address-line-3
                                          highlight prompt " ".
       03 value "]".
       03 line 10 column 1 value "         [".
       03 pic x(40) using wa-address-line-4
                                          highlight prompt " ".
       03 value "]".
       03 line 12 column 1 value "age      [".
       03 pic zz9 using wa-age            highlight prompt " ".
       03 value "]".
       03 line 20 column 1 value
          "----------------------------------
-         "---------------------------------------".
       03 line 21 column 1 value "f1" highlight.
       03 value "=/help".
       03 column 75 value "esc" highlight.
       03 value "ape".
     01 help-screen.
       03 blank screen.
       03 line 1 column 34 value "help screen".
       03 line + 1 column 34 value "-----------".
       03 line 4 value "escape" highlight.
       03 value "     leave this program.".
       03 line 6 column 1 value "f1 or /h" highlight.
       03 value "   obtains this screen.".
       03 line 8 column 1
          value "use cursor keys to move around ".
       03 value "the fields on the screen".
       03 value "enter will".
       03 line + 1 column 1 value "accept the data ".
       03 value " present new blank form to fill in.".
```

```
    03 line 24 column 25
        value "press any key to continue ...".

*************************************************
* 手続き部
*************************************************

 procedure division.
 entry-point section.

* 最初に、必要なキーが有効になっていることを確認します。
* Escape と F1 キーを有効にします。

     call x"AF" using set-bit-pairs
                      enable-esc-and-f1

* 他のユーザファンクションキーをすべて無効にします。
     call x"AF" using set-bit-pairs
                      disable-all-other-user-keys

* ファンクションキーとして動作するように「/」を設定し、
* ACCEPT 操作を終了します。

     call x"AF" using set-bit-pairs
                      enable-slash-key

* ACCEPT の終了時に、カーソル位置が返されることを確認します。
* ACCEPT が終了しました。 行 1、カラム 1 に設定されました。
* カーソルの初期位置が確実に最初のフィールドの
* 開始位置になります。

     move 1 to cursor-row
     move 1 to cursor-column

* Esc キーが押されるまでループします。

     perform until exit-flag = 1
         display main-screen
         accept main-screen
         evaluate key-type
          when "0"


* ACCEPT 操作が正常に終了しました。
* Enter キーが押されました。 この場合、単純に Work 領域が空白になり、
* 最初のフィールドから再開します。
```

```
            initialize work-areas
            move 1 to cursor-row
            move 1 to cursor-column

          when "1"

* ユーザファンクションキーが押されました。他のキーが
* 無効であるため、Esc または F1 キーのどちらかです。

            if key-code-1 = 0

* Esc キーが押されました。プログラムを終了します。

                move 1 to exit-flag
            else

* F1 キーが押されました。ヘルプ画面が表示されます。
                perform display-help-screen
            end-if

          when "3"
* データキーによって ACCEPT 操作が終了しました。他のキーでこの処理
* が有効になっていない場合は、"/" になるはずです。
* 「H」または「h」のどちらが押されたかを確認します。

            call x"AF" using get-single-character
                             key-status
            if key-type = "3" and
             (key-code-1-x = "h" or
              key-code-1-x = "H")
                perform display-help-screen
            end-if

        end-evaluate
    end-perform
    stop run.

display-help-screen section.

* ヘルプ画面を表示して、
* キー入力を待ちます。

    display help-screen
    call x"AF" using get-single-character
```

key-status.