Helproutines

Helproutines have specific characteristics to facilitate the processing of help requests. They may be used to implement complex and interactive help systems. They are created with the program editor.

This chapter covers the following topics:

- Invoking Help

- Specifying Helproutines

- Programming Considerations for Helproutines

- Passing Parameters to Helproutines

- Equal Sign Option

- Array Indices

- Help as a Window

Invoking Help

A Natural user can invoke a Natural helproutine either by entering the help character in a field, or by pressing the help key (usually PF1). The default help character is a question mark (?).

- The help character must be entered only once.

- The help character must be the only character modified in the input string.

- The help character must be the first character in the input string.

If a helproutine is specified for a numeric field, Natural will allow a question mark to be entered for the purpose of invoking the helproutine for that field. Natural will still check that valid numeric data are provided as field input.

If not already specified, the help key may be specified with the SET KEY statement:

```
SET KEY PF1=HELP
```

A helproutine can only be invoked by a user if it has been specified in the program or map from which it is to be invoked.

Specifying Helproutines

A helproutine may be specified:

- in a program: at statement level and at field level;

- in a map: at map level and at field level.

If a user requests help for a field for which no help has been specified, or if a user requests help without a field being referenced, the helproutine specified at the statement or map level is invoked.

A helproutine may also be invoked by using a `REINPUT USING HELP` statement (either in the program itself or in a processing rule). If the `REINPUT USING HELP` statement contains a `MARK` option, the helproutine assigned to the `MARK`ed field is invoked. If no field-specific helproutine is assigned, the map helproutine is invoked.

A `REINPUT` statement in a helproutine may only apply to `INPUT` statements within the same helproutine.

The name of a helproutine may be specified either with the session parameter `HE` of an `INPUT` statement:

```
INPUT (HE='HELP2112')
```

or using the extending field editing facility of the map editor (see *Creating Maps* and the *Editors* documentation).

The name of a helproutine may be specified as an alphanumeric constant or as an alphanumeric variable containing the name. If it is a constant, the name of the helproutine must be specified within apostrophes.

Programming Considerations for Helproutines

Processing of a helproutine can be stopped with an `ESCAPE ROUTINE` statement.

Be careful when using `END OF TRANSACTION` or `BACKOUT TRANSACTION` statements in a helproutine, because this will affect the transaction logic of the main program.

Passing Parameters to Helproutines

A helproutine can access the currently active global data area (but it cannot have its own global data area). In addition, it can have its own local data area.

Data may also be passed from/to a helproutine via parameters. A helproutine may have up to 20 explicit parameters and one implicit parameter. The explicit parameters are specified with the `HE` operand after the helproutine name:

```
HE='MYHELP','001'
```

The implicit parameter is the field for which the helproutine was invoked:

```
INPUT #A (A5) (HE='YOURHELP','001')
```

where `001` is an explicit parameter and `#A` is the implicit parameter/the field.

This is specified within the `DEFINE DATA PARAMETER` statement of the helproutine as:

```
DEFINE DATA PARAMETER
1 #PARM1 (A3)          /* explicit parameter
1 #PARM2 (A5)          /* implicit parameter
END-DEFINE
```

Please note that the implicit parameter (#PARM2 in the above example) may be omitted. The implicit parameter is used to access the field for which help was requested, and to return data from the helproutine to the field. For example, you might implement a calculator program as a helproutine and have the result of the calculations returned to the field.

When help is called, the helproutine is called before the data are passed from the screen to the program data areas. This means that helproutines cannot access data entered within the same screen transaction.

Once help processing is complete, the screen data will be refreshed: any fields which have been modified by the helproutine will be updated - excluding fields which had been modified by the user before the helproutine was invoked, but including the field for which help was requested. *Exception:* If the field for which help was requested is split into several parts by dynamic attributes (DY session parameter), and the part in which the question mark is entered is *after* a part modified by the user, the field content will not be modified by the helproutine.

Attribute control variables are not evaluated again after the processing of the helproutine, even if they have been modified within the helproutine.

Equal Sign Option

The equal sign (=) may be specified as an explicit parameter:

```
INPUT PERSONNEL-NUMBER (HE='HELPROUT',=)
```

This parameter is processed as an internal field (format/length A65) which contains the field name (or map name if specified at map level). The corresponding helproutine starts with:

```
DEFINE DATA PARAMETER
1 FNAME (A65)              /* contains 'PERSONNEL-NUMBER'
1 FVALUE (N8)              /* value of field (optional)
END-DEFINE
```

This option may be used to access one common helproutine which reads the field name and provides field-specific help by accessing the application online documentation or the Predict data dictionary.

Array Indices

If the field selected by the help character or the help key is an array element, its indices are supplied as implicit parameters (1 - 3 depending on rank, regardless of the explicit parameters).

The format/length of these parameters is I2.

```
INPUT A(*,*)  (HE='HELPROUT',=)
```

The corresponding helproutine starts with:

```
DEFINE DATA PARAMETER
1 FNAME   (A65)            /* contains 'A'
1 FVALUE  (N8)             /* value of selected element
1 FINDEX1 (I2)             /* 1st dimension index
1 FINDEX2 (I2)             /* 2nd dimension index
END-DEFINE
...
```

Help as a Window

The size of a help to be displayed may be smaller than the screen size. In this case, the help appears on the screen as a window, enclosed by a frame, for example:

```
*****************************************************************************
                        PERSONNEL INFORMATION
PLEASE ENTER NAME: ?_____
PLEASE ENTER CITY: _____
                  +--------------------------+
                  !                          !
                  ! Type in the name of an   !
                  ! employee in the first    !
                  ! field and press ENTER.   !
                  ! You will then receive    !
                  ! a list of all employees  !
                  ! of that name.            !
                  !                          !
                  ! For a list of employees  !
                  ! of a certain name who    !
                  ! live in a certain city,  !
                  ! type in a name in the    !
                  ! first field and a city   !
                  ! in the second field      !
                  ! and press ENTER.         !
******************!                          !*****************************
                  +--------------------------+
```

Within a helproutine, the size of the window may be specified as follows:

- by a `FORMAT` statement (for example, to specify the page size and line size: `FORMAT PS=15 LS=30`);

- by an `INPUT USING MAP` statement; in this case, the size defined for the map (in its map settings) is used;

- by a `DEFINE WINDOW` statement; this statement allows you to either explicitly define a window size or leave it to Natural to automatically determine the size of the window depending on its contents.

The position of a help window is computed automatically from the position of the field for which help was requested. Natural places the window as close as possible to the corresponding field without overlaying the field. With the `DEFINE WINDOW` statement, you may bypass the automatic positioning and determine the window position yourself.

For further information on window processing, please refer to the `DEFINE WINDOW` statement in the *Statements* documentation and the terminal command `%W` in the *Terminal Commands* documentation.