Net Express

キャラクタユーザーインタフェース

Micro Focus NetExpress™

キャラクタユーザーインタフェース

Micro Focus®

第 3 版

1998 年 10 月

Micro Focus は、このマニュアルの内容が公正かつ正確であるよう万全を期しておりますが、このマニュアルの内容は予告なしに随時変更されることがあります。

このマニュアルに述べられているソフトウェアはライセンスに基づいて提供され、その使用および複写は、ライセンス契約に基づいてのみ許可されます。特に、Micro Focus 社製品のいかなる用途への適合性も明示的に本契約から除外されており、Micro Focus はいかなる必然的損害に対しても一切責任を負いません。

Micro Focus® は登録商標です。Form Designer™ 、Micro Focus COBOL™ 、および NetExpress™ は Micro Focus Limited の商標です。

Microsoft® 、Windows® および Windows for Workgroups® は Microsoft Corporation の登録商標です。

Visual Basic™ および Windows NT™ は、Micorosft Corporation の商標です。

IBM® は、International Business Machines Corporation の登録商標です。

UNIX® は、X/Open Company Limited の登録商標です。

序文

本書では、キャラクタユーザーインターフェースの作成に使用する方法について説明します。

表記規則

このユーザー ガイドでは、次の書体や規則を使用します。

- 入力するテキストは、次のように表示します。

  ```
  cat script_name | more
  ```

  斜体テキストはコマンドの一部として入力する変数を表します。

- コマンド行やコード例でオプション入力するテキストは、角かっこ ([]) で囲みます。次の式では、オプションの単語 NOT を入力しない場合、*column_name* は pattern_value に設定されます。一方、NOT を入力した場合、*column_name* は *pattern_value* 以外に設定されます。

  ```
  column_name [NOT] LIKE pattern_value
  ```

- 特定のモデルやオペレーティング システムだけに適用する項や段落については、段落のすぐ前に太字斜体の項目名が記載されています。例えば、次のようになります。

  UNIX
  この段落は UNIX システムだけに適用されます。

目次

第1章　概説

ADISは、拡張ACCEPT／DISPLAY構文用に提供されたランタイム支援モジュールです。標準ANSI ACCEPT／DISPLAY構文とADISの機能を次に示します。

構文の種類	機能
ANSI ACCEPT	データ項目の入力
	曜日、日付、時間のデータ項目への受け取り
ANSI DISPLAY	定数の出力
	データ項目の内容の出力
ADIS	画面位置と画面属性の指定
	データの受け取りおよび表示
	- 基本データ項目
	- 集団項目
	- 画面節項目
	キーボードのキーの構成

第2章　ADIS

ADISは、Screen Sectionと拡張ACCEPT/DISPLAYに提供されるランタイムサポートモジュールです。 syntax. ADISは、その構成ユーティリティADISCFを使用して、アプリケーションの要求に合わせて構成できます。あるいは、COBOLアプリケーションからADISに呼出しをかけて、実行時にADISを構成できます。例えば、ファンクションキーを使用可能にできます。- *「ADIS構成ユーティリティ(ADISCF)」*章を参照してください。

AIDSは標準のANSI ACCEPT構文を超える機能を実現します。

次の表で、標準のANSI ACCEPT および DISPLAY構文を超えるADISの機能を紹介します

構文型	機能
ANSI ACCEPT	データ項目の入力
	曜日、日付、時間のデータ項目への受け取り
ANSI DISPLAY	定数の出力
	データ項目の内容の出力
ADIS	画面位置と画面属性の指定
	データの受け取りおよび表示
	- 個別のフィールドに基礎データ項目
	- 複数フィールドに集団項目[1]
	- screen sectionの項目
	キーボードのキーの構成

[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、FILLERは次のフィールドへ飛越す間を占める文字数を記述します。DISPLAY操作では、FILLERは定数間のスペース数を定義します。FILLERとして定義されている領域は全て、ACCEPTまたはDISPLAY動作の影響は受けません。

これらについては、後の節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るようにすることもできます。

- ADISキーの種類の識別
- ADISファンクションキーの使用

- ADISキーの機能への写像

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	主支援モジュール
ADISINIT	初期化モジュール（ADISが最初にロードされたときにのみ使用）
ADISKEY	キーボード操作モジュール
ADISDYNA	ADIS動的属性モジュール

これらは下の図に示したようなユーザーインターフェースになっています。

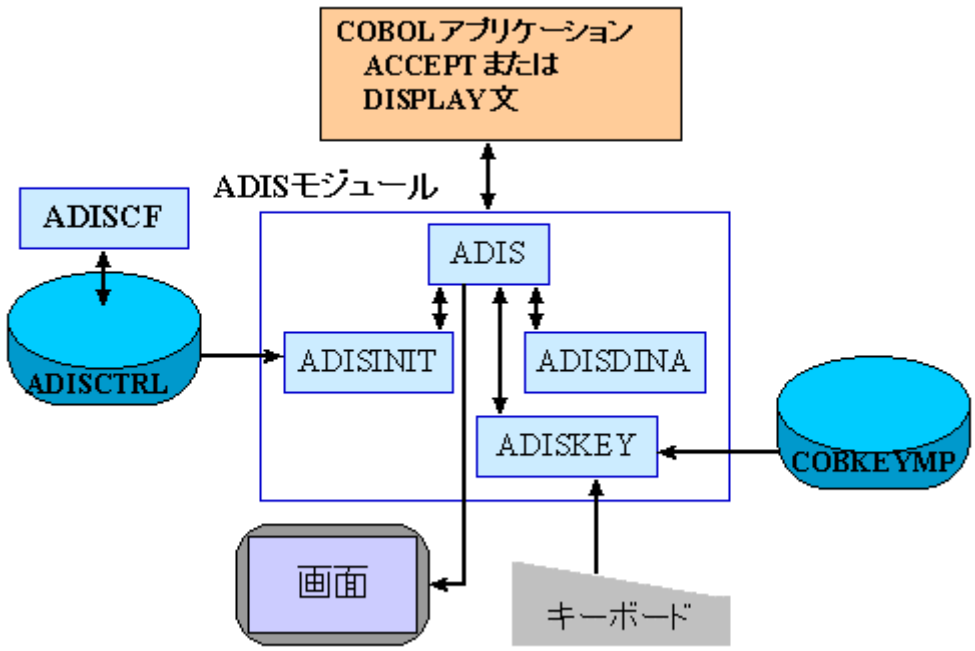


図 Ch2-1: ADIS モジュール

2.1 基本データ項目の単一フィールドからの受け取り

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 単一フィールドを受け取る

実行時に、ACCEPTおよびDISPLAY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 ...
```

ここで

data-item　　プログラムのData Divisionで指定された基本項目

xxyy　　DISPLAYまたはACCEPTフィールドが始まる画面での位置（*xx*=行、*yy*=列）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ステートメントのSIZE句によって決まります。

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形式フィールド全てを意味します。ACCEPTの目的として、英数字編集形式フィールドは、同じ長さの英数字フィールドとして処理されます。斜線文字(／)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。従って、PIC XX0XX0XXと定義されているフィールドは、PIC X(8)と指定されたかのように処理されます。

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"の文字とスペースのみしかそのフィールドに入れることはできません。

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。カーソルキーを用いて、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、後退や削除といった編集機能が提供されています。

2.1.3 数字フィールドおよび数字編集フィールド

データ入力数字フィールドおよび数字編集フィールドの形式は、次のどれかにすることができます。

- 固定形式または定形式化
- 自由形式
- RM様式

ADISFを使用して希望する様式を選択します。データ入力の省略値の形式は固定方式です。どちらの方式が望ましいか、自由方式を用いて試してみる必要があります。特に、単純数字フィールドへの受け取りの場合に必要です。

2.1.3.1 固定形式データ入力

固定形式データ入力は定形式化モードとしても知られています。"What you see is what you get"という表現が本様式を表すのにいちばん適しています。各キーを押すたびに、形式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。従って、ADIS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。

従って、DISPLAY文の場合と同じように、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受け取る場合だけが有効となります。

データ入力中、数字、正号 (+)、負号(-)および小数点 (.)以外の文字は全て拒否されます。編集したフィールド内の挿入文字は、カーソルを前後に動かしたとき、自動的に飛び越されます。+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、符号標識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にのみ、挿入文字として受け付けられます。例えば、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。

単純数字やゼロ抑制をしていない数字編集フィールドのデータ入力は、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入力と同じです。小数点文字を単純数字フィールドで押すと、数字は右詰めされます。例えば、初期値がゼロ列であるPIC 9(5)として定義されたフィールドは、次のように受け取られます(下線の付いた文字はカーソル位置を表す)。

初期表示	00000
1を押すと	10000
2を押すと	12000
4を押すと	12400
後退キーを押すと	12000
"."を押すと	00012

ゼロ抑制したフィールドへのデータ入力の処理は異なります。カーソルはゼロ抑制されていない最初の文字に初め位置しています。小数点より前の桁全てが抑制されている場合、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、小数点に来るまで、数字が入力されるたびにカーソルは右に移動します。整数部が全て埋まるまで、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度に先に進みます。

整数部がまだ全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。

例えば、数字編集データフィールドはPIC ZZZ99.99と定義され、初めゼロが入っているとすると、そのフィールドはACCEPT中次のように表示されます。

初期表示	00.00

1を押すと	10.00
2を押すと	12.00
3を押すと	123.00
4を押すと	1234.00
後退キーを押すと	123.00
5を押すと	1235.00
6を押すと	12356.00
7を押すと	12356.70
8を押すと	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を押すと	10.00
2を押すと	12.00
3を押すと	123.00
"."を押すと	123.00
4を押すと	123.40
5を押すと	123.45

固定方式は、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。このデータ項目が32文字よりも長くなると、次に説明する自由方式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、フィールドもまた自由形式として自動的に処理されます。

2.1.3.2 自由形式データ入力

自由方式は、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を用いて行えます。

自由形式フィールドへのデータ入力中は、フィールドは適当な長さの英数字フィールドとして処理されます。形式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、利用者がそのフィールドを抜けるときだけです。数字、符号文字および小数点以外の文字は全て捨てられます。

フィールドは、記憶域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。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。従ってPIC S99V99と定義されたデータ項目は、固定方式では4文字を占有するのに対し、自由方式では画面上で6文字を占有します。

DE-EDIT"1"を用いてコンパイルを行なってください。その理由は、自由形式では、画面項目の PICTURE句に従わないデータが受け入れられるからです。DE-EDIT"2"を用いた場合、USINGまたはTOデータ項目に転記した場合、ゼロと解釈されます。

データ入力は、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、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Format 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-**様式データ入力**

RM様式データ入力は、ADISCFの該当するAccept/Displayオプションを用いて選択します。RM様式データ入力は、RM/COBOL V2.0での数字および数字編集形式ACCEPT動作をエミュレートするために提供されています。本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入力は全て無視されます。データ入力は英数字フィールドの場合と同じです。ACCEPTおよびDISPLAY文は全て、MODE IS BLOCK句が指定されたものとして動作します。

本様式の数字入力は、本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用に書かれたプログラムにのみ適用するものです。

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、記憶域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符号化体系で、記憶空間と表示サイズが異なる文字を含むことです。

2.2.1 表示されるデータ形式 ― 例

データの説明	画面でのサイズ	コメント
X(5)	5 キャラクタ	
N(5)	10 キャラクタ	各PIC Nキャラクタはメモリで2バイトを占有

G(5)	10 キャラクタ	各PIC Gキャラクタはメモリで2バイトを占有
9(5)	5キャラクタ	
99.99	5キャラクタ	
Z(4)9	5キャラクタ	
99V99	4キャラクタ	Vは明示的に指定していない小数点なので、画面に表示されません。
S9(4)	4キャラクタ	この記号は明示的に指定していないので、画面に表示されません。
S9(4) SIGN LEADING		
SEPARATE	5キャラクタ	
9(4) COMP	2キャラクタ	COMPフィールドはバイナリで格納されます。9(4) COMPフィールドはメモリで2バイトを占有します。

上の例で分かるように、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だけが有効になります。フィールドが表示されるとき、 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。

備考: ANSI DISPLAY動作は、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 単一フィールドを表示する

単一フィールドからACCEPTおよびDISPLAY基本項目を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 ...
```

ここで

data-item プログラムのData Divisionで定義された基本項目。

xxyy DISPLAYまたはACCEPTフィールドが始まる画面での位置（*xx*=行、*yy*=列）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ステートメントの

SIZE句によって決まります。

2.2.3 表示データ内の制御符号列

表示されているデータに、制御符号列を埋め込むことはできません。あえて制御コード（つまり、32未満のASCIIコード）をデータに埋め込むと、そのコードに対応する文字が表示されます。例えば、ASCIIコード7の表示は、警報を鳴らす代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、そのような文字はスペースで置き換えられます。)全ての強調表示、カーソル制御等は、提供される構文を介して行う必要があります。

2.2.4 原文の強調表示

DISPLAY文にCRT-UNDER指定を使用すると、通常表示の原文から強調した表示にする属性を付けて、データ項目が表示されます。利用者の環境によって、CRT-UNDER指定を使用すると、データ項目が下線付き、強調、着色、あるいは、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 集団項目の受け取りまたは表示

集団項目を表示（DISPLAY）または受け取って（ACCEPT）いるとき、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、その基本項目は単に適当なサイズの位置決め項目として使用されます（つまり、FILLER項目への受け取り、またはFILLER項目の表示はできません）。

2.3.1 集団項目への受け取り

集団項目への受け取り（ACCEPT）の際、各フィールドは、本章で後述する単一フィールドACCEPT文の場合のように受け取られます。

CURSOR IS句を用いて、カーソルを明示的に位置づけしないと、カーソルは初め最初のフィールドの先頭に位置します。

フィールドの終わりに近づくと、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を用いて自動飛び越しをオフにすれば防げます。カーソル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(通常タブと後退タブ)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

次のような集団項目を考えてみます。

```
01 表示項目
    03 表示項目1              pic x(20).
    03 filler                      pic x(35).
    03 表示項目2              pic 9(10).
```

```
    03 filler                       pic x(105).
    03 表示項目3                  pic z(4)9.
```

80文字幅の画面上で次の文を実行すると、

```
display 表示項目 at 0101
```

表示項目1は行1、カラム1に表示され、

表示項目2は行1、カラム56に表示され、

表示項目3は行3、カラム11に表示されます。

画面のその他の領域は全てDISPLAY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既に存在するデータは、DISPLAYによって変更されることはありません。

データ項目が次のように定義され、

```
01データ項目
    03データ文字                    pic x occurs 2000.
```

さらに、次の文が実行されると、

```
displayデータ項目 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違うものが表示されます。これを防ぐには、データ項目をPIC X（2000）と再定義して表示するか、あるいは、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持った基本項目であるかのように処理されます。

備考: プログラムをIBM-MS、MS"1"、MS"2"またはRM指令を用いてコンパイルすると、コンパイラは、あたかもMODE IS BLOCKが指定されているかのように集団項目の全てのDISPLAYおよび ACCEPT動作を処理します。

2.4 画面節のACCEPTおよびDISPLAY文

画面節項目を含んだACCEPTおよびDISPLAY動作は、他のACCEPTまたはDISPLAY動作と同様に処理されますが、1つだけ重要な違いがあります。次のコードを考えてみましょう。

```
作業領域 section.
 01 項目A                pic 9(5).
 01 項目B                pic 9(10).
 01 項目C                pic x(10).
 ...
 screen section.
 01デモ画面
    03 blank screen.
    03 line 1 column 1 pic z(4)9 from項目A.
    03 line 3 column 1 pic 9(10) to 項目B.
    03 line 5 column 1 pic x(10) using 項目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01レベル項目（レコード）ごとに作業領域を設定します。従って、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、デモ画面用に設定されます。

次の文が実行されると、

display デモ画面

項目Aと項目Cの内容が、作業場所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。ACCEPTのみのフィールドだからです。

次の文が実行されると、

accept デモ画面

項目Bと項目Cについてのデータが作業領域に受け取られ、次にこれら2つの領域用データは作業領域から作業場所に転記されます。データは項目Aには受け取られません。項目AはDISPLAYのみのフィールドだからです。

ここで注意しなければならない重要なことは、データ部から作業領域への転記は、DISPLAY文を実行したときにのみ発生し、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ということです。従って、2つのACCEPT文が間にDISPLAY文の実行が入ることなしに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って、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、データ部節と画面節のいずれにおいても、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。そうしないと、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形式文字列を持っていた

としても同じことが言えます。

従って、プログラムの次の各行をコード化すると、

```
作業領域 section.
 01 作業場所項目   pic zz9.99 value 1.23
 screen section.
 01 デモ画面
    03 pic zz9.99 using 作業場所項目
 procedure division.
    display デモ画面
```

DISPLAYの結果は不定となります。データ部で画面節ACCEPTまたはDISPLAY文用送り出し項目または受け取り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文にカーソルを配置する

CURSOR IS句で、ACCEPT動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正確に指定でき、ACCEPT動作が終了したらカーソルはカーソル位置にもどります。プログラムでCURSOR IS句を指定しないと、各ACCEPT動作ごとにカーソル位置は最初にかならず第一番目のフィールドの先頭になります。

構文:

CURSOR IS句は特殊名段落において次のように定義されます。

```
special-names.
    cursor is カーソル位置.
```

ここで、*カーソル位置*は作業場所節において次のように定義されるフィールドです。

```
01カーソル位置
    03 カーソル行          pic 99.
    03 カーソルカラム      pic 99.
```

または

```
01カーソル位置
    03 カーソル行          pic 999.
    03 カーソルカラム      pic 999.
```

ここで、

カーソル行　　カーソルが位置する行を指定。有効な数値は1と画面上の行数の間の数です。

カーソルカラム　　カーソルが位置するカラムを指定。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、ADISは*「カーソル位置」*で指定された位置に最初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、*「カーソル行」*または*「カーソルカラム」*に有効な値が入っていない）、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。

*「カーソル位置」*に指定されている値が有効な場合、ADISはフィールド全てを探索して、その中に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点に位置します。もしなければ、最初のフィールドの先頭にカーソルは位置します。従って、カーソル位置を最初のフィールドの先頭にしたい場合は、*「カーソル行」*と*「カーソルカラム」*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上に来る場合、カーソルはその右側にある最初の有効な文字に移動します。もしそれより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、*「カーソル位置」*の値がACCEPT開始時有効であった場合、終了キーが押されると、カーソル位置は*「カーソル位置」*にもどります。しかし、ここで、現在カーソル位置とはかならずしも同じにならないということに注意してください。その理由は、ADISはACCEPT動作の終了時に通常カーソルをフィールドの最後に移動し、相対的に位置づけされたACCEPT文が画面上の正しい位置から開始するようにするためです。

ACCEPTの開始時*「カーソル位置」*の値が無効であった場合、ACCEPTを終了するとき、 *「カーソル位置」*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においてオペレータに要求される操作は、必要な選択肢に対応する画面上の位置にカーソルを移動することだけです。オペレータの選択は*「カーソル位置」*の戻り値で決定できます。

2.5 大画面でのデータの受け取りまたは表示

25行よりも大きな画面を、COBOLシステムで検出できます。

ANSI ACCEPT／DISPLAY文、拡張ACCEPT／DISLAY構文、画面節を使用するプログラムおよびCOBOLシステムライブラリルーチンが、大画面上で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、それ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のであれば、複数のACCEPT／DISPLAYフィールドからはみ出す行は失われます。更に、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文を扱っている画面のFormat 2のWITH句内、およびADISのACCEPTおよびDISPLAY文内で指定します。

2.6 ADISを介してのキーボード処理

このセクションでは以下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。
- ユーザーファンクションキー
- ADISキー。
- ユーザーキーおよびADISキーリストの両方に定義する。
- データキー処理。
- ADIS互換GET SINGLE CHARACTER呼出し。.
- ACCEPT/DISPLAYステートメントのCONTROL句。
- Screen Sectionのサイズ。

2.6.1 ADISによるキーボード操作

ADISキー種類には、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	ユーザーファンクションキー			
カーソルキー	F1…F12	ASCIIコード32-255	Alt	Caps Lock
Tab	Esc		Ctrl	Insert
Backspace			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という定義です。この定義はキーボード上の明示的なファンクションキー(通常F1、F2...F12などと表示されている)と、Esc、カーソルキー、タブ、および後退などというキーを含んでいます。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、特別な意味合いがあります。

これらのファンクションキーは、２つに大別されます。

- ADISキー。ACCEPT文の実行中にADISにより使用されるキーをADISキーといいます。ADISキーには、カーソルキー、タブ、後退、削除およびEnterの各キーが含まれます。

  通常、これらのキーはACCEPT中に定義されたように動作します。カーソル左キーは、カーソルを左に移動し、後退キーは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を終了することはありません。しかし、必要であれば、これらのキーにACCEPTを終了させるようにすることは可能です。

- 利用者ファンクションキー。利用者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書くとき、これらのキーをどういう目的で使用するかを、プログラマが決めるからです。これらのキーに割り当てられた、事前に定義された動作はありません。利用者ファンクションキーには一般に、 F1、 F2 ...F12などの表示の付いたキー、 Escapeキー、ならびに、キーボードにあるそ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。しかし、キーを完全に使用できなくしたり、あるいは、キーを押すことでACCEPT処理を終了させる可能性があります（ファンクションキーのような動作）。

ASCIIコード0～31はデータキーとも、また、ファンクションキーともみなされることが問題を複雑にします。大部分の場合、ファンクション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みなすキーのことです。シフトキーのADISの例としては、 Altキー、Ctrlキーおよび Shiftキーがあります。シフトキーの詳細については、本章の後半のセクション*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、押したときに切り換わるキーのことです。Caps Lock、Insert、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、本章の後半のセクション*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の処理

本節では、この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メソッドについても説明します。 x"B0"ルーチンを使用する、もう1つのメソッドもあります。マシンの種類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するときは、ある種の構成でどのように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衝突*節を参照してください。

2.6.3 どのファンクションキーが押されたかを調べる

アプリケーションにファンクションキーをしようするときは、たいてい、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、下に示したようにCRT STATUS句をプログラムのSpecial-Namesパラグラフに 入れる必要があります。

2.6.3.1 CRT STATUS句

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

ここで、

key-status　　次の定義を持つ、プログラムの作業領域 Sectionの3バイトデータ。

```
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ステートメントが実行されると、*key-status*が、ACCPETが終了した方法を示すように設定されます。*key-status*の個々のフィールドの正確な使用法については、後半で説明しますが、一般に次のように使用します。

key-type　　ACCEPTが終了した方法を示します。戻される値は以下のようになります。

"0" - ACCEPTの正常な終了。
"1" - ユーザー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、この節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。この番号の正確な意味は、*key-type*で戻される値に依存します。

key-code-2　　*key-type*と*key-code-1*が0のとき、*key-code-2*には、ACCEPT操作を終了させたキーの生のキーボードコードが含まれています。単一キーではなくキーストロークのシーケンスで１つの機能を実行するように構成されている場合は、最初のキーストロークだけが戻されます。

*key-type*が4のとき、*key-code-2*には、ACCPET操作を終了させたキャラクタの2つ目のバイトが含まれています。

そうでないときは、*key-code-2*の中身が定義されていません。

CRT STATUS句の詳細については、*COBOL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、「キー種類」には値"0"が、「キーコード1」には次の値が戻されます。

48　　(0を表すASCIIコード)。ACCEPTがEnterキーを押して終了される場合。

1　　画面の最後のフィールドへの自動飛越しでACCEPTが終了される場合。

例

```
accept data-item at 0101
if key-type = "0"
       if key-code-1 = 48
         display "Terminated by return key"
       else
         display "Terminated by auto-skip last field"
       end-if
end-if.
```

2.6.5 標準のユーザーファンクションキー

最大で128までのファンクションキーがあります。COBOLシステムに標準で添付される内容は、オペレーティングシステムの種類によって異なりますが、いくつかのキーは標準です。

キーストローク	ユーザーファンクションキー番号
Escape	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ないこともありますが、あるキーについては上のように構成してください。さらに、以下のユーザーファンクションキーを定義します。

キーストローク	ユーザーファンクションキー
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

キーストローク	ユーザーファンクションキー
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 ユーザーファンクションのキーを有効または無効にする

省略時には、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することもできます。

利用者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。利用者キーが有効になると、そのキーを押すことでACCEPT動作が終了します。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、この章の後半にある*ADISキーを有効または無効にする*節および*ユーザーファンクションキーを有効または無効にする*節を参照してください。

以下の呼出しを使用して選択的にユーザー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
```

ここで、 *set-bit-pairs*および*user-key-control*は、 作業場所節において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.
```

ここで、

user-key-setting	は0に設定されてキーを無効にするか、1に設定されてキーを有効にします。
first-user-key	有効または無効になる最初のキーの番号です。
number-of-keys	有効、無効を設定するキーの続き番号。

ファンクションキーは、別の呼出しx"AF"によって明示的に変更されるまで、あるい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つ目を呼出しをすると、両方のキーが有効になります。

例

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

*** Escape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とF10を有効にします。**

```
move 1 to user-key-setting
```

* F1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を有効にする。

```
    move 10 to first-user-key
    call x"AF" using set-bit-pairs
                     user-key-control
```

2.6.5.2 ユーザーファンクションキーを検出する

ACCPET操作中ユーザー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ときは、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
key-type	"1"
key-code-1	押されたユーザーキーの番号を設定する。
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、1つの利用者ファンクションキーリストを使用するだけで十分です。しかし、COBOLの別の方言からの変換を行う場合、あるプログラムは標準利用者キーリストから戻される値を待っており、また、あるプログラムはCOBOLのある別の方言でファンクションキーにより戻される値を待つ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全く同じです。どのプログラムも、標準利用者キー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。

キー変換の方法ついて詳しくは、この章の後半にある*ADISキーの互換キーリストを使用する*を参照してください。

2.6.5.3.1 ADISキーの互換キーリスト

システムの全プログラムに互換キーリストを使用させたい場合は、Adiscfを使ってその使用を選択することができます。 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
```

ここで、*set-bit-pairs*と*key-list-selection*は、 プログラムの作業領域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.
```

ここで、*key-list-number*が以下の値のいずれかに設定されます。

1　標準のユーザーファンクションキーリストを選択するには

2　互換キーリストを選択するには

2.6.5.4 ユーザーファンクションキーと確認句

通常、FULLまたはREQUIREDのような確認句がACCEPTステートメントで指定されると、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、次のステートメントが実行されると、

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

フィールドに何かを入力しない限り、ACCEPT操作を終了することができません。

ただし、有効に設定されたユーザーファンクションキーがACCEPT操作中に押されると、例外だと見なされ、確認句の条件が見なされていないときでもACCEPT操作が終了されます。確認句の条件を満たしていないときに、Adiscfを使っ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を設定することができます。詳細については、*Adis 構成ユーティリティ（Adiscf）*章の*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

機能を実行するキーとそれらの機能自身の間の区別をしておく必要があります。その理由は、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」写像が存在するからです。これは、ADISキーの１つが実行する機能を、プログラマが変更できることを意味しています。

2.6.6.1 標準ADIS機能へのキーの写像

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。各キーに機能が割り当てられているので、キーが押されると、割り当てら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*節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確認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だけなので、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こなと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	標準のキー/キーストローク
0	ACCEPTを終了	None
1	プログラムを終了	Ctrl+K
2	キャリッジ リターン	Enter
3	カーソル左	Cursor Left ()
4	カーソル右	Cursor Right ()
5	カーソルアップ	Cursor Up ()
6	カーソルダウン	Cursor Down ()
7	Home	Home
8	Tab	Tab
9	Backtab	Backtab
10	End	End
11	次のフィールド	None
12	前のフィールド	None
13	大文字小文字の変更	Ctrl+F
14	文字の消去	Backspace
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

キー番号	機能	標準のキー/キーストローク
19	クリアしてフィールドの終わりに移動	Ctrl+Z
20	フィールドをクリア	Ctrl+X
21	クリアして画面の終わりに移動	Ctrl+End
22	画面をクリア	Ctrl+Home
23	挿入モードを設定	Insert
24	置換モードを設定	None
25	フィールドをリセット	Ctrl+A
26	フィールドの開始位置	None
27	マウスの位置に移動	None

備考:キャリッジリターン（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、拡張ACCEPT/DISPLAY構文キャリッジリターンキーをCRとして、拡張ACCEPT/DISPLAY構文キー "Terminate Accept"を Enterとしてセットアップしてください。

2.6.6.1.1 標準ADISキーの機能

以下は、ADISキーによって実行される機能のリストです。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、状態に応じて様々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ステートメントで指定されているかどうかに応じて、各種の処理があります。

0 - Acceptを終了　　この機能はACCPET操作を終了します。 CRT STATUSフィールド（pic 9 displayとして定義)の最初のバイトはすべて、 キャラクタ"0" (ASCII 48)にセットされ、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 - プログラムを終了　　有効になっている場合は、この機能によって確認中止のメッセージが画面に出力され、ユーザーにキャラクタの入力を求めます。ユーザーが"Y"または"y"以外を入力すると、メッセージが空白になって、通常のACCPET操作の処理が継

続します。ユーザーが"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か終了します。

2 - キャリッジリターン　　カーソルが、画面の次の行の第1列またはその後のフィールド内で、最初のキャラクタ位置となる場所に移動します。該当するフィールドがない場合は、処理は行われません。

3 - カーソル左　　カーソルをフィールド内の前のキャラクタに移動します。現在のキャラクタがフィールド内で最初のキャラクタである場合、カーソルが前のフィールドの最終キャラクタに移動します。現在のキャラクタが画面で最初のフィールドの先頭キャラクタである場合は、エラーがユーザーに伝えられます。

4 - カーソル右　　フィールド内で次のキャラクタにカーソルを移動します。現在のキャラクタがフィールドの最終キャラクタである場合は、カーソルが次のフィールドの先頭キャラクタに移動します。現在のキャラクタは画面で最後のフィールドの最終キャラクタである場合、エラーがユーザーに伝えられます。

5 - カーソル上　　カーソルが次の未保護キャラクタ位置、すなわち上の行で現在の位置のすぐ上にあたるフィールド内の位置に移動します。

6 - カーソル下　　カーソルを、下の行で現在の位置のすぐ下にあたる、次の未保護キャラクタ位置に移動します。

カーソル上/カーソル下（機能5および6）の場合、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項目と同じ規則に従います。このような場合、カーソルが直ちにフィールド内の保護されていない最初のキャラクタ位置にいどうします。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で保護されていない最後のキャラクタ位置に移動します。

7 - 画面の先頭に移動　　カーソルを現在の画面の保護されていない最初のキャラクタ位置に移動します。

8 - 次のタブ位置に移動　　カーソルを次の列のタブ停止位置に移動します。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）の末尾の後ろでは現在の先頭のキャラクタ位置よりも前には移動できません。

9 - 前のタブ位置に移動　　カーソルを前の列のタブ停止位置に移動します。現在のフィールド（または複数行フィールドの行）の開始位置の前では最後のキャラクタ位置よりも後ろに

カーソルは移動できません。

10 - End

以下の順で、カーソルを未保護キャラクタ位置に移動します。

1　複数行の英数字フィールドの最後のキャラクタ位置。

2　現在のフィールドで最後のキャラクタ位置。

3　現在の画面で最終フィールドの最初のキャラクタ位置。

11 - 次のフィールドに移動

カーソルを、画面の次のフィールドで、最初の未保護キャラクタ位置に移動します。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キャラクタ位置に移動し、要求は不成功になったとみなされます。

12 - 前のフィールドに移動

カーソルを、現在のフィールドで最初未保護キャラクタ位置に移動します。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キャラクタ位置に移動します。カーソルが最初のフィールドの先頭キャラクタ位置にすでにある場合は、要求が不成功になったとみなされます。

13 - 現在のキャラクタの大文字小文字を変更

現在のカーソル位置にあるキャラクタを取得し、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。この機能は、数字や漢字のフィールドでは禁止されています。アルファベット以外のキャラクタで大文字小文字を変換する処理は、単純にそのキャラクタをもう一度入力する処理として扱われます。この機能を使って大文字保持機構を無効にすることはできません。大文字保持機構は、キャラクタを小文字に使用とした場合に、単純に大文字に戻すだけです。

14 - Backspaceキャラクタ

論理的に現在のキャラクタ位置の前にある未保護キャラクタ位置にカーソルを移動し、そこにあったキャラクタを再入力バッファに入れて、復元バッファにから引き出したキャラクタで置き換えます。挿入モードのときは、フィールド内に残っているキャラクタの位置を1つだけ移動して、エンドの隙間を埋めるようにオーバーフローバッファからキャラクタを引き出すことによって、削除を行います。

どちらの場合も、関連のバッファが空だったり、次のキャラクタがフィールドで有効でなかったりしたときは、スペースまたはゼロ（フィールドの型による）が使用されます。数値フィールドの小数点位置の左にあるときは処理が多少違いますが、論理的な結果は同じです。キャラクタを入力したとの結果と逆になります。

論理的な要件から生じる異常な結果の1つは、キャラクタ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、キャラクタ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取り出し、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、あるいは、文字が現在フィールドでは違法となる場合、エラーをユーザーに通知します。

16 - 文字挿入

間隔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って、現在カーソル位置に入れます。数字フィールドの小数点の左側の場合を除いて、有意文字は端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、エラーがユーザーに通知されます）。数字フィールドの小数点の左側の場合、有効数字が挿入により失われることになるか、あるいは、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桁の前に挿入することを意味しているからです）。

17 - 文字削除

現在カーソルの位置にある文字を復元バッファに押し出し、フィールド内の残りの文字列を左に1文字位置だけ移動し、終わりのギャップを占有するために1文字オーバフローバッファから取り出します。オーバフローバッファが空の場合、スペースまたはゼロ（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、文字は捨てられ、復元バッファには押し出されません。

18 - 文字復元

この機能の効果と制限は、挿入される文字は復元バッファから取り出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、あるいは、取り出された文字が現在フィールドでは有効でない場合、エラーがユーザーに通知されます。この機能は数字フィールドでは利用できません。

19 - フィールドの終わりまで消去

現在フィールドの現在カーソルカーソル位置およびその右側にある文字列が復元バッファに押し出され、スペース列またはゼロ列（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、後続非有意スペース列およびゼロ列を含みます。カーソルは動きません。複数行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 - フィールドの消去

第1文字位置でフィールドの終わりまでを消去する場合のように、現在フィールド（複数英数字フィールドの場合現在行）の全内容が復元バッファに押し出されます。フィールドへは、それから、スペース列（英数字）またはゼロ列（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置かれます。

21 - 画面の終わりまで消去　フィールドの終わりまで消去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スペース列またはゼロ列が適宜転記されます。

22 - 画面の消去　画面上の全てのフィールドにはスペース列またはゼロ列（適宜）が転記され、カーソルはそのホームポジション（最初のフィールドの最初の非保護文字）に移動されます。

23 - 挿入モードの設定　現在編集モードを挿入に設定し、関連する構成されている標識を表示または消去します。フィールドを外れた標識は本機能により消去されます。

24 - 置換モード設定　現在編集モードを置換に設定し、関連する構成されている標識を表示または消去します。フィールドを外れた標識は本機能により消去されます。

挿入モードと置換モードの概念は英数字フィールドにおいてのみ適用されることに留意しておいてください。モードフラグは、従って、数字フィールドにおいては抑制され、別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ています。

25 - フィールドのリセット（UNDO）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もどします。現在フィールドからの隠れた飛び出し（画面の消去、画面の終わりまで消去）を含むある種の動作は、フィールドタブや自動飛越しのようなより明白な動作と同様に、編集内容がリセットできない状態を解除する場合があることに留意していおてください。

26 - ホーム　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　現在画面の最初の文字位置

2.6.6.1.2 ADISの特殊な写像機能

55 - RMフィールドをクリア　この機能はRMとの互換性のために提供されます。ACCEPT操作時に、UPDATE句が指定されると、カーソルホーム（機能7）が実行されます。指定されていないときは、フィールドをクリア（機能20）が実行されます。

56 - RM Backspace　この機能はRMとの互換性のために提供されます。ACCEPT操作時に、UPDATE句が指定されると、カーソル左（機能3）が実行されます。指定されていないときは、バック

スペースキャラクタ（機能24）が実行されます。

57 - RMタブ　この機能はRMとの互換性のために提供されます。ACCEPT操作時に、UPDATE句が指定されると、ACCEPT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 - 挿入のトグル　現在のモードが挿入モードになっているとき、上書きモード設定（機能24）を実行します。そうでないときは、挿入モード設定（機能23）が実行されます。

59 - 上書きのトグル　現在のモードが上書きモードのとき、挿入モード設定（機能23）を実行します。そうでないときは、上書きモード設定（機能24）を実行します。

60 - 前方タブ　マルチフィールドACCEPTのとき、次のフィールドに移動（機能11）が実行されます。そうでないときは、次のタブ停止位置に移動（機能8）が実行されます。

61 - 後方タブ　マルチフィールドACCEPTのとき、前のフィールドに移動（機能12）が実行されます。そうでないときは、前のタブ停止位置に移動（機能9）が実行されます。

62 - 復元　現在のフィールドが数値である場合や復元バッファが空の場合は、単位置キャラクタ挿入（機能16）が実行されます。そうでないときは、キャラクタの復元（機能18）が実行されます。

255 - 未定義の写像　この値は、1つの機能にどのキーセットアップされていないときに使用します。

2.6.6.1.3 ADISキーを機能へ写像

一般に、ADISキーは同じ名前の機能へ写像されます。従って、<カーソル左> キーはカーソルを左に移動し、後退キーは一つ前の文字を消去するといった具合です。しかし、省略時に異なる機能へ写像されるキーもあります。

キー	機能
キャリッジ リターン	ACCEPT操作を終了します。機能0
Tab	次のフィールドに移動します。機能11
Backtab	前のフィールドに移動します。機能12
挿入モードを設定	トグルを挿入します。機能58

従って、端末で Enterキーを押すと、ACCEPTが終了します。これは、Enterキーが復帰改行へ写像されるからで、復帰改行はACCEPTを終了します。

機能が写像されるので、x"AF"値呼びルーチンを用いて、ADISキー8と9をそれぞれ有効、または無効にするには、キー11と12を参照する必要があることに注意してください。

この段階では、キーを写像するという概念は、不必要な混乱を招きます。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。例えば、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（復帰改行）の写像を0（ACCEPTの終了）から11（次のフィールド）へ変更することで容易に行えます。ADISCFの機能写像画面で、Microsoft COBOL V2.2互換構成を見れば、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互換構成の場合は、RM/COBOLでのキーの動作をエミュレートするために、もっと多くの変更が省略値の写像に含まれています。

キーが255という値に写像されると、ACCEPT中そのキーは何の機能も実行しません。

2.6.6.2 特殊なADIS機能へのキーの写像

拡張ACCEPT/DISPLAY構文キー写像のすべての標準機能は、状況に依存せず、いつも同じ機能を実行します。たとえば、次のフィールドへ移動する機能では、いつも次のフィールドへの移動を試みます。ただし、状況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、この章の前半の*ADISの特殊な写像機能*節で説明しています。

例えば、挿入モード設定キー（キー番号23）は、通常機能58(挿入のトグル)へ写像されます。これは、挿入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ADISキー写像を変更する

次の呼出しを使用します。

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

ここで、 *set-map-byte* と*adis-key-mapping*は、 プログラムの作業場所節において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.
```

ここで、

adis-map-byte は、写像するキーの機能番号に設定します。

adis-key-number 変更するキーの蛮行に設定します。

例

次のコードでは、 Backspaceキー（キー番号14）の処理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。

*** Backspaceキーの写像を変更**

```
    move 14 to adis-key-number
    move 3 to adis-mapping-byte
    call x"AF" using set-map-byte
                     adis-key-mapping
```

*** タブキーの写像を変更**

```
    move 8 to adis-key-number
    move 8 to adis-mapping-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衝突

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、このセクションで定義されているファンクションキーの検出メソッドを使用することをお勧めします。

一般に、x"B0"ルーチンを使用して、ACCPET操作を終了させるファンクションキー定義します。ただし、1つだけ制限があります。x"B0"を使う場合、キャリッジリターンキーがACCEPT操作終了機能に写像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。キャリッジリターンの写像を変更する場合や、写像を変更する構成を使用する場合（Microsoft COBOL V2.2またはIBM COBOL 1.0互換）は、x"B0"呼出しによってセットアップされたキーは、ACCEPT操作を終了しません。そのかわりに、キャリッジリターンが写像されている機能を実行します。

2.6.6.5 ADISキーの有効、無効の設定

次の呼出しを使用します。

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

ここで、 *set-bit-pairs* と*adis-key-control* は、 作業場所節において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.
```

ここで、

adis-key-setting 次のように、影響を受けるキーの処理を定義します。

0 キーが無効に設定されています。ACCEPT操作中にキーが押されると、拒否されます。

1 キーが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、キーが標準の処理を行います。これが標準です。

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 は、影響の受ける最初のキーの番号です。

number-of-adis-keys は影響を受ける連続キーの番号です。

2.6.6.6 ADISファンクションをキーを検出する

ADISキーがファンクションキーとして動作するようにセットアップしている場合は、ACCEPT操作を終了して、*key-status*が以下の値で戻されます。

データ項目	設定内容
key-type	"2"
key-code-1	押された拡張ACCEPT/DISPLAY構文キーの番号に設定します。 このキー番号は、キー写像されたファンクションの番号ではありません。
key-code-2	未定義。

例

次のコードはTabとBacktabがファンクションキーとして動作するようにセットアップします。カーソルがフィールドを出るときは･と ･キーがファンクションキーとして動作するようにセットアップします。

*** Tab（キー8）とBacktab（キー9）がファンクションキーとして動作するようにセットアップ。**

```
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だけ、カーソル左（key 3）と**

*** カーソル右（キー4）がファンクションキーとして動作するようにセットアップ。**

```
    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
-           "leave the field"
             when 4
             display "cursor right caused the cursor to
-           "leave the field"
             when 8
             display "the tab key was pressed"
             when 9
             display "the back tab key was pressed"
      end-evaluate
    end-if.
```

2.6.7 ユーザーキーリストとADISキーリストの両方にキーを定義する

一般に、キーはユーザー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されますが、両方で定義されることはありません。ただし、同じキーを両方のリストで定義してはいけないという制限はありません。

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、以下の一連の処理が実行されます。

1. キーはユーザー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、goto 5。

3. ユーザー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*にキー番号を戻します。

5. キーが拡張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ーに通知。

7. 有効、無効、またはファンクションキーとして動作という設定状態に応じて拡張ACCEPT/DISPLAY構文キーを実行します。

2.6.8 データキー処理

データキーは拡張ASCIIキャラクタセットの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、キャラクタが単純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コードを持つキー、つまりコントロールキーです。これらは通常は無効になっています。

2.6.8.1 データキーを制御する

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て、ACCEPT操作を終了することが可能です。これを行うには、次の呼出しを使用します。

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

ここで、*set-bit-pairs*と*data-key-control*は、プログラムの作業領域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、次のようにセットしてください。

data-key-setting　　影響を受けるキーの処理を次のように定義します。

0　キーが無効です。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　キーが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　キャラクタが単純にフィールドに入力されます。これが標準です。

first-data-key　影響を受ける最初のキャラクタ。

number-of-data-keys　影響を受けるキャラクタの番号。

2.6.8.2 ファンクションキーとして動作するようセットアップされたデータキーを検出する

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セットアップされているときは、キーが押されるとACCEPT操作が終了して、*key-status*が次のようにセットアップされます。

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

例

```
* "A"から"Z"までのキャラクタで、
* ACCEPT操作を終了するようにセットアップ。
    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.
```

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できるようにする、ルーチンを提供しています。これらのルーチンについては、以降のセクションで説明します。

UNIX:
ほとんどのUNIX端末では、他の有効なキーと一緒に押されない限り、AltキーやCtrlキーが押されたのを検出することはできません。このため、移植するアプリケーションにこれらのキーを単独で使用しないでください。そのかわり、/a と /c というキーシーケンスを使って、AltキーとCtrlキーをシミュレートすることができます。

2.6.9.1 利用できるシフトキーを判定する

このルーチンでは、固有のイベントとして検出できるシフトキーを探すことかできます。

次の呼出しを使ってプログラムで使用できるシフトキーを判定します。

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

ここで、*adis-function*と*adis-parameter*はプログラムの作業領域で次のように定義されます。

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

ここで、

adis-function　必ず44。

adis-parameter　プログラムで利用できるシフトキーを戻します。*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が最も重要度が低くなります。

Bit	関連のキー
4 - 15	予備
3	Alt
2	Ctrl
1	左のShift
0	右のShift

特定のビットの値1は、関連の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します

このルーチンは、該当がある場合、どのシフトキーが現在押されているかを判定します。

次の呼出しを使って、現在どのシフトキーが押されているかを判定します。

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

ここで、*adis-function*と*adis-parameter*は、プログラムの作業領域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.
```

ここで、

adis-function　必ず46。

adis-parameter　現在どのシフトキーが押されているかを戻します。 *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の重要度が最も低くなります。

Bit	関連のキー
4 - 15	予備
3	Alt
2	Ctrl
1	左のShift
0	Right Shift

特定のビットの値1は、関連のキーが現在押されていることを示します。

2.6.9.3 シフトキーを有効または無効に設定してACCEPTを終了する

標準では、ACCEPT操作中はすべてのシフトキーが無効になっています。または、x"AF"呼出しを使ってキーを取得します。ただし、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設定することができます。

次の呼出しを使って、シフトキーの有効、無効を設定します。

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

ここで、*adis-function*と*adis-parameter*は、次のようにプログラムの作業領域節で定義されています。

```
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.
```

ここで、

adis-function 必ず1にします。

shift-key-setting 次のように影響を受けるキーの処理を定義します。

0　キーが無効になります。キーが押されても無視されます。

1　キーが有効になります。

first-shift-key 影響を受ける最初のキーの数。有効にするイベントには次のように番号が付けられます。

0 - Alt押された
1 - Alt離された
2 - Ctrl押された
3 - Ctrl離された
4 - Left Shift押された
5 - Left Shift離された
6 - Right Shift押された
7 - Right Shift離された

number-of-shift-keys 影響を受ける連続キーの数。

例

次のコードは、ACCEPT操作を終了して、ACCEP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を有効にする**

```
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 pressed"
        when other
             display "Other shift key pressed"
     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できるようにする、ルーチンを提供しています。これらのルーチンについては、以降のセクションで説明します。

2.6.10.1 利用できるロックキーを判定する

このルーチンでは、固有のイベントとして検出できるロックキーを探すことかできます。

次の呼出しを使ってプログラムで使用できるロックキーを判定します。

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

ここで、*adis-function*と*adis-parameter*は、次のようにプログラムの作業領域節で定義されています。

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

ここで、

adis-function　必ず45。

adis-parameter　プログラムで利用できるロックキーを戻します。*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が最も重要度が低くなります。

Bit	関連のキー
4 - 15	予備

3	Ins Lock
2	Caps Lock
1	Num Lock
0	Scroll Lock

特定のビットの値1は、関連の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する

このルーチンは、該当がある場合、どのロックキーが現在有効であるかを判定します。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。

次の呼出しを使って、現在どのロックキーが有効であるかを判定します。

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

ここで、*adis-function*と*adis-parameter*は、次のようにプログラムの作業領域節で定義されています。

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

ここで、

adis-function　必ず47。

adis-parameter　現在どのロックキーが有効であるかを戻します。 *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の重要度が最も低くなります。

Bit	関連のキー
4 - 15	予備
3	Ins Lock
2	Caps Lock
1	Num Lock
0	Scroll Lock

特定のビットの値1は、関連のキーが現在有効であることを示します。

2.6.10.3 ロックキーを有効または無効に設定してACCEPTを終了する

標準では、ACCEPT操作中はすべてのロックキーが無効になっています。または、x"AF"呼出しを使ってキーを取得します。ただし、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設定することができます。

次の呼出しを使って、ロックキーの有効、無効を設定します。

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

ここで、*adis-function*と*adis-parameter*は、次のようにプログラムの作業領域節で定義されています。

```
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.
```

ここで、

adis-function　　必ず1にします。

lock-key-setting　　次のように影響を受けるキーの処理を定義します。

0　キーが無効になります。キーが押されても無視されます。

1　キーが有効になります。

first-lock-key　　影響を受ける最初のキーの番号。有効にするキーには次のように番号が付けられます。

0 -　Ins Lock

1 -　Caps Lock

2 -　Num Lock

3 -　Scroll Lock

number-of-lock-keys　　影響を受ける連続キーの数。

2.6.10.4 受け取ったキャラクタを大文字に変更関するには

以下の呼出しを使用します。

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

ここで

```
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場合にキャラクタが受け取られるようにADISを復元するには、ゼロのビットペア設定を使用します。

第3章　ADISCFを使ったADISの構成

拡張ACCEPT／DISPLAY構文(ADIS)の機能のひとつは、ACCEPTおよびDISPLAY文の動作を別々に構成できるという点です。ADISCFユーティリティを用いて、ADISCFユーティリティが作成したADISCTRLファイル経由で、フルコンフィグレーションもできます。

ADISCTRLはADISモジュール用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、どれも使用可能です。ADISCTRLファイル先頭の記述項により、ADISがどの構成を使用するか決められます。

ADISCTRLファイルには、ADISが必要とするマシンに依存しない情報が全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。

- ACCEPTおよびDISPLAY文がどのように実行されるか
- エラーに応じてADISが出力するメッセージ
- どのキーが有効／無効か

ADISCTRLに保持されているどの構成でも、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の設計となっています。これらのメニューは、いつでも画面の下に表示でき、使用可能なオプションを列挙します。

COBOLシステムと一緒に提供される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格納されています。 ほとんどの場合、これらの構成からADISCFを用いて使用する構成を選択する以外、特にしなければならないことはありません（ADISCFの主メニューで選択オプションを使用）。

COBOLシステムと一緒に提供される各種構成には次のものを含みます。

- 省略時構成

  本構成は、ADISCTRLファイルが見つからなかった場合、実行時に使用されるADIS内構成と全く同じです。ACCEPTおよびDISPLAY文実行中の標準モード処理を提供します。

- RM互換性

  プログラムが最初RM/COBOL V2.0で書かれた場合、本構成を選択する必要があります。本構成は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互換性

  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互換性.

IBM COBOL V1.0をエミュレートするために提供されています。ごく近いエミュレーションとは言えません。特に、数字および数字編集項目へのACCEPT処理において顕著です。

- Microsoft V2互換性

  Microsoft COBOL V2.2をエミュレートするために提供されています。ごく近いエミュレーションとは言えません。特に、数字および数字編集項目へのACCEPT処理において顕著です。

利用者自身の構成を作成したい場合、提供された構成を直接変更しないことを推奨します。代わりに、変更したい構成を別の名前でセーブして、それに変更を加えるようにしてください。

カスタム構成ADISCTRLファイルを作成する場合、実行時プログラム(現行ディレクトリまたはCOBDIRディレクトリ)からそのファイルへのアクセスが可能である必要があります。そうでないと、省略時構成が使用されます。

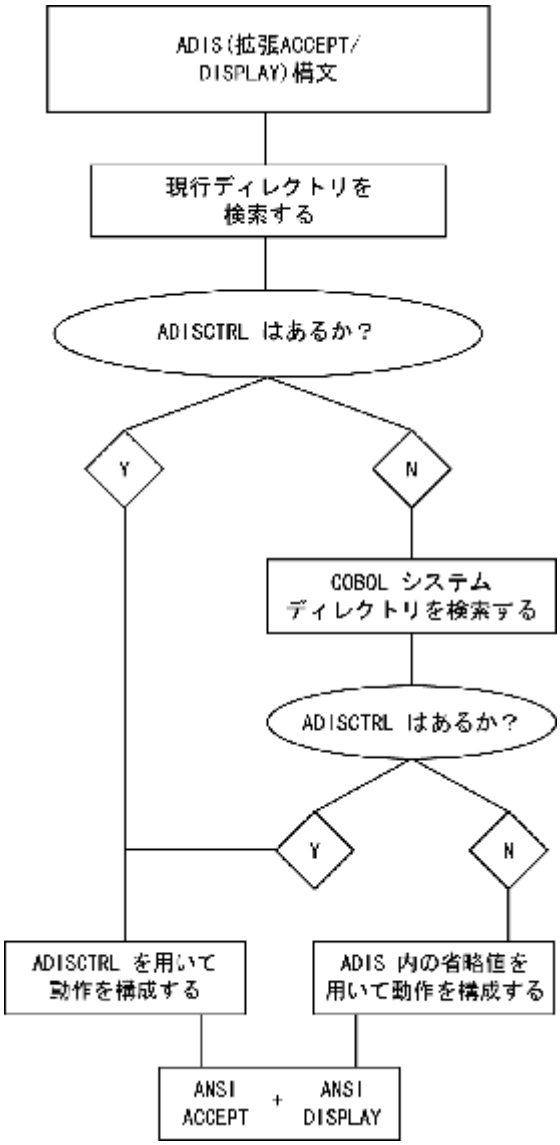
ADISには省略値一式が含まれており、ADISCTRLファイルが存在しない場合は使用されます。この省略値を用いて問題なく動作するアプリケーションはADISCTRLファイル無しで出荷される場合があります。

3.1 ADISCFの呼出し

ADISCFを呼出すには、コマンド行で次のような入力をします。

```
adiscf
```

次の図は、実行時に使用される検索メカニズムを示したもので、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ファイルを使用せずに作業してください。可能であれば、必要な場合にのみ、現行ディレクトリに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、ADISCFが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ると思われますが、ADISCFは古い形式を読み込んで変換することができます。ファイルを新しい形式に一旦変換すると、古いバージョンのCOBOLシステムと提供されたADISは、それらのファイルを以後読み込むことはできません。

インストール中には、省略時の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を忘れないでください。省略時の構成を変更したい場合は、ADISCTRLファイルを現行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。

構成を行うとADISCTRLファイルを変更しますが、ADIS内の省略値は変更しません。

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を押せば、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。そして、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。

このセクション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、現行メニューに関する情報、あるいは主メニューにいるときは現在ロードされている構成に関する情報を列挙します。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主メニュー

ADISCFを呼び出すと、主メニューと図3-1のような情報行が表示されます。

図3-1:ADISCF主メニュー

ADISCF主メニューから多くのサブメニューにアクセスし、以下のような動作を実行することができます。

- 構成の任意のアスペクトを変更する。
- 構成をロードする。
- 構成をセーブする。
- 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。このメニューは、図3-2に示されています。

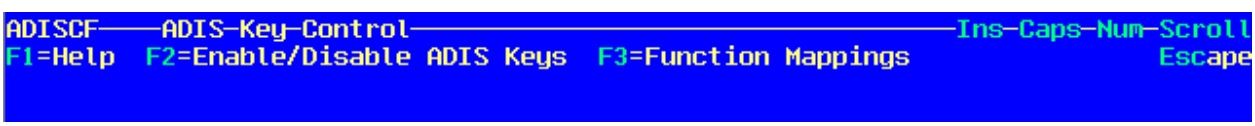


図3-2:ADIS キーコントロールメニュー

このメニューを使うと、以下のような動作を実行することができます。

- 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。

ただし、ADISファンクションキーが編集機能を実行するためのACCEPT処理中に使用されるキーである場合。たとえば、カーソルの移動など。

3.2.1.2 ADISオプション変更メニュー

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

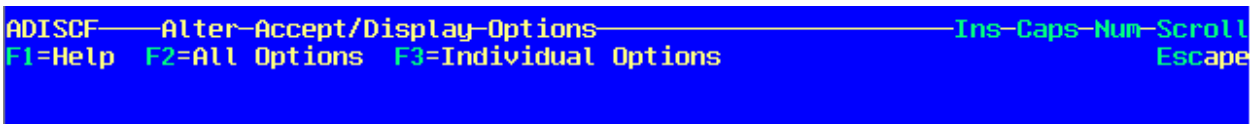


図3-3: ADISオプション変更メニュー

このメニューを使うと、ACCEPT処理の間、カーソルをどのように動かすか、項目をどのような様子にするかを指定することができます。利用者は、使用できるオプションをすべて、あるいは個別に変更するかを選択することができます。

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。

このメニューを使うと、ある条件が生じたとき(カーソルが項目の終端にあるときなど)ADISが表示するメッセージを変更することができます。

3.2.1.4 すべてのオプション変更メニュー

ADIS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。

このメニューを使うと、ACCEPTおよびDISPLAY処理の動作を決定するオプションの集合(ACCEPT処理の間に使用できるデータキーの範囲など)を変更することができます。

3.2.1.5 構成変更メニュー

ADISCF主メニューからF2=Alterを押して変更メニューを選択すると、構成のすべて、または一部を変更することができます。図3-4には、構成変更メニューが示されています。

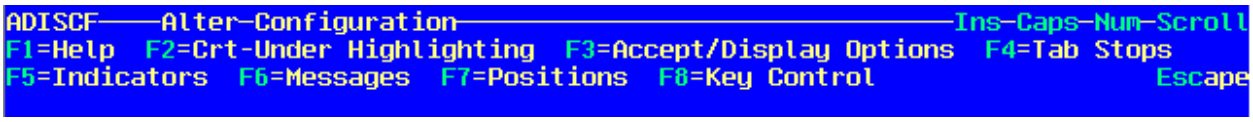


図3-4: 構成変更メニュー

3.2.1.6 Crt-Underハイライト変更メニュー

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

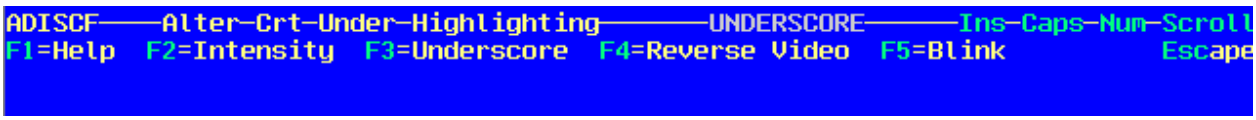


図3-5: Crt-Underハイライト変更メニュー

このメニューを使うと、DISPLAY ... UPON CRT-UNDER文、DISPLAY ... WITH UNDERLINE文で使用されるハイライトのタイプ、あるいは利用者のプログラムの画面セクションでUNDERLINE句が使用されるときのハイライトのタイプを変更することができます。

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。

このメニューを使うと、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するようにすることができます。

3.2.1.8 インディケーター変更メニュー

構成変更メニューからF5=Indicatorsを押すと、インディケーター変更メニューが表示されます。このメニューは、図3-6に示されています。

図3-6: インディケーター変更メニュー

このメニューを使うと、さまざまな状態を示すために、ACCEPT処理の間にADISによって表示されるテキストを変更することができます。各メッセージのテキストの長さは、最大32文字までです。利用者自身のメッセージを入力したら、Enter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。

このメニューには、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横に数字を付けた状態で列挙されます。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 を押します。

3.2.1.10 個別オプション変更メニュー

ADIS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。

このメニューを使うと、ACCEPTおよびDISPLAY処理の使用を管理するオプションの一つを変更することができます(ACCEPT処理の間に有効なデータキーの範囲など)。

3.2.1.11 メッセージ／インディケーター位置変更メニュー

構成変更メニューでF7=Positionsを押すと、メッセージ／インディケーター位置変更メニューが表示されます。

このメニューを使うと、ACCEPT処理の間にADISによって通常表示されるメッセージおよびインディケーターが表示されるかどうかを管理することができます。それらを表示するという選択をすると、このメニューを使ってそれらが表示される画面上の位置を変更することもできます。

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。このメニューは、図3-7に示されています。

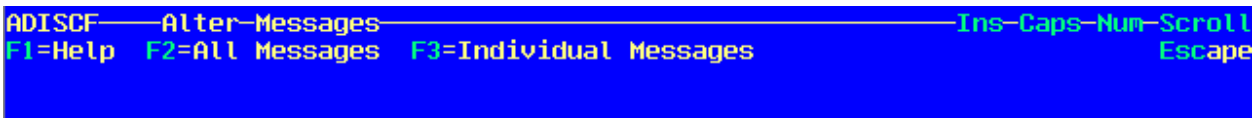


図3-7: メッセージ変更メニュー

このメニューを使うと、さまざまなエラー状態を示すために、ACCEPT処理の間にADISによって表示されるメッセージのテキストを変更することができます。利用者は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。

このメニューを使うと、実行時に使用されるタブストップを96個まで設定することができます。タブストップ画面では、画面の一番上にルーラーが標示され、その上に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。ルーラーにそってカーソルを移動するには、カーソルキー･と･を使用します。

3.2.1.14 構成選択メニュー

ADISCF主メニューからF6=Chooseを押すと、構成選択メニューが表示されます。

このメニューを使うと、利用者が自分の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することができます。

3.2.1.15 構成削除メニュー

ADISCF主メニューから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。このメニューは、この章の後の方に出てくる図3-17に関連画面とともに示されています。

このメニューを使うと、adisctrlファイルから既存の構成を削除することができます。

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。

このメニューを使うと、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することができます。

3.2.1.17 構成ロードメニュー

ADISCF主メニューからF3=Loadを押すと、構成ロードメニューが表示されます。

このメニューを使うと、既存の構成をメモリにロードすることができます。利用者は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しなければなりません。

3.2.1.18 セーブメニュー

ADISCF主メニューからF4=Saveを押すと、セーブメニューが表示されます。このメニューは、図3-8に示されています。

図3-8:セーブメニュー

変更メニューを使って構成を変更したら、このメニューを使って変更をセーブすることができます。新しいバージョンの構成を使用したい場合は、こうする必要があります。構成に変更を加えた後でセーブをせずにADISCFから出ようとすると、本当にそうしたいの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されます。以下の表では、使用可能な機能をアルファベット順に列挙し、詳

細な説明と、それらにアクセスするために押すキーを示してあります。

Add New Tab Stop	F2, F4, F2	現在のカーソル位置にタブストップを設定します。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処理の間は、タブは、"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、次のタブストップにしか移動しません。
Alter ACCEPT/DISPLAY Options	F2, F3	ADISオプション変更メニューが表示されます。
Alter All ACCEPT/DISPLAY Options	F2, F3, F2	一画面ごとに一つのオプションの割合で、各オプションの説明とそのオプションの現行値が順番に表示されます。 Enterを押してある画面から次の画面に移動するか、あるいは Escを押してADISオプションの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、関連の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。
Alter All Messages	F2, F6, F2	一画面に一メッセージの割合で、各メッセージの現行テキストとそのメッセージが出力される条件が順番に表示されます。次の画面に移動するには、Enterを押してください。 メッセージを変更するには、関連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を押して新しいメッセージをセーブし、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を押すと、メッセージ変更メニューに戻り、現行メッセージへの変更は廃棄されます。
Alter Auto Clear Indicator	F2, F5, F4	オートクリアとクリアオートクリアインディケーターの現在のテキストを表示します。オートクリアインディケーターは、ある項目に最初の文字を入力するとき、その文字の入力前にその項目がクリアされるような場合に、ACCEPT処理中に表示されます。クリアオートクリアインディケーターは、ACCEPT処理の最後に表示され、オートクリアインディケーター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、カーソルキー(·と·)を使用します。変更内容をセーブするにはEnterを押し、インディケーターをどちらも変更しないままオートクリアインディケーター変更メニューを出る場合はEscを押します。

Alter Configuration	F2	構成変更メニューが表示されます。
Alter CRT-UNDER Hightlighting	F2, F2	CRT-UNDER ハイライト変更メニューが表示されます。
Alter Indicators	F2, F5	インディケーター変更メニューが表示されます。
Alter Individual ACCEPT/DISPLAY Option	F2, F3, F3	利用者が変更できる個別ADISオプションの画面が表示されます。現在選択されているオプションは、ハイライトされます。さらにオプションのページを見るには、F2=Next Pageを押し、元のオプションページに戻るには、もう一度F2を押します。 変更したいオプションの番号を入力するか、あるいはカーソルキーを使って関連行にカーソルを置きます。Enter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行値が表示されます。オプションの値を変更するには、関連のプロンプトで必要な選択番号を入力し、Enterを押します。個別オプション変更メニューに戻るには、Escを押します。
Alter Individual Messages	F2, F6, F3	変更できるメッセージの画面が表示されます。現在選択されているメッセージはハイライトされます。さらに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を押します。 変更したいオプションの番号を入力するか、あるいはカーソルキー(·と·)を使って関連行にカーソルを移動します。Enterを押して、そのメッセージを選択します。現在そのエラー状況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規メッセージを入力します。Enterを押すと入力したメッセージがセーブされてメッセージのリストに戻り、Esc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。
Alter Insert/Replace Indicator	F2, F5, F2	挿入、置換、挿入／置換インディケーターのクリアの現行テキストが表示されます。挿入インディケーターは、ACCEPT処理の間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。置換インディケーターは、ACCEPT処理の間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／置換インディケーターのクリアは、ACCEPT処理の最後に表示され、他の2つのインディケーターが現在表示されている場合に、それを削除します。 3つの項目の間を移動するには、カーソルキー(·と·)を使用します。Enterを押すと変更内容がセーブされ、Escを押すといずれのインディケーターも変

		更することなく、挿入／置換インディケーター変更メニューから出ます。
Alter Key Control	F2, F8	拡張ADIS構文キーコントロールメニューが表示されます。
Alter Messages	F2, F6	メッセージ変更メニューが表示されます。
Alter Off End of Field Indicator	F2, F5, F3	フィールドのオフエンドの現行テキストが表示され、フィールドインディケーターのオフエンドをクリアします。フィールドインディケーターのオフエンドは、入力される次の文字がフィールドの終端を越える場合に、ACCEPT処理の間に表示されます。フィールドインディケーターのオフエンドのクリアは、ACCEPT処理の最後に表示され、現在表示されているインディケーターを削除します。 2つの項目の間を移動するには、カーソルキー(·と·)を使います。変更内容をセーブするにはEnterを押し、インディケーターを変更せずに挿入／置換インディケーターの変更メニューを出るにはEscを押します。
Alter Position of Messages and Indicators	F2, F7	利用者が位置を変更できるメッセージとインディケーターの4つのカテゴリーと、各カテゴリーに関する位置を入力する項目が表示されます。 画面上である項目から別の項目に移動するには、カーソルキー(·、·、·と·)を使用してください。ACCEPT処理中にエラーメッセージまたは特定のインディケーターを表示させたい場合は、関係する項目に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っ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処理中に中断キーを押すと、中断確認メッセージが必ず表示されます。 メッセージあるいは特定のインディケーターを表示する選択をした場合は、ACCEPT処理中にそれらが画面上に表示される省略時位置を変更することができます。カーソルキーを使ってカーソルを関連項目に置き、メッセージあるいはインディケーターを表示したい新規の行と列を入力してください。 行1列1は、画面の左上の角になります。 同じ位置に2つ以上のインディケーターが表示され、しかも2つ以上が同時に表示されるように構成する場合は、項目のオフエンドとオートクリアインディケーターが、挿入／置換インディケーターより優先されます。

		どのインディケーターについても行255を選択すると、そのインディケーターは、画面の実際の長さにかかわらず、画面の一番下の行に表示されます。 入力したらEnterを押して構成変更メニューに戻り、変効内容をセーブしてください。Escを押して構成変更メニューに戻ると、行った変更はすべて廃棄されます。
Alter Tab Stops	F2, F4	タブストップ変更メニューが表示されます。
Choose Configuration	F6	使用可能な構成が表示され、現在有効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、あるいはカーソルキー(·と·)を使って関連の構成名にカーソルを合わせることによって、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。 選択したい構成をハイライトしたら、Enterを押します。ハイライトされた構成が有効な構成になり、ADISCF主メニューに戻ります。利用者が今度COBOL システムを使ってプログラムを実行したときに、拡張ADIS構文は、自動的に今選択した構成を使用します。代わりに、Escを押せば、新しい構成を選択せずに、ADISCF主メニューに戻ることができます。
Delete Configuration	F5	既存の構成が表示され、現在の構成はハイライトされます。プロンプトで番号を入力するか、あるいはカーソルキー(·と·)を使って、削除したい構成をハイライトさせます。 削除したい構成をハイライトさせたら、Enterを押してください。その構成はadisctrlから削除され、ADISCF主メニューに戻ります。代わりに、 Escを押して、構成を削除せずに、ADISCF主メニューに戻ることもできます。 --- 注意:現在選択されている構成を削除することはできません。これを削除したい場合は、まず選択メニューを使って、代わりの構成を選択する必要があります。 ---
Delete Tab Stop	F2, F4, F3	現在のカーソル位置にあるタブストップ(Tで表される)を削除します。まず、カーソルキー(·と·)を使って、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。

<table>
<tr><td>Enable/Disable Enhanced ACCEPT/DISPLAY Syntax Keys</td><td>F2, F8, F2</td><td>各キーの現在状況とともに、拡張ADIS構文キーによって提供されるすべての機能が一覧表示されます。

どのキーの状態も、以下のうち1つに変更することができます。<table><tr><td>状態</td><td>説明</td></tr><tr><td>D</td><td>無効 - このキーは、ACCEPT処理の間、関連機能を実行しません。</td></tr><tr><td>E</td><td>有効 - このキーは、ACCEPT処理の間、関連機能を実行します。</td></tr><tr><td>F</td><td>ファンクションキー - このキーは、ACCEPT処理の間、ファンクションキーとして動作します。</td></tr></table>カーソルキー(·、·、·と·)を使って、画面上の項目から項目へ移動し、変更したい項目に必要な文字を入力してください。

入力をしたら、Enterを押して、拡張ADIS構文キーコントロールメニューに戻り、変更内容をセーブしてください。Escを押して、拡張ADIS構文キーコントロールメニューに戻ると、行った変更は廃棄されます。</td></tr>
<tr><td>Escape</td><td>Esc</td><td>より高いレベルのメニューに戻るか、ADISCF主メニューにいる場合は、ADISCFから完全に出ます。EscとEnterが両方とも使用できるオプションである場合は、Escは入力した更新をせずにメニューを抜け出し、Enterは更新をしてから出ます。</td></tr>
<tr><td>Load Configuration</td><td>F3</td><td>使用可能な構成を一覧表示します。

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て表示されます。構成を選択するには、プロンプトでその番号を入力するか、カーソルキー(·と·)を使って、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て表示され、画面の一番下のプロンプトの位置にその番号が表示されます。

必要な構成をいったんハイライトさせたら、Enterを押してください。ADISCFが選択された構成をadisctrlからメモリにロードします。

これで、ADISCF主メニューに戻ります。今選択した構成は、情報行に表</td></tr>
</table>

<table>
<tr><td></td><td></td><td>示されます。これで、変更メニューを選択することによって、この構成を変更することができます。

ロードする構成を選択したくない場合は、Escを押すと、ADISCF主メニューに戻ります。</td></tr>
<tr><td>Map Adis Keys</td><td>F2, F8, F3</td><td>サポートされている拡張ADIS構文機能のページを表示します。さらに機能のページを見るにはF2=Next Pageを押し、元の機能のページに戻るにはもう一度F2を押してください。

この機能を使うと、通常キーに関連付けられている機能を変更することができます。どの機能でもどのキーにもマップすることができます。各ファンクションキーには、利用者が変更できる関連項目が3つあります。
<table>
<tr><td>項目</td><td>説明</td></tr>
<tr><td>妥当性検査</td><td>この項目をYに設定すると、ACCEPT処理中にカーソルが項目を出る前に、すべての妥当性検査基準が満たされなければなりません。Nと入力すると、ACCEPT処理中に妥当性検査基準はチェックされません。</td></tr>
<tr><td>位置合わせ</td><td>省略値では、この項目はNに設定されています。Yに設定すると、ACCEPT 処理の間に、カーソルの現在位置から項目の終端まで、項目がクリアされます。数字項目は、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。</td></tr>
<tr><td>マップされた番号</td><td>このフィールドには、関連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。</td></tr>
</table>
さらに機能マッピングのページを表示するには、F2=Next Pageを押してください。

カーソルキー(･、･、･および･)を使って項目間を移動します。変更をした場合は、Enterを押して、拡張ADIS構文キーコントロールメニューに戻り、変更内容をセーブしてください。変更内容をセーブしたくない場合、あるいは何もしなかった場合は、Escを押して、拡張ADIS構文キーコントロールメニューに戻ってください。</td></tr>
</table>

Overwrite Existing Configuration	F4, F3	既存の構成と同じ名前で利用者の構成をadisctrlにセーブします。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、慎重に使用してください。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、あるいはカーソルの上下キーを使って、カーソルを間連行に移動してください。指定した名前に構成をセーブして、ADISCF主メニューに戻るには、Enterを押してください。セーブせずに、ADISCF主メニューに戻るには、代わりにEscを押してください。 adisctrlには、構成を16個までセーブすることができます。この17個以上セーブしようとすると、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` このメッセージが表示されたら、既存の構成を削除するか、既存の構成を上書きしないと、新しい構成をセーブすることができません。
Save	F4	セーブメニューが表示されます。
Save New Configuration	F4, F2	adisctrlに利用者の構成を新規構成としてセーブします。利用者は、この構成の名前を入力するように求められます。Enterを押すと、入力した名前に構成をセーブして、ADISCF主メニューに戻ります。セーブせずに、ADISCF主メニューに戻るには、代わりにEscを押してください。
Save New Tab Stop Settings	F2, F4, F4	利用者がタブストップの位置について行った変更をセーブし、構成変更メニューに戻ります。
Set CRT-UNDER Attribute to Bl	F2, F2, F5	DISPLAY ... UPON CRT-UNDER文が実行されるときに使用される属性が明滅するように指定します。情報行にBLINKという言葉が表示されます。
Set CRT-UNDER Attribute to Bo	F2, F2, F2	DISPLAY ... UPON CRT-UNDER文が実行されるときに使用される属性を太字に指定します。情報行にINTENSITYという文字が表示されます。
Set CRT-UNDER Attribute to Reverse Video	F2, F2, F4	DISPLAY ... UPON CRT-UNDER文が実行されるときに使用される属性をリバースビデオに指定します。リバースビデオの情報行にREVERSE VIDEOという言葉が表示されます。
Set CRT-UNDER Attribute to Underscore	F2, F2, F3	DISPLAY ... UPON CRT-UNDER文が実行されるときに使用される属性を下線に指定します。情報行にUNDERSCOREという言葉が表示されます。

3.4 構成可能なADISオプション

以下に挙げるのは、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利用者が変更できるすべてのADISオプションのリストです。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. ユーザーファンクションキーの有効化／無効化

   利用者がユーザー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、普通、キーボード上のファンクションキーです。以下から選択することができます。

   - ユーザーファンクションキーをすべて無効にします。ACCEPT処理中にユーザーファンクションキーを押すと、それは無効なキー操作として扱われます。これは省略値です。
   - ユーザーファンクションキーをすべて有効にします。ACCEPT処理中にユーザーファンクションキーを押すと、ACCEPT処理が終了させられます。

2. 受け入れられるデータキーの範囲。

   利用者は、ACCEPT処理に入力中に使用できる文字を指定することができます。利用者は、必要なオプションの番号を入力するよう求められます。

   1　0から127までのASCIIコードのある文字が許可されます。

   2　0から255までのASCIIコードのある文字が許可されます。

   3　32から127までのASCIIコードのある文字が許可されます。

   4　32から255までのASCIIコードのある文字が許可されます。これは省略値です。

   1または2のオプションを使って、0から31までの範囲内の文字を有効にする場合は、まだ項目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カーソル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、これらはデータキーよりも優先されます。

3. プロンプト文字。

   画面セクション項目のACCEPT処理の間、あるいはPROMPT句を指定するACCEPT処理の間、項目の空の部分に表示できる文字を指定することができます。システムは、まだデータを入力していない項目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、項目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を除くすべてのピクチャタイプに使用されます。

4. PIC G項目のプロンプト文字。

画面セクション項目のACCEPT処理中、またはPROMPT句を指定するACCEPT処理中にPIC G項目の空の部分に表示される文字を指定することができます。

5. ACCEPTの前に項目を事前表示します。

   利用者がACCEPT文の前に自動的にデータ項目の内容が表示されるようにしたいかどうかを指定することができます。ACCEPT文の前にデータ項目の自動表示を指定しない場合は、画面はそのままです。以下のオプションから選択することができます。

   1　カーソルが数字編集項目に移動したときに、数字編集を使ってそれらの項目の事前表示ができます。他の事前表示は発生しません。

   2　カーソルが数字項目に移動すると有効になる数字編集を使って、すべての数字項目を事前表示することができます。他の事前表示は発生しません。

   3　項目がデータを受け付ける直前にすべての項目の事前表示が可能になります。

   4　データ入力が許可される前にすべての項目を事前表示します。これは省略値です。

6. SECURE項目へのACCEPT。

   利用者がカーソルをどのように動かしたいかと、SECURE項目へのACCEPT中の項目の外観を指定できるようにします。可能なオプションは、以下のようなものです。

   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。これは省略値です。

   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スペースが表示され、カーソルが次の文字の位置に進みます。

7. 項目間の自動スキップ。

   現在の項目が埋まったら、カーソルが自動的に次の項目に移動するようにしたいかどうかを指定します。これは、1つのACCEPT文内にある複数のデータ項目にしか適用されません。使用できるオプションは、以下のようなものです。

   1　自動スキップなし。次の項目に移動するには、タブキーとして設定されたキーを押すか、あるいはカーソルキー(･以外のもの)を押さなければなりません。

   2　自動スキップが有効。現在の項目が埋まったときに、カーソルを動かしたり、文字キーを押すと、次の項目にカーソルが移動します。これは省略値です。

   このオプションは、画面セクション項目上のACCEPT処理に何の影響も与えません。これらの処理では、自動スキップは省略値でオフになっています。利用者は、自分のソースプログラムのAUTO句を指定すること

によって、それをオンにすることができます。詳しくは、AUTO句 を参照してください。

8. ACCEPTの終了。

   利用者がACCEPT処理を終了させる動作を指定することができます。ACCEPT処理を終了させるには、以下を指定することができます。

   1　"ACCEPT終了"キーを押します。これは省略値です。

   2　カーソルがACCEPT処理の最後の項目に来たときに、"次項目" キーを押します。

   3　項目間の自動スキップが有効になっている場合は、ACCEPT処理の最後の使用可能な文字位置でデータ文字を打ち込むか、再入力します。

   このオプションは、通常の終了しか管理しません。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される場合の妥当性検査管理。

   ACCEPTがファンクションキーを使って終了されるとき、妥当性検査句が満たされる必要があるかどうかを指定することができます。以下から選択することができます。

   1　妥当性検査は行われません。これは省略値です。

   2　現在の項目について、通常の妥当性検査基準が満たされなければなりません。

10. 項目終端の影響。

    項目が埋まったときにデータを入力しようとした場合に、カーソルにどのような動作をさせたいかと指定することができます。以下のオプションから一つを選択することができます。

    1　カーソルが項目の終端を超えて移動し、オーバータイプが拒否されます。

    2　カーソルは項目の終端に留まったままで、オーバータイプが拒否されます。

    3　カーソルは項目の終端に留まったままで、オーバータイプは許可されます。これは省略値です。

11. 項目オーバーフローバッファの有効化／無効化。

    項目の終端からはずれたときに、オーバーフローバッファにデータがセーブされるかどうかを指定することができます。以下のオプションから選択することができます。

    1　場所からはずれたデータは、オーバーフローバッファにセーブされます。これは省略値です。

    2　場所からはずれたデータは、オーバーフローバッファにセーブされません。

12. 置換編集モードでのバックスペース中の自動リストア。

置換編集モードにあるときに、自由書式項目でのBackspaceキーの動作を指定することができます。以下のオプションから選択することができます。

1　　自動リストアが有効です。文字が削除されると、前にオーバータイプした文字がリストアされます。これは省略値です。

2　　自動リストアは無効です。削除された文字は、スペースで置き換えられます。

13. 数字編集された項目へのACCEPT。

ACCEPT処理中の数字編集された項目の外観をどうするかを指定することができます。以下のオプションから選択することができます。

a　　入力は、英数字項目については受け入れられ、その項目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、記号、小数点、カンマしか入力できません。

c　　長さ32 文字までの数字項目は、フォーマットモードで受け付けられます。つまり、数字、記号、小数点以外の文字は拒否されるということです。項目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るのがわかります。32文字より長い項目は、オプションaの場合と同じように受け付けられます。これは省略値です。

d　　オプション cの場合と同様ですが、32文字より長い項目がオプション bの場合のように受け付けられる点だけが例外です。

14. 非編集数字項目へのACCEPT。

ACCEPT処理中の非編集数字項目の外観を指定することができます。以下のオプションから選択することができます。

a　　書式9(*m*)V(*n*)のPICTURE句のある符号なしおよび埋め込みの符号付き非編集数字項目は、PIC 9(*n*)項目が後ろに続くPIC 9(*m*)項目のように扱われます。分割記号のある項目は、PIC S9(*m*+*n*)のように扱われます。これは省略値です。

b　　PIC 句のVのある項目がPIC S9(*m*+*n*)として扱われること以外は、オプションaと同様です。

c　　非編集数字項目はすべて、英数字項目として扱われます。

非ゼロ SIZE句を指定すると、非編集数字項目は、このオプションの設定にかかわらず、すべて自由書式項目として扱われます。

15. 自動クリアまたは事前クリアの有効化／無効化

カーソルが最初に項目に入ったときに、項目がどのような様子にしたいかを指定することができます。以下

のオプションのうちどれでも指定することができます。

a　事前クリアあるいは自動クリアは起こりません。これは省略値です。

b　事前クリアモード。項目がクリアされ、スペースまたはゼロになります。Undoキーを押すと、その項目の元の内容がリストアされます。

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、その項目がクリアされ、スペースまたはゼロになります。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、その項目の元の内容がリストアされます。

d　Undoキーを押しても、その項目の元の内容がリストアされない点以外は、オプションbと同様です。

16. 項目が更新されない場合の、項目の強制的更新。

利用者が項目を変更せずに項目から出た場合に、項目の内容がどのように表示されるかを指定できるようにします。以下のオプションから選択することができます。

1　項目が変更されない場合は、データ項目は更新されません。これは省略値です。

2　項目が変更されなくても、データ項目はつねに更新されます。このオプションは、項目が数字あるいは数字編集であり、元のデータ項目が数字データを含まない場合、あるいは項目が右に桁寄せされており、項目の元の内容はそうではない場合にのみ、効果があります。

17. 項目の末端の位置の記憶。

拡張ADIS構文が項目内の最後の文字の位置を記憶するかどうかを指定することができます。以下のオプションから選択することができます。

1　拡張ADIS構文は、最後の文字の位置を記録しません。これは省略値です。

2　拡張ADIS構文が、ACCEPT処理の間に項目に入力された最後の文字の位置を記録します。このオプションは、RM形式の数字項目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、適用されます。

18. RM 形式の数字データ入力。

利用者のシステムが数字データ項目のRM/COBOL形式入力をエミュレートするようにしたいかどうかを指定することができます。以下のオプションから選択することができます。

1　数字データ項目のCOBOLシステム入力が有効です。これは省略値です。

2　RM/COBOL形式の数字および数字編集データ入力が有効です。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. 最大項目サイズを1行に制限。

    ACCEPT処理の項目のサイズが1行に制限されるかどうかを指定することができます。以下のオプションから選択することができます。

    1　　項目は1行に制限されません。これは省略値です。

    2　　項目が1行に制限されます。

20. ACCEPT後のカーソル位置の管理。

    ACCEPT処理の終わりに、カーソルをどこに置くかを管理することができます。以下のオプションから選択することができます。

    1　　カーソルは、現在の項目の終端まで来ると、次の文字の位置に移動します。これは省略値です。

    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管理。

    利用者がUPDATE節をしたときにCONVERT句が暗示されるかどうかを指定することができます。以下のオプションから選択することができます。

    1　　CONVERT句は暗示されません。これは省略値です。

    2　　CONVERT句は暗示されません。

    このオプションは、利用者がRM/COBOL V2.0および V2.1の動作をエミュレートできるように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ーファンクションキーにマップするために、利用者のシステムがどのファンクションキーリストを使うべきかを指定することができます。以下のオプションから選択することができます。

    1　　標準的なユーザーファンクションキーのリストが使用されます。これは省略値です。

    2　　互換性ファンクションキーのリストが使用されます。COBOLシステムと一緒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持つようにこれを変更することができます。

23. COLUMN + n 句の動作の選択。

    画面セクションのこの句の後続文字の数を決めることができます。以下のオプションから選択することができます。

1　COLUMN + 1 句を使うと、先行する項目の直後にその項目が表示されます。2つの項目の間に間隔は開きません。これは省略値です。

2　COLUMN + 1 句を使うと、2つの項目の間に1文字分の間隔が開きます。

3　COLUMN +*n* 個の句で、 *n* >> 1である場合は、しかるべく扱われます。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省略時動作の選択。

カラーが指定されていない場合に、画面セクション項目の省略時カラーを指定することができます。

1　省略時の画面色が使用されます。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。これは省略値です。

2　省略時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、黒い背景に白い前景で表示されるか、あるいは最後のBLANK SCREEN 句で設定されたカラーを使って表示されます。

25. カーソルの左／右キーで項目を出ることができるかどうかの管理。

項目の先頭／終端にいるときに、キーカーソルの左／右キーを押すと、それぞれ前／次の項目に移動するかどうかを管理することができます。以下のオプションから選択することができます。

1　キーを押すと、前または次の項目に移動します。これは省略値です。

2　キーを押しても、項目を出られません。

26. 自由書式編集数字の左桁寄せ。

自由書式編集数字を入力したときに、数字が左桁寄せされるかどうかを管理することができます。以下のオプションから選択することができます。

1　項目は、左桁寄せされません。これは省略値です。

2　RM数字処理がオフになっている場合には、項目は左桁寄せされます。これは、ADIS オプション18で管理されます。

27. FULL/REQUIRED項目での妥当性検査管理。

ACCEPT処理を終了する前に、FULLまたはREQUIRED句のある項目はすべて適切な条件を満たさなければならないかどうかを管理できます。以下のオプションから選択することができます。

1　ユーザーが変更しなかった項目は、妥当性検査をされません。これは省略値です。

2 FULLまたはREQUIRED句のある項目はすべて、その内容が有効でないと、ACCEPT処理を終了できません。

28. ファンクションキーとして定義された拡張ADIS構文キーの管理。

ファンクションキーとして定義されたとき、拡張ADIS構文キーがどのような動作をするかを管理することができます。以下のオプションから選択することができます。

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の初期の製品と互換性を持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ADIS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。これは省略値です。

29. ACCEPTの画面読み込みオプションの管理。

現在の画面内容からACCEPT項目の最初の内容が読めるようにする機能を管理することができます。以下のオプションから選択することができます。

1 画面READは起こりません。項目の最初の内容は、変更されません。これは省略値です。

2 項目の事前表示が無効になっている場合には、項目の最初の内容が画面から読み込まれます。これは、ADISオプション5で管理されます。

30. 隠れた項目がスキップされるかどうかの管理。

隠れた項目をスキップできる状況を管理することができます。以下のオプションから選択することができます。

1 隠れた項目はスキップされません。これは省略値です。

2 隠れた項目はスキップされます。

31. 漢字修飾文字の特殊な動作の管理。

ACCEPT 処理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管理することができます。以下のオプションから選択することができます。

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。これは省略値です。

2 利用者が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ADIS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バイト文字セットのマシンだけです。

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秒の単位で指定することができます。

1　TIMEOUT句で指定されている時間の単位は秒です。これは省略値です。

2　TIMEOUT句で指定された時間の単位は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管理。

文字が入力されるたびにTIMEOUTタイマーがリセットされるか、あるい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することができます。以下のオプションから選択することができます。

1　タイマーはリセットされません。タイムアウトは、ACCEPT処理開始後、指定された時間が経過したときに発生します。これは省略値です。

2　タイマー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うと、ACCEPT 処理中に拡張ADIS構文によって表示されるメッセージを変更することができます。

以下のリストには、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句で定義された項目が、完全に埋められていません。

You must not leave the field empty.

- REQUIRED句で定義された項目が空欄のままになっています。

You are at the end of the field.

- カーソルが項目の終端にあります。

The cursor is past the end of the field.

- カーソルが項目の終端を越えました。

You have lost data from the end of the field.

- 項目の終端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.

- もう移動する項目がありません。

You cannot change character case here.

- 大文字小文字の変更ができない場所で、大文字と小文字の変更を行おうとしました。

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.

- リストアするものがないときに文字をリストアしようとしました。

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.

- 記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を使おうとしました。

No message defined.

- 2バイト文字を入れるスペースがありません。

You cannot use more than one decimal point.

- 小数点を2つ以上使おうとしました。

You must enter some numeric digits here.

- 数字項目に数字以外のデータを入力しようとしました。

付録A: チュートリアル - ACCEPT／DISPLAY

このチュートリアルでは、NetExpressを用いた、キャラクタベースのプログラミングをご紹介します。Adsamp.cblプログラムでのユーザー向けの 文字画面のセットアップ方法と、入力方法についてご説明します。

ここでは、拡張ACCEPT／DISPLAY構文(ADIS)を用いてCOBOLテキストウィンドウにデータ入力画面を表示する方法を学びます。 これで、ユーザーが入力したデータを受け取る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 ユーザーごとに特定されるということを意味しています。

以下のセクションを順番に実行し、ステップごとのチュートリアルに従ってください。

1. ADSAMPプログラムを準備する
2. ACCEPTを終了するキーを指定する
3. データ入力画面を表示する
4. データを受け取る
5. ユーザーの入力を照会する

A.1 ステップ 1 - ADSAMPプログラムを準備する

1. [スタートープログラムーMicro Focus NetExpress]を選択し、NetExpressアイコンをクリックし、NetExpressを呼出します。NetExpressで[ファイルー開く]をクリックし、 Adsampプロジェクトをロードし、¥demoディレクトリの中身を見ます。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。これが終わると、出力領域に, "Rebuild complete" が表示されます。
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強調表示され、いつでも開始できる状態になります。

A.2 ステップ 2 - ACCEPTを終了するキーを指定する

ACCEPT／DISPLAYモジュール("XAF")で、すべてのデータが入力された時、利用者が どのキーを押してACCEPTを終了するのかを指定できます。このモジュールは、その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、最初のステートメントを実行します。

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、EscapeとF1の2つのファンクションキーだけを有効にする設定を行う集団項目です。 2つめの基本パラメータは値"1"に設定され、利用者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番目のステートメントを実行します。

ここで同じ機能を異なるの2つめのパラメータと一緒に使用しました。Disable-all-other-user-keysは、Escape 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つめの基本パラメータは、ここでも値"1"を持っています。

3.  をクリックして3つ目のステートメントを実行します。

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つめ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トピックで、これがどのように動作するか確認できます。

A.3 ステップ 3 - データ入力画面を表示する

1. 最初に、  を2回クリックしてカーソルいちをセットアップします。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、実行に移動し、再びdisplayステートメントを実行します。

データ入力画面は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 ステップ 4 - データを受け取る

1.  をクリックしてacceptステートメントを実行します。
2. データ入力画面でフィールドにデータを入力します。
3. F1キーを押します。

このファンクションキーを特別に有効にしたので、プログラムに戻されます。Escapeキー、または"/"キーを押すと、 同じ動作をしますが、その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。Escapeキー、または"/"キーを押 すと、同じ動作をしますが、その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. [キー種類]を右クリックし、表示されるポップアップメニューで["キー種類"の検索]をクリックします。

   キー状態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箇所を右クリックし、表示されるポップアップメニューで[Return もどる]をクリックします。

A.5 ステップ 5 - ユーザーの入力を照会する

1.  をクリックして、evaluateステートメントを実行します。

   条件"1"は真です。これは利用者ファンクションキー用のコードだからです(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.  をもう一度クリックして、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.  を再びクリックして、繰り返します。

   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出すよう命令することになります。

4.  をクリックしてdisplay-help-screenセクションに移動します。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出しは以前使用したset-bit-pairs呼出しとは異なる機能です。今回は、以前のようにデータを受け取るのではなくて、 利用者が単一キャラクタを入力します。

5.  をクリックして呼出しを実行します。

   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プログラムの操作に慣れている場合は、set-bit-pairs呼出しを用いて、別のキーを有効にしてみてから、evaluate key-type文で区切り点を設定します。ループで、1回ごとに異なるキーを押し、これがキー状態の値に与える影響を確認します。

付録B: 事例とプログラム例

本章では、前の章で説明した内容を具体的に示す事例とコードの例を記載しています。コードだけを見るのではなく、前の章を参照して理解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 must not be zero"
              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ステートメント

```
call x"AF" using set-bit-pairs
                 parameter-block
```

ここで、パラメータは以下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2つのパラメータに必要な値については、それぞれ説明します。

すべてのx"AF"呼出しで、エラーが発生すると、*set-bit-pairs*が呼出しからの戻り値として、値255に設定されます。

B.4 CRT ステータス句の構文

```
special-names
```

```
crt status is key-status
```

ここで、*key-status*は、以下の定義を持つプログラムのWorking-Storage Sectionにある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ステートメントが実行されると、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*の個々のフィールドは次のように使用します。

key-type　ACCEPTが終了した方法を示します。戻される値は次のようになります。

"0"　ACCEPTの正常な終了

"1"　ユーザーの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数を示します。この数の正確に意味は、key-typeで戻された値によって変わります。

key-code-2　key-typeとkey-code-1が0のとき、key-code-2には、ACCEPT操作を終了させたキーのローキーボードコードが含まれています。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ところでは、最初のキーストロークのコードだけが返されます。
key-typeが4のとき、key-code-2には、ACCPET操作を終了させた文字の2つ目のバイトが含まれています。
そうでない場合は、key-code-2の中身は未定義です。

B.5 ライブラリルーチンを使ってキャラクタユーザーインターフェースを作成する

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

以下のようなグループ項目を考えます。

```
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.
```

次のステートメントが実行されると

```
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項目の中身も表示されます。

B.7 Screen Sectionを使用してキャラクタユーザーインターフェースを作成する

ボタンをクリックしてNetExpressを起動し、デモプロジェクトをロードする。

```
     $set ans85

*****************************************************************
* Copyright Micro Focus Limited 1994.   All Rights Reserved.   *
* This demonstration program is provided for use by users of   *
* Micro Focus products and may be used, modified and           *
* distributed as part of your application provided that you    *
* properly acknowledge the copyright of Micro Focus in this    *
* material.                                                    *
*****************************************************************

*****************************************************************
*                                                               *
* ADSAMP.CBL                                                    *
*                                                               *
* このプログラムでは、Adiscfを使用して標準の構成が                *
* 選択されていることを前提にしています。                         *
*****************************************************************
```

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

data division.
working-storage section.

************************************************
* x"AF"呼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の終了後に戻されるステータス
************************************************
 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が終了したときの *
* カーソル位置を含むADISによって戻されます。 *
************************************************
 01 cursor-position.
    03 cursor-row             pic 99.
    03 cursor-column          pic 99.

************************************************
* プログラムが使用するWork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.
```

```
************************************************
* Screen Section.
************************************************
 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 ...".

************************************************
* Procedure Division
************************************************

 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キーとf1キーを有効にします。

    call x"AF" using set-bit-pairs
                  enable-esc-and-f1

* その他のユーザー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

* "/"キーをセットアップしてファンクションキーと動作させ
* ACCEPT操作を終了します。

    call x"AF" using set-bit-pairs
                  enable-slash-key

* これで、ACCEPTが終了するとカーソル位置が戻されます。
* 1行1列に設定すると、カーソルの最初の位置が1番目の
* 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

* Escape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。つまり、Enterキーが
* 押されました。この場合、単純にWork領域を空白にして
* 最初のフィールドから開始します。

           initialize work-areas
           move 1 to cursor-row
           move 1 to cursor-column

         when "1"

* ユーザーファンクションキーが押されました。他のキーが無効のときは、
* Escapeキーまたは F1キーのどちらかです。

           if key-code-1 = 0

* Escapeが押されたので、プログラムを終了します。

               move 1 to exit-flag
           else

* F1が押されたのでヘルプ画面を表示します。
               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
* が有効に鳴っていない場合は、"/"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 プログラムのマッピングを変更する

次のコードは、 Backspace キー（キー番号14）の処理を変更して）、単純にカーソルを左方向に移動し（ファンクション3）、Tabキー（キー番号8）を変更してTab機能を実行させます（ファンクション8）

*** Backspaceキーのマッピングを変更**

```
    move 14 to adis-key-number
    move 3 to adis-mapping-byte
    call x"AF" using set-map-byte
                   adis-key-mapping
```

*** タブキーのマッピングを変更**

```
    move 8 to adis-key-number
```

```
    move 8 to adis-mapping-byte
    call x"AF" using set-map-byte
                   adis-key-mapping
```

B.9 ACCPETを終了するようにADISを構成する

以下は、拡張ACCEPT/DISPLAY構文を構成してACCEPT操作を終了させる例です。

```
accept data-item at 0101
if key-type="0"
                if key-code-1=48
           display "Terminated by return key"
     else
           display "Terminated by auto-skip last field"
     end-if
end-if.
```

B.10 ADISキーを削除する

次のコードではTabと Backtabを、ファンクションキーとして動作するようにセットアップします。･キーと･キーをセットアップして、カーソルがフィールドから出てしまう場合にファンクションキーとして動作するようにセットアップします。

*** Tab（key 8）とBacktab（Key 9）を次のファンクションキーとして動作するようにセットアップします。**

```
    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てしまう場合にだけ、cursor-left (key 3)と**

*** cursor-right (key 4) をファンクションキーとして動作するようにセットアップ**

*** 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
-         "leave the field"
        when 4
          display "cursor right caused the cursor to
-         "leave the field"
        when 8
         display "the tab key was pressed"
        when 9
         display "the back tab key was pressed"
      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セットアップを検出する

```
* ACCEPT操作を終了させるように、"A"から"Z"
* までのキャラクタをセットアップします。
    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 ユーザーファンクションキーを検出する

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、有効になっているただ1つのキーが Escape、F1、および F10キーであるとみなします。

```
        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を使用する - プログラム例

ファンクションキーを利用するプログラムの記述方法示した1例です。It assumes that Escapeキーが利用でき、その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が前提になっています。

```
$set ans85
************************************************
* このプログラムでは、Adiscfを使用して標準の構成が
* 選択されていることを前提にしています。
************************************************

 special-names.
     cursor is cursor-position
     crt status is key-status.
 data division.
 working-storage section.
************************************************
* x"AF"呼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の終了後に戻されるステータス
************************************************
 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.

************************************************
* ACCEPTが終了したときにカーソル位置を含むADIS
* によって戻されるCursor-Position
************************************************
 01 cursor-position.
   03 cursor-row                        pic 99.
   03 cursor-column             pic 99.

************************************************
* プログラムが使用するWork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.
************************************************
* Screen Section.
************************************************
 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 ...".

************************************************
* Procedure Division.
************************************************

 procedure division.
 entry-point section.
```

*** まず、必要なキーを有効にします。**

*** EscapeとF1キーを有効にします。**.

```
    call x"AF" using set-bit-pairs
                  enable-esc-and-f1
```

* その他の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"/"をセットアップします。
* ACCEPT操作を終了します。

```
    call x"AF" using set-bit-pairs
                  enable-slash-key
```

* ACCEPTが終了したときにカーソル位置が戻されるようにします。
* 行1列1に設定すると、カーソルが初めに1番目のフィールドの開
* 始位置にきます。

```
    move 1 to cursor-row
    move 1 to cursor-column
```

* Escape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。つまり、

```
* Enterキーが押されました。 この場合、単純にWork領域を空白にして
* 最初のフィールドから開始します。

           initialize work-areas
           move 1 to cursor-row
           move 1 to cursor-column

         when "1"

* ユーザーファンクションキーが押されました。
* 他のキーがすべて無効になっているので、EscapeキーかF1キーです。

           if key-code-1 = 0

* Escapeが押されたので、プログラムを閉じます。

               move 1 to exit-flag
           else

* F1が押されたので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。他のキーでは
* この処理が無効になっているので、"/"です。
* ここで、次の文字を取得して、"H"または"h"
* のどちらが押されのか確認します。

           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.
```

索引