Definition:

A report program in ABAP/4 is a program which reads and analyzes data from database tables without modifying the database. Usually, the result of such a report program is in the form of a list which is output to the screen or sent to a printer. Therefore, a report program is a program which creates a report based on the evaluation of one or more database tables.

From the technical point of view, a report program is an ABAP/4 program with program attribute type 1 that runs standalone. The runtime environment of standalone programs differs from the runtime environment of module pool programs. The general flow of a report program is controlled exclusively by external events, while module pool programs are controlled mainly by the screen flow logic of the dialog program.

The general flow of a report program is controlled exclusively by external events, while module pool programs are controlled mainly by the screen flow logic of the dialog program.

What are Selection Screens?

The selection screen is the part of a report program that you can design for interactive input of field values and selection criteria.

The selection screen is always processed directly after a report program is started.

The user can enter field values and selection criteria on this screen.

Texts on the selection screen can be maintained as language dependent text elements.

In an ABAP/4 program, you use the following statements to design selection screens:

PARAMETERS to define input fields for variables
SELECT-OPTIONS to define input fields for selection criteria
SELECTION-SCREEN to format the selection screen.

PARAMETERS:
To declare a parameter and its data type, use the PARAMETERS statement as follows:

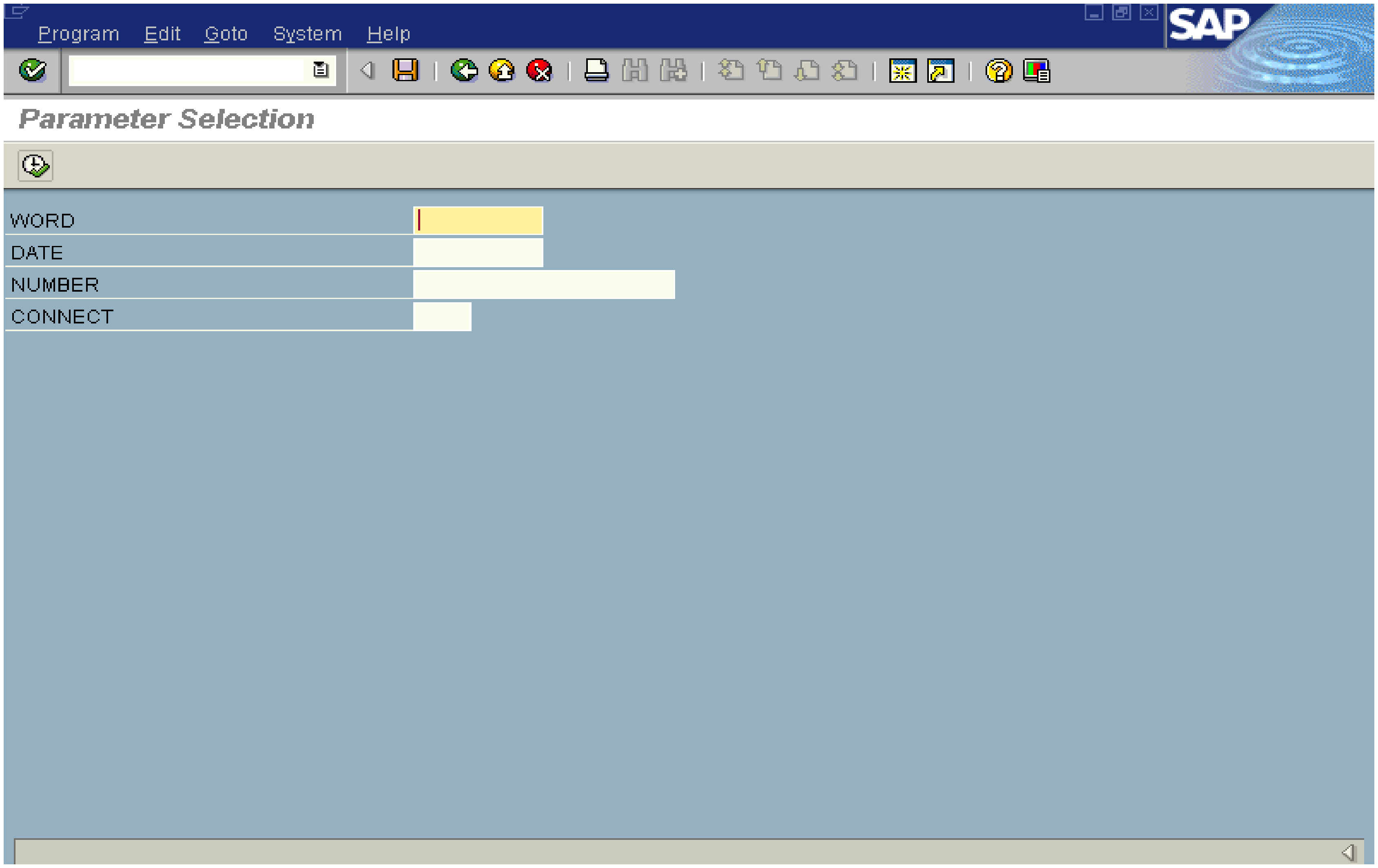
Syntax
PARAMETERS <p>[(<length>)] <type> [<decimals>].

Caution
Parameters cannot have data type F. The data type F is not supported on the selection screen.

Example:

```
REPORT SAPMZTST.
TABLES SPFLI.
PARAMETERS : WORD(10) TYPE C,
             DATE TYPE D,
             NUMBER TYPE P DECIMALS 2,
             CONNECT Like SPFLI-CONNID.
```

This example creates four fields, namely a character field, WORD, of length 10; a date field, DATE of length 8; a packed number field, NUMBER, with two decimals; and a field, CONNECT, which refers to the ABAP/4 Dictionary structure SPFLI-CONNID. When the user starts the report SAPMZTST, input fields fo the four declared fields will appear on the selection screen as follows :



Parameter Selection

WORD	
DATE	
NUMBER	
CONNECT	

Selection Screen using Parameters

You can use parameters, for example, to control the program flow, or in connection with SELECT statement to enable the report user to determine selection criteria for database accesses .

Example:

```
TABLES SPFLI.
PARAMETERS : LOW LIKE SPFLI-CARRID,
                         HIGH LIKE SPFLI-CARRID.

SELECT * FROM SPFLI WHERE CARRIED BETWEEN LOW AND HIGH.
             ----------------
ENDSELECT.
```

In this example, the system reads all rows from the database table SPFLI, where the contents of field CARRIED are between the limits LOW and HIGH. The report user can enter the fields LOW and HIGH on the selection screen.

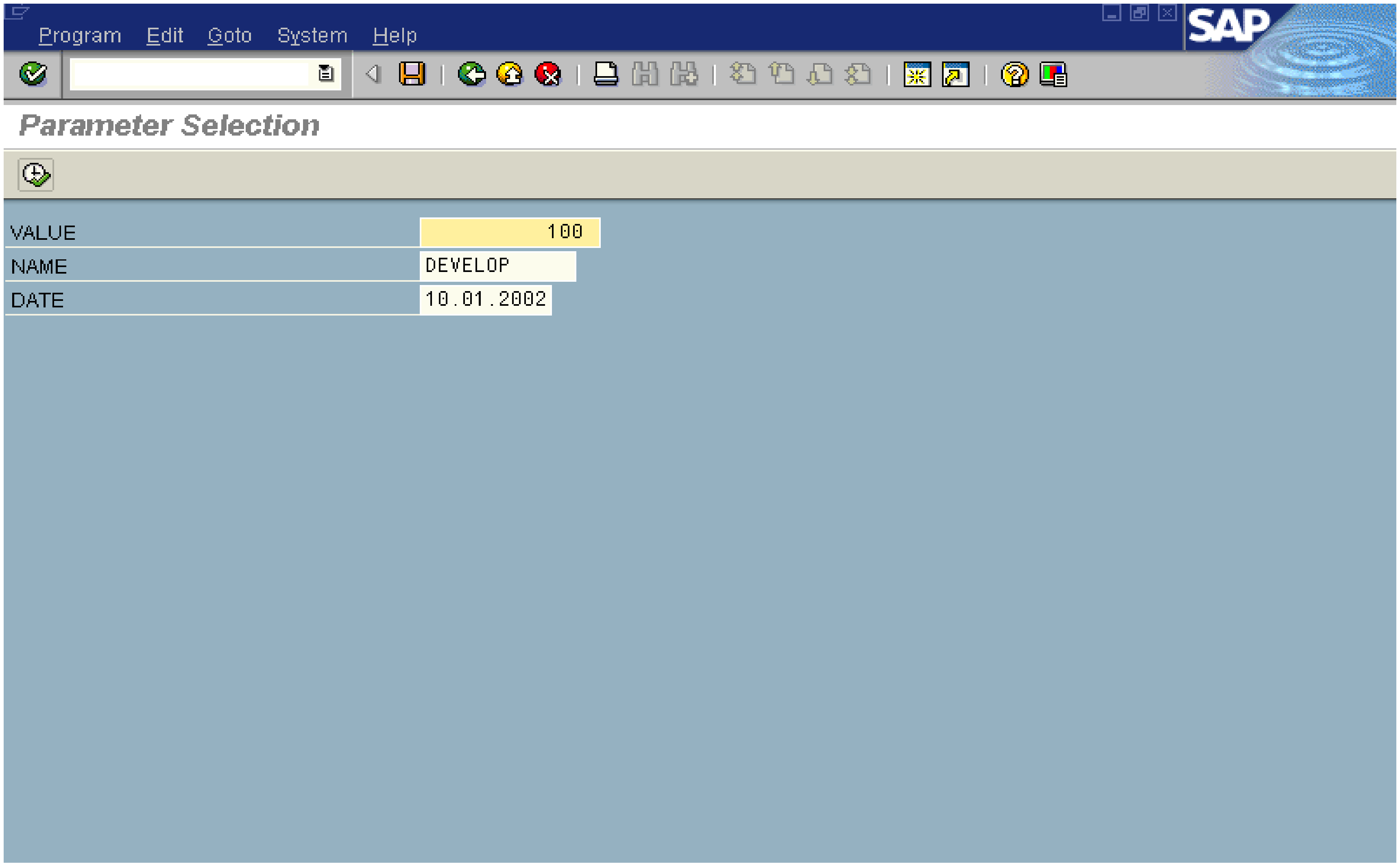
Variants of Parameters statements are:

```
PARAMETERS <p> ...... DEFAULT <f> ......
PARAMETERS <p> ...... NO-DISPLAY ......
PARAMETERS <p> ...... LOWER CASE ......
PARAMETERS <p> ...... OBLIGATORY ......
PARAMETERS <p> ...... AS CHECKBOX ......
PARAMETERS <p> ...... RADIOBUTTON GROUP <radi>......
PARAMETERS <p> ...... MEMORY ID <pid>......
PARAMETERS <p> ...... MATCHCODE OBJECT <obj> ......
PARAMETERS <p> ...... MODIF ID <key> ......
```

Assigning Default Values to Parameters :

```
REPORT ZTEST2 .
TABLES SPFLI.
PARAMETERS :     value type i default 100,
                 name like sy-uname default sy-uname,
                 date like sy-datum default '20020110'.
```

The declared fields will appear on the selection screen as follows :



Parameter Selection

VALUE	100
NAME	DEVELOP
DATE	10.01.2002

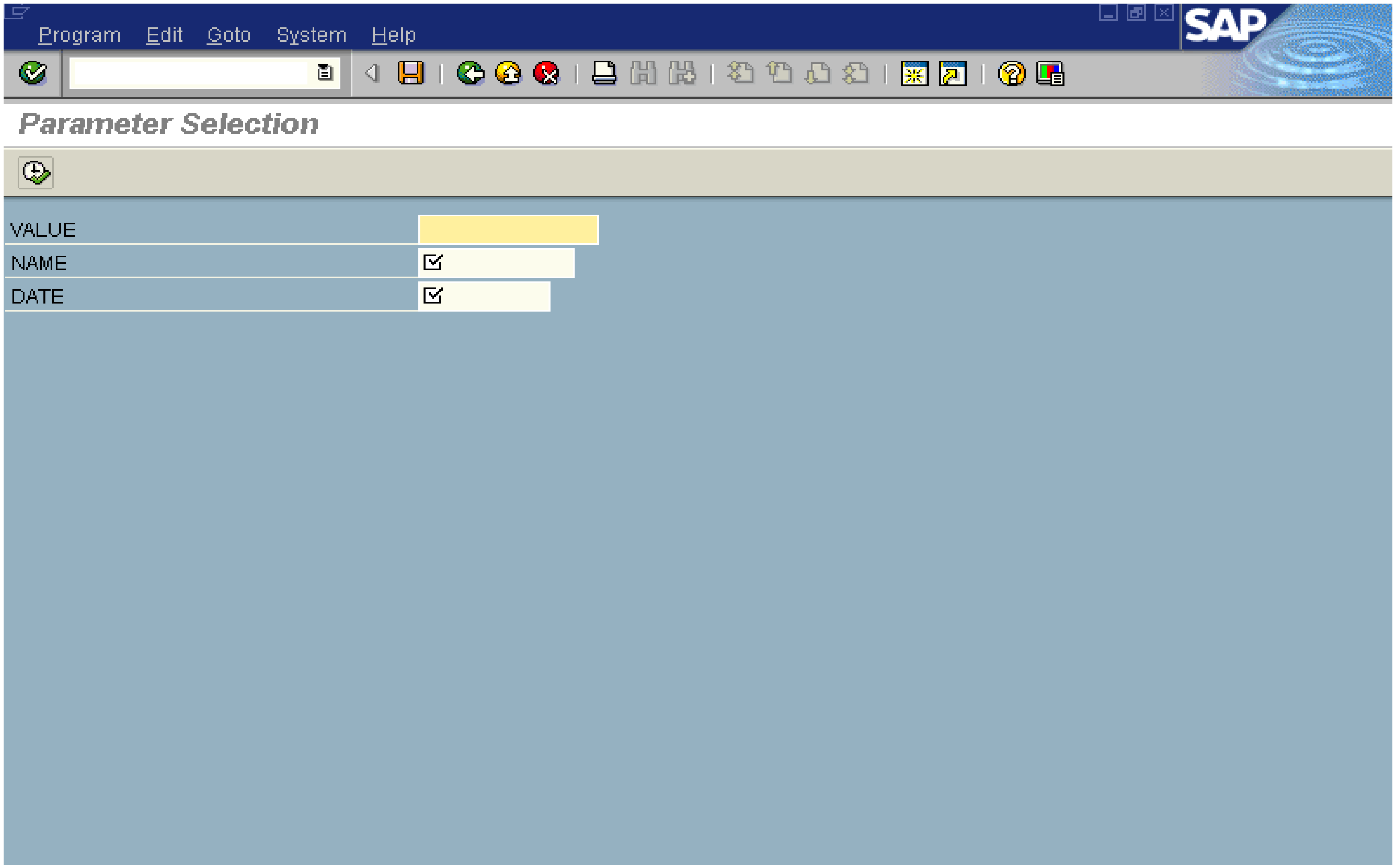
Assigning Default Values to Parameters

Parameters <p> Obligatory:

Example :

When you use this option, a question mark appears in the input field for parameter <p>. The user cannot continue with the program without entering a value I this field on the selection screen.

The declared fields will appear on the selection screen as follows :



Parameter Selection

VALUE	
NAME	✓
DATE	✓

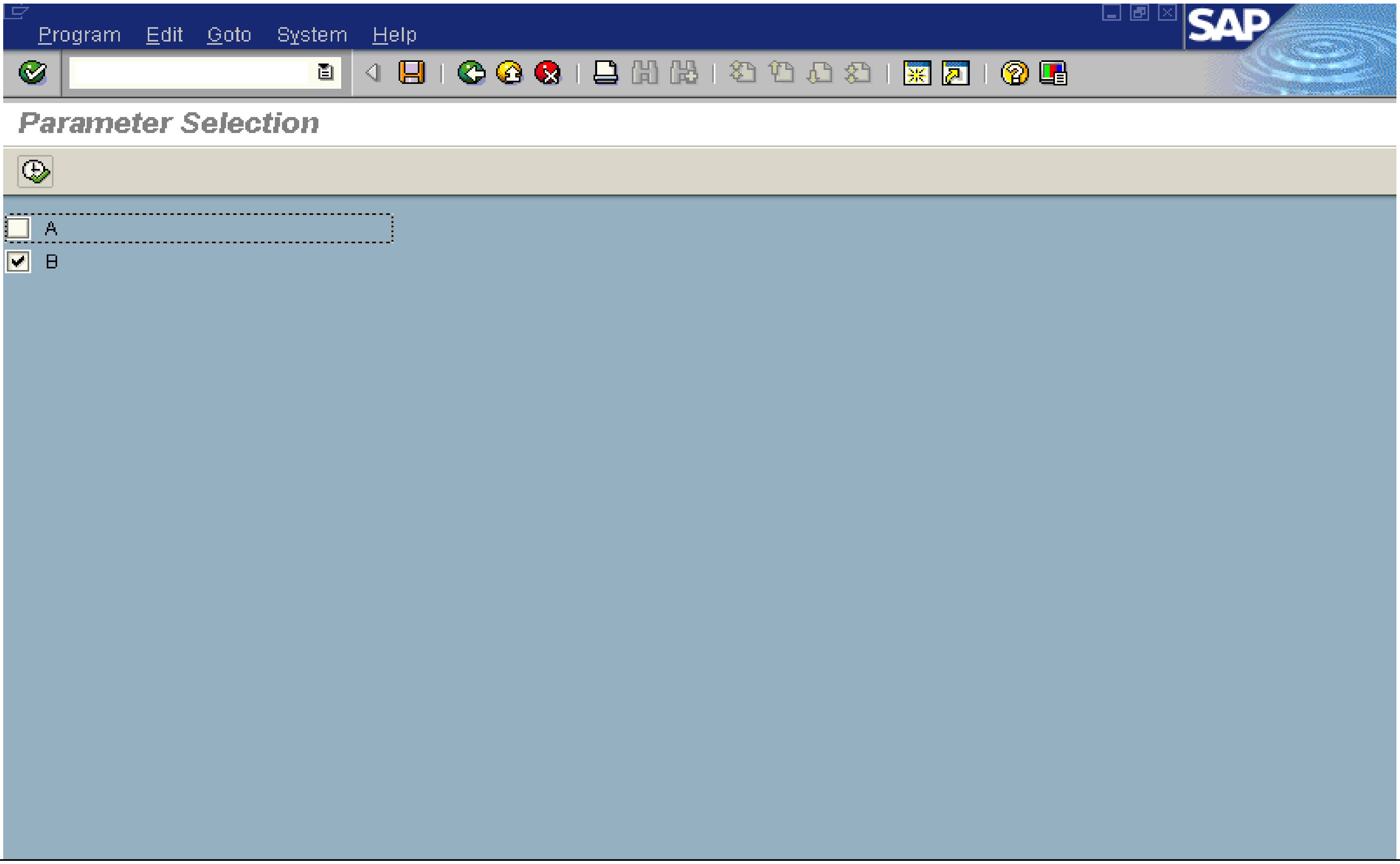
Making Parameters Required Input Field

Parameters <p> …as CHECKBOX…

Example :

```
REPORT ZTEST2 .
TABLES SPFLI.
PARAMETERS : A as Checkbox,
             B as Checkbox Default 'X'.
```

The declared fields will appear on the selection screen as follows :



Parameter Selection

- [ ] A
- [x] B

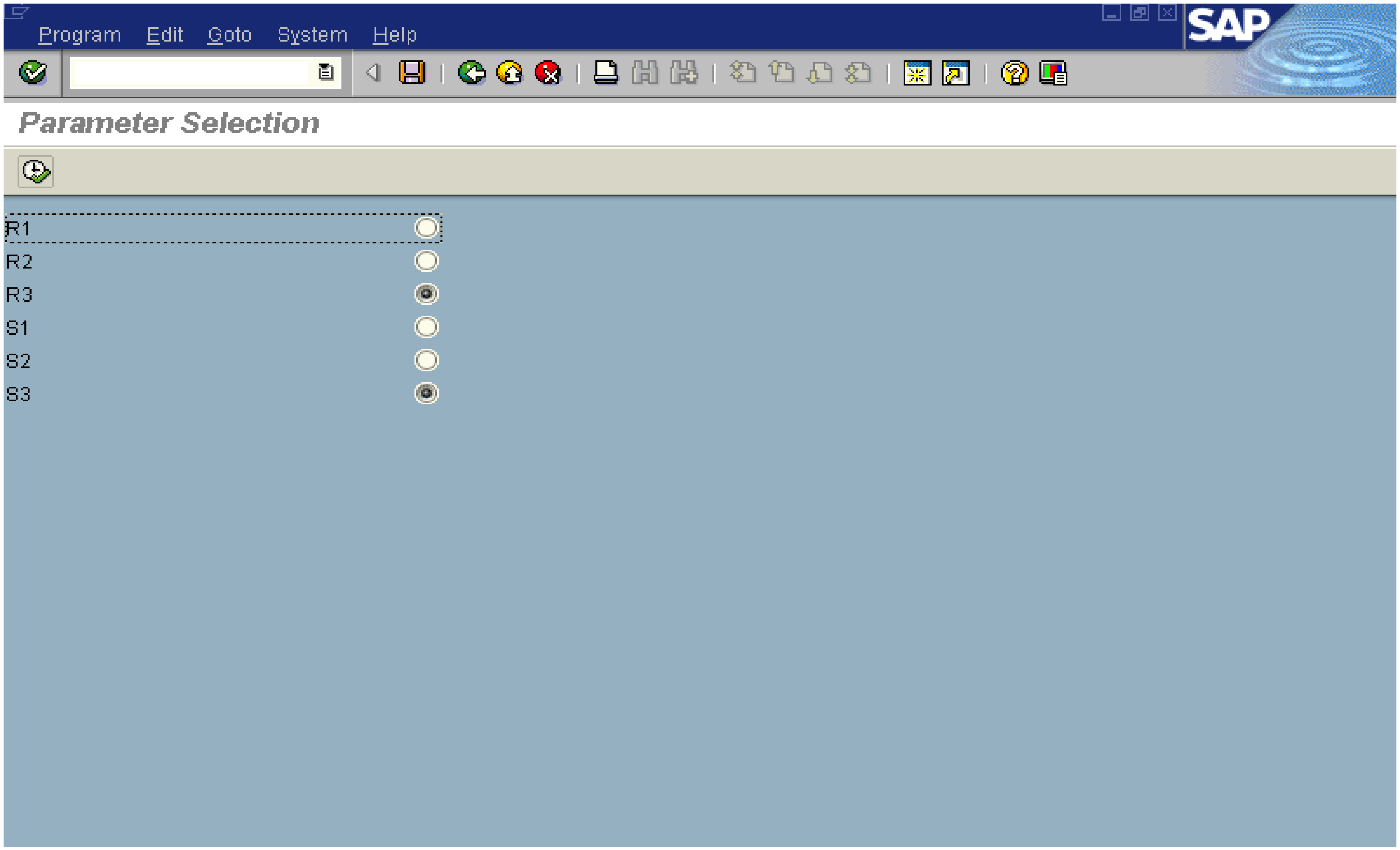
Making Checkbox on the Selection Screen

Parameters <p>….RADIOBUTTON GRUP <radi>…..

Example :

```
REPORT ZTEST2 .
TABLES SPFLI.
PARAMETERS : R1 RADIOBUTTON GROUP RAD1,
             R2 RADIOBUTTON GROUP RAD1,
             R3 RADIOBUTTON GROUP RAD1  DEFAULT ‘X’,
             S1 RADIOBUTTON GROUP RAD2,
             S2 RADIOBUTTON GROUP RAD2,
             S3 RADIOBUTTON GROUP RAD2 DEFAULT ‘X’.
```

The declared fields will appear on the selection screen as follows :



Parameter Selection

R1 ○
R2 ○
R3 ●
S1 ○
S2 ○
S3 ●

Making Radio Button Groups on the Selection Screen.

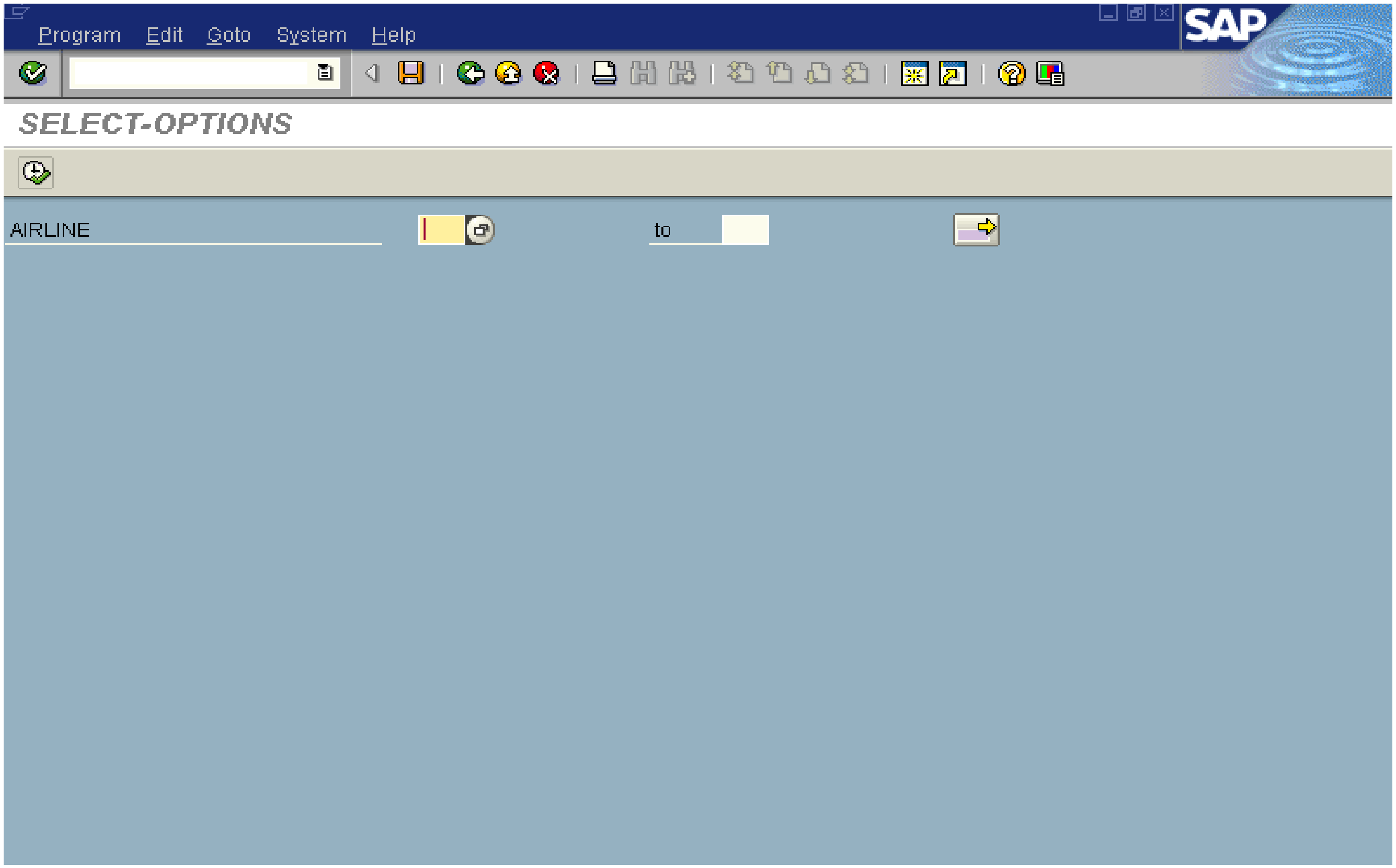
What are Selection Criteria?
You define selection criteria with the SELECT-OPTIONS statement.

•If you use the SELECT-OPTIONS statement, the report user can enter the selection criteria on the selection screen.

•During its definition, you normally attach a selection criterion to a specific column of a database table that must be declared in the program.

•If you want to program complex selections, selection criteria, stored in special internal tables, are preferable to the PARAMETERS statement because they relieve you from coding them in lengthy WHERE conditions.

SELECT-OPTIONS statement
Syntax
SELECT-OPTIONS <seltab> FOR <f>.



SELECT-OPTIONS

AIRLINE to

After Executing Program a Selection Screen appears as above:

Note :

•The field <f> cannot have data type F. The data type F is not supported on the selection screen.

•For SELECT-OPTIONS statement, the system creates a selection table. The purpose of selection tables is to store complex selection limits in a standardized manner.

•Their main purpose is to transfer the selection criteria directly to database tables using the WHERE clause of Open SQL statements.

Selection table
It is an internal table(with name <seltab>) with a header line. Its line structure is a field string of four components,

* SIGN : The data type of SIGN is C with length 1. Possible values are I and E.

* OPTION : The data type of OPTION is C with length 2. OPTION contains the selection operator.

- If HIGH is empty, you can use EQ, NE, GT, LE, LT,CP, and NP.
- If HIGH is filled, you can use BT and NB.

* LOW : The data type of LOW is the same as the column type of the database table, to which the selection criterion is attached.

* HIGH : The data type of HIGH is the same as the column type of the database table, to which the selection criterion is attached.

The SELECT-OPTIONS statement has several variants which are:

```
SELECT-OPTIONS <seltab> FOR <f> DEFAULT <g> [TO <h>] ....
SELECT-OPTIONS <seltab> FOR <f> ... NO-EXTENSION .....
SELECT-OPTIONS <seltab> FOR <f> ... NO INTERVALS .....
SELECT-OPTIONS <seltab> FOR <f> ..  NO DATABASE
                                     SELECTION…….
SELECT-OPTIONS <seltab> FOR <f> ... NO-DISPLAY .............
SELECT-OPTIONS <seltab> FOR <f> ... LOWER CASE .............
SELECT-OPTIONS <seltab> FOR <f> ... OBLIGATORY .............
SELECT-OPTIONS <seltab> FOR <f> ... MEMORY ID <pid>..........
SELECT-OPTIONS <seltab> FOR <f> ... MODIF ID <key>...........
SELECT-OPTIONS <seltab> FOR <f> ... MATCHCODE OBJECT   <obj>...
```

The Following example shows how the selection table is filled with the user inputs on the selection screen:

Example:

```
REPORT SAPMZTST.
TABLES SPFLI.
SELECT-POTIONS AIRLINE FOR SPFLI-CARRID.
LOOP AT AIRLINE.
WRITE : / ‘SIGN:’, AIRLINE-SIGN,
          ‘OPTION:’, AIRLINE-OPTION,
          ‘LOW:’, AIRLINE-LOW,
          ‘HIGH:”, AIRLINE-HIGH.
ENDLOOP.
```

Using Selection Tables in the WHERE Clause:

To limit the database access of the Open SQL statements SELECT, UPDATE, and DELETE, you use the WHERE clause.

Syntax

```
            ......... WHERE <f> IN <seltab>.
```

Example :

```
REPORT SAPMZTST.
TABLES SPFLI.
SELECT-OPTIONS AIRLINE FOR SPFLI-CARRID.
SELECT * FROM SPFLI WHERE CARRID IN AIRLINE.
 WRITE : SPFLI-CARRID.
ENDSELECT.
```

In the SELECT-OPTIONS statement of this example, the selection table AIRLINE is attached to the CARRIED column of the database table SPFLI. The WHERE clause of the SELECT statement causes the system to check if the contents of the CARRIED column meet the selection criteria stored in AIRLINE.

Assume that the report user enters two lines in the selection table, namely an interval selection and a single value selection, as follows :

SIGN	OPTION	LOW	HIGH
I	BT	DL	UA
E	EQ	LH	

Then, the report output appears as follows :
DL DL SQ UA UA UA

Using Selection Tables in Logical Expressions

To control the internal flow of your program, you must program conditions with logical expressions. You can program special logical expressions using selection tables.

Syntax

```
... <f> IN <seltab> ....
```

The logical expression is true if the contents of the field <f> meet the selection limits stored in the selection table <seltab>. <f> can be any internal field or the column of a database table.

If the selection table (seltab) is attached to <f> with the SELECT-OPTIONS statement, you can use the following short form for the logical expression.

Syntax :   …<seltab>….

Example :

```
REPORT SAPMZTST.
TABLES SPFLI.
SELECT-OPTIONS AIRLINE FOR SPFLI-CARRID.
WRITE : ‘Inside’, ‘Outside’.
SELECT * FROM SPFLI.
   IF SPFLI-CARRID IN AIRLINE.
            WRITE : / SPFLI-CARRID UNDER ‘Inside’.
     ELSE.
                 WRITE : / SPFLI-CARRID UNDER ‘Outside’.
 ENDIF.
ENDSELECT.
```

Assume that the report user enters two lines in the selection table, namely an interval selection and a single value selection, as follows :

SIGN	**OPTION**	**LOW**	**HIGH**
I	**BT**	**DL**	**UA**
E	**EQ**	**LH**	

Then, the report output appears as follows :

Inside	**Outside**
	AA
DL	
	LH
SQ	
UA	

In the SELECT loop, all lines are read from the database table SPFLI. Using the IF statement, the program flow is branched into two statement blocks according to the logical expression. The short form IF AIRLINE is also possible in this program.

Formatting the Selection Screen

•The selection screen, which appears when the PARAMETERS   or SELECT-OPTIONS statements are used in a program, has a standard layout where all parameters appear line after line.

•SELECTION-SCREEN statement can be used to format selection screen when the standard selection screen layout is not sufficient.

•The SELECTION-SCREEN statement works only on selection screens. It has no effect in reports which have no selection screen. You cannot, for example, create pushbuttons wihout executing the PARAMETERS or SELECT-OPTIONS statement.

Blank Lines

To produce blank lines on the selection screen, use the SKIP option with the SELECTION-SCREEN statement.

Syntax

SELECTION-SCREEN SKIP [<n>].

Underlines

To underline a line or part of a line on the selection screen, use the ULINE option with the SELECTION-SCREEN statement.

Syntax

```
SELECTION-SCREEN ULINE [[/]<pos(len)>] [MODIF ID <key>].
```

Comments

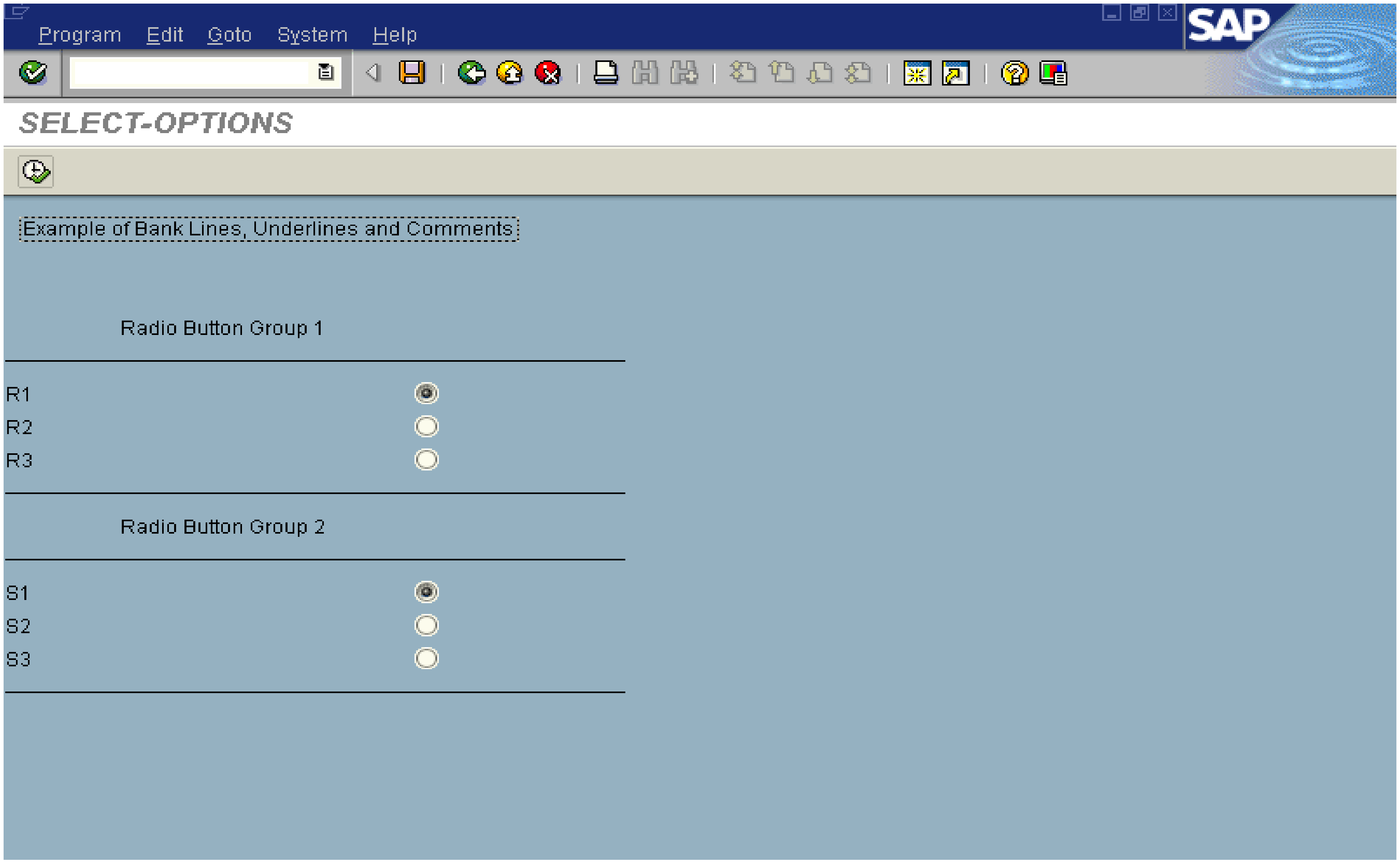
To write text on the selection screen, use the COMMENT option with the SELECTION-SCREEN statement. The syntax is as follows:

Syntax

```
          SELECTION-SCREEN COMMENT [/]<pos(len)> <name>
                                             [MODIF ID <key>].
```

Example of Blank Lines, Underlines, and Comments :

```
SELECTION-SCREEN COMMENT /2(5) TEXT-001 MODIF ID SC1.
SELECTION-SCREEN SKIP 2.
SELECTION-SCREEN COMMENT /10(30) COMM1.
SELECTION-SCREEN ULINE.
PARAMETERS : R1 RADIOBUTTON GROUP RAD1,
             R2 RADIOBUTTON GROUP RAD1,
             R3 RADIOBUTTON GROUP RAD1.
SELECTION-SCREEN ULINE /1(50).
SELECTION-SCREEN COMMENT /10(30) COMM2.
SELECTION-SCREEN ULINE.
PARAMETERS : S1 RADIOBUTTON GROUP RAD2,
             S2 RADIOBUTTON GROUP RAD2,
             S3 RADIOBUTTON GROUP RAD2.
SELECTION-SCREEN ULINE /1(50).
```



SELECT-OPTIONS

Example of Bank Lines, Underlines and Comments

Radio Button Group 1

R1	●
R2	○
R3	○

Radio Button Group 2

S1	●
S2	○
S3	○

Example of Blank Lines,Underlines and Comments

Placing Several Elements On a Single Line

To position a set of parameters or comments on a single line on the selection screen, you must declare the elements in a block enclosed by the following two statements:

Syntax

```
SELECTION-SCREEN BEGIN OF LINE.
  ...
SELECTION-SCREEN END OF LINE.
```

Note that the selection text (name of the parameter or text element) is not displayed when you use this option. To display a selection text you must use SELECTION-SCREEN statement with the COMMENT option.

Note :   Do not use a slash with the format option <pos(len)

Example :

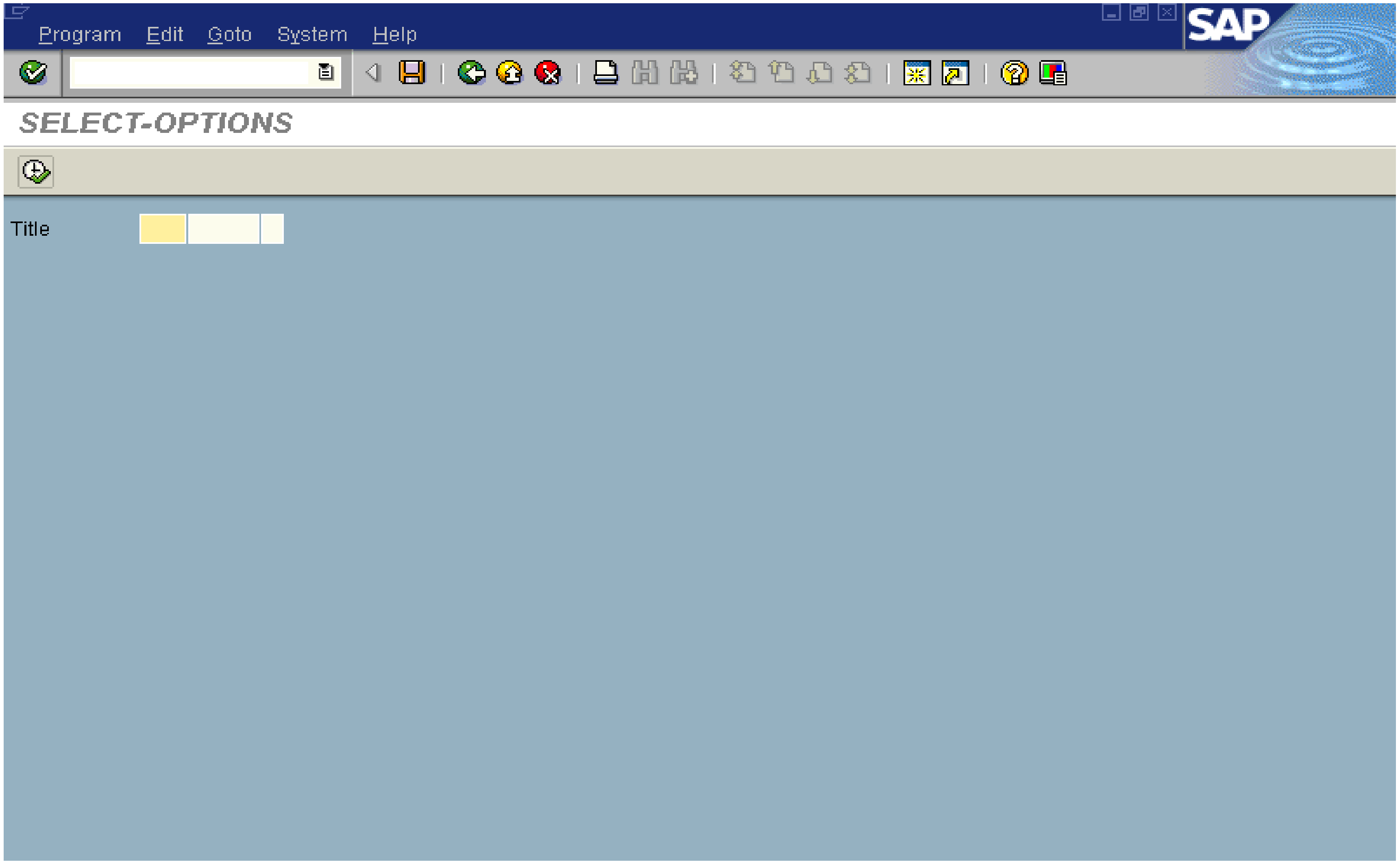
```
SELECTION-SCREEN BEGIN OF LINE.
SELECTION-SCREEN COMMENT 1(10) TEXT-001.
PARAMETERS : P1(3), P2(5), P3(1).
SELECTION-SCREEN END OF LINE.
```

Positioning an Element

To position the next parameter or comment on the selection screen, use the POSITION option with the SELECTION-SCREEN statement.

Syntax

```
SELECTION-SCREEN POSITION <pos>.
```



SELECT-OPTIONS

Title

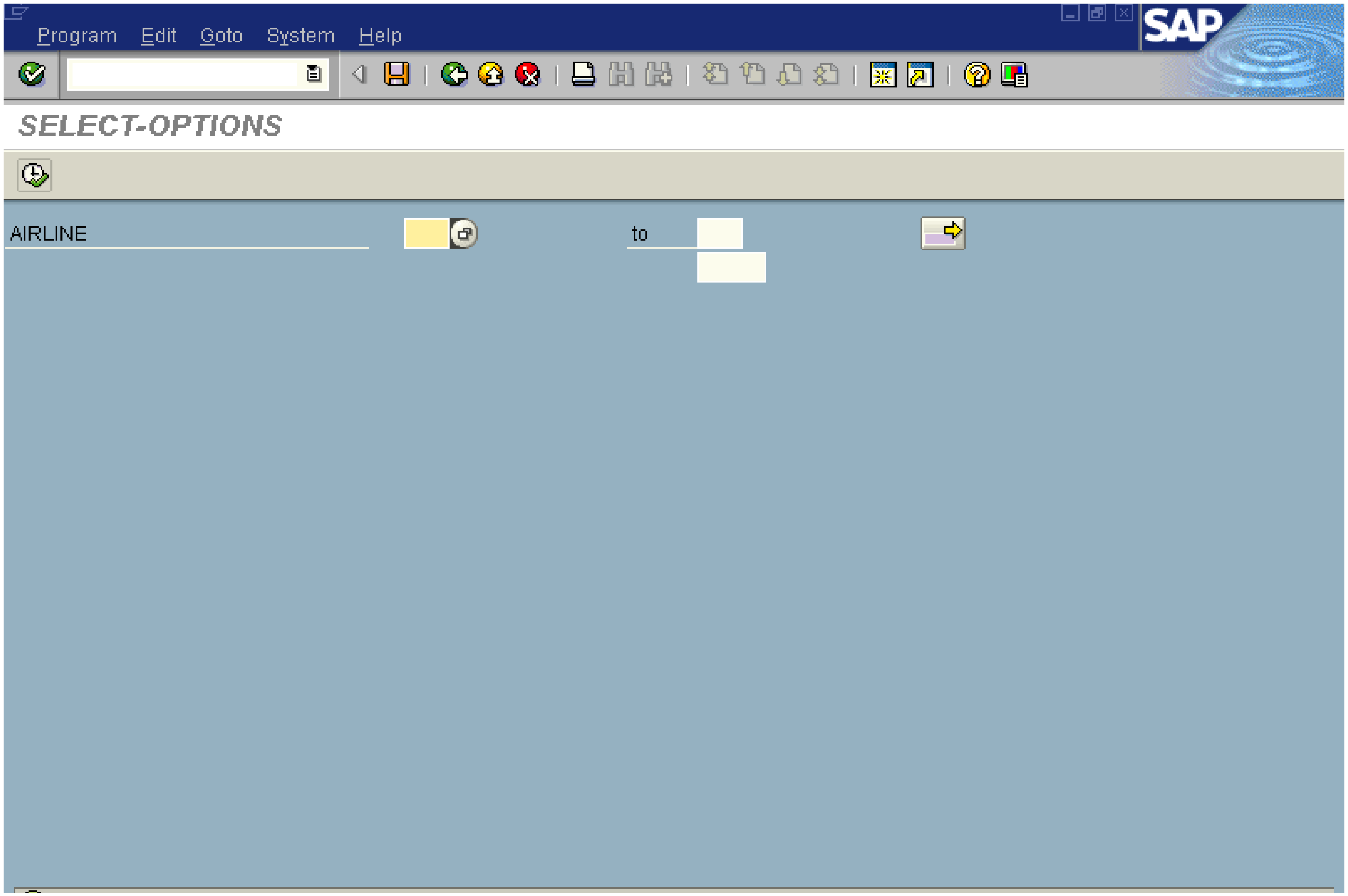
Placing Several Elements on a Single Line

For <pos>, you can specify a number, POS_LOW, or POS_HIGH.

Note:
Use the POSITION option only between the BEGIN OF LINE and END OF LINE options.

Example:

```
REPORT SAPMZTST.
TABLES SPFLI.
SELECT-OPTIONS AIRLINE FOR SPFLI-CARRID.
SELECTION-SCREEN BEGIN OF LINE.
   SELECTION-SCREEN POSITION POS_HIGH.
   PARAMETERS FIELD(5).
SELECTION-SCREEN END OF LINE.
```



SELECT-OPTIONS

AIRLINE to

Example of Positioning an Element

Creating Blocks of Elements

Syntax

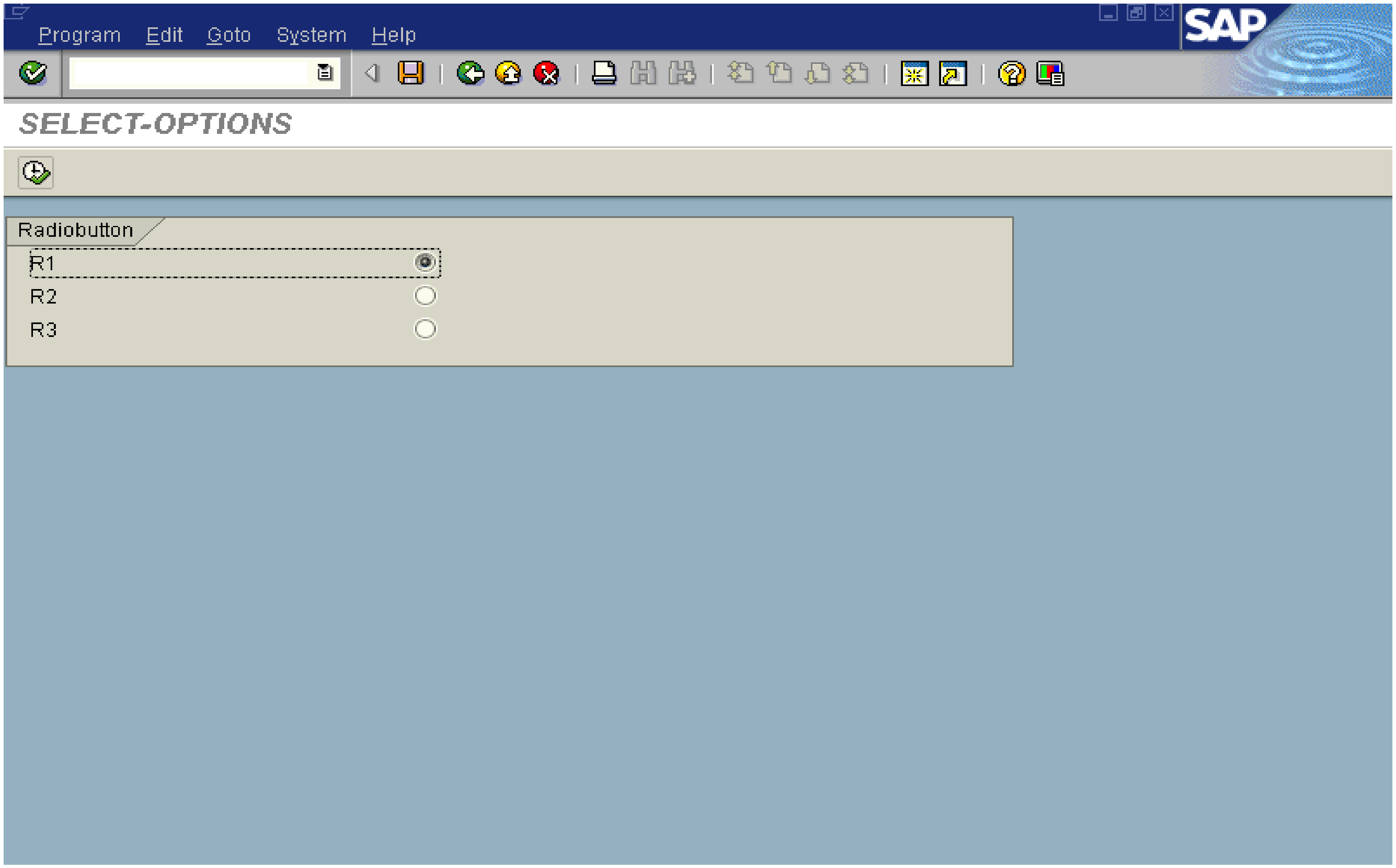
```
SELECTION-SCREEN BEGIN OF BLOCK <block>
        [WITH FRAME [TITLE <title>]] [NO INTERVALS].
          ...
SELECTION-SCREEN END OF BLOCK <block>.
```

•You must define a name <block> for each block. You can nest blocks.
•If you add the WITH FRAME option, a frame will be drawn around the block.
•You can add a title to each frame by using the TITLE option .
•If you use the NO INTERVALS option, the system processes all SELECT-OPTIONS statements in this block as if they had this option .

Example :

```
SELECTION-SCREEN BEGIN OF BLOCK RAD1 WITH FRAME TITLE
TEXT-002.
    PARAMETERS R1 RADIOBUTTON GROUP GR1.
    PARAMETERS R2 RADIOBUTTON GROUP GR1.
    PARAMETERS R3 RADIOBUTTON GROUP GR1.
SELECTION-SCREEN END OF BLOCK RAD1.
```

On the selection screen, the three radio buttons R1, R2, R3 form a block, which is surrounded by a frame and has the title specified in the text symbol 002.



SELECT-OPTIONS

Radiobutton

R1 [✓ ●]

R2 ○

R3 ○

Creating Block of Elements

Creating Pushbuttons in the Application Toolbar

You can create up to five pushbuttons in the application toolbar on the selection screen. These buttons are also automatically connected to function keys.

Syntax

SELECTION-SCREEN FUNCTION KEY <i>.

- **<i> must be between 1 and 5.**
- **You must specify the text to appear on the buttons during runtime in ABAP/4 Dictionary fields SSCRFIELDS-FUNCTXT_0<i>.**

•You must declare SSCRFIELDS with a TABLES statement.

•When the user clicks this button, FC0<i> is entered in the field SSCRFIELDS-UCOMM, which can be checked during the event AT SELECTION-SCREEN.

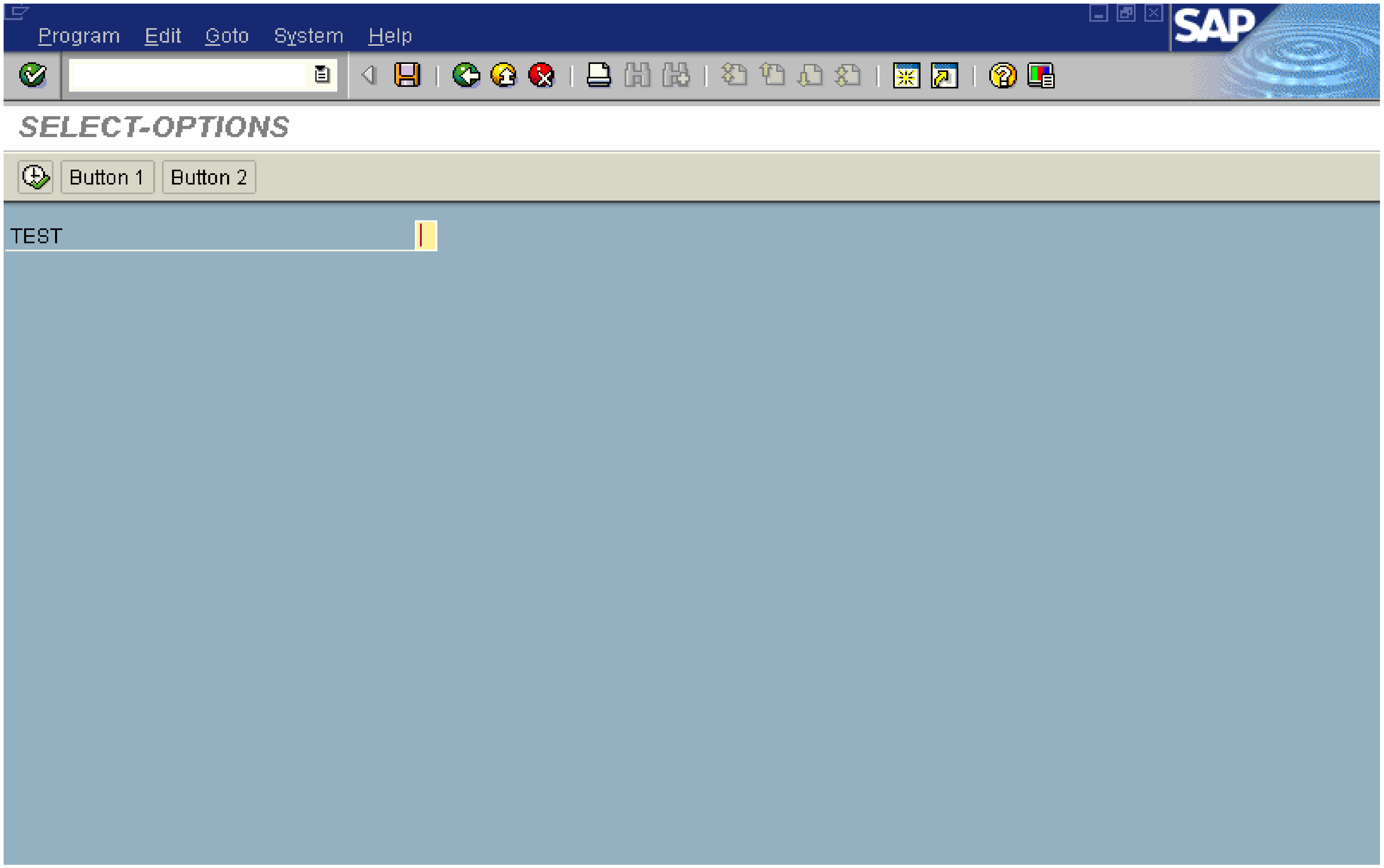
Example:

```
TABLES SSCRFIELDS.
DATA FLAG.
PARAMETERS TEST:
SELECTION-SCREEN FUNCITON KEY 1.
SELECTION-SCREEN FUNCITON KEY 2.
INITIALIZATION.
SSCRFIELDS-FUNCTXT_01 = ‘Button 1’.
SSCRFIELDS-FUNCTXT_02 = ‘Button 2’.
```

```
AT SELECTION-SCREEN.

 IF SSCRFIELDS-UCOMM = ‘FC01’.
        FLAG = ‘1’.
 ELSEIF SSCRFIELDS-UCOMM = ‘FC02’.
FLAG = ‘2’.
ENDIF.
START-OF-SELECTION.
IF FLAG = ‘1’.
WRITE : / ’Button 1 was clicked’.
ELSEIF FLAG = ‘2’.
Write : / ‘Button 2 was clicked’.
```

This example causes two pushbuttons with the texts ‘Button 1’ and ‘Button 2’ to appear in the application toolbar on the selection screen.



Creating Pushbuttons in the application Toolbar

Creating Pushbuttons on the Selection Screen

Syntax

```
SELECTION SCREEN PUSHBUTTON [/]<pos(len)> <name>
                USER-COMMAND <ucom> [MODIF ID <key>].
```

The parameters [/]<pos(len)>, <name>, and the MODIF ID option are the same as described for the COMMENT option in Comments.

The text specified in <name> is the pushbutton text.

For <ucom>, you must specify a code of up to four characters.

You must declare SSCRFIELDS by using a TABLES statement.

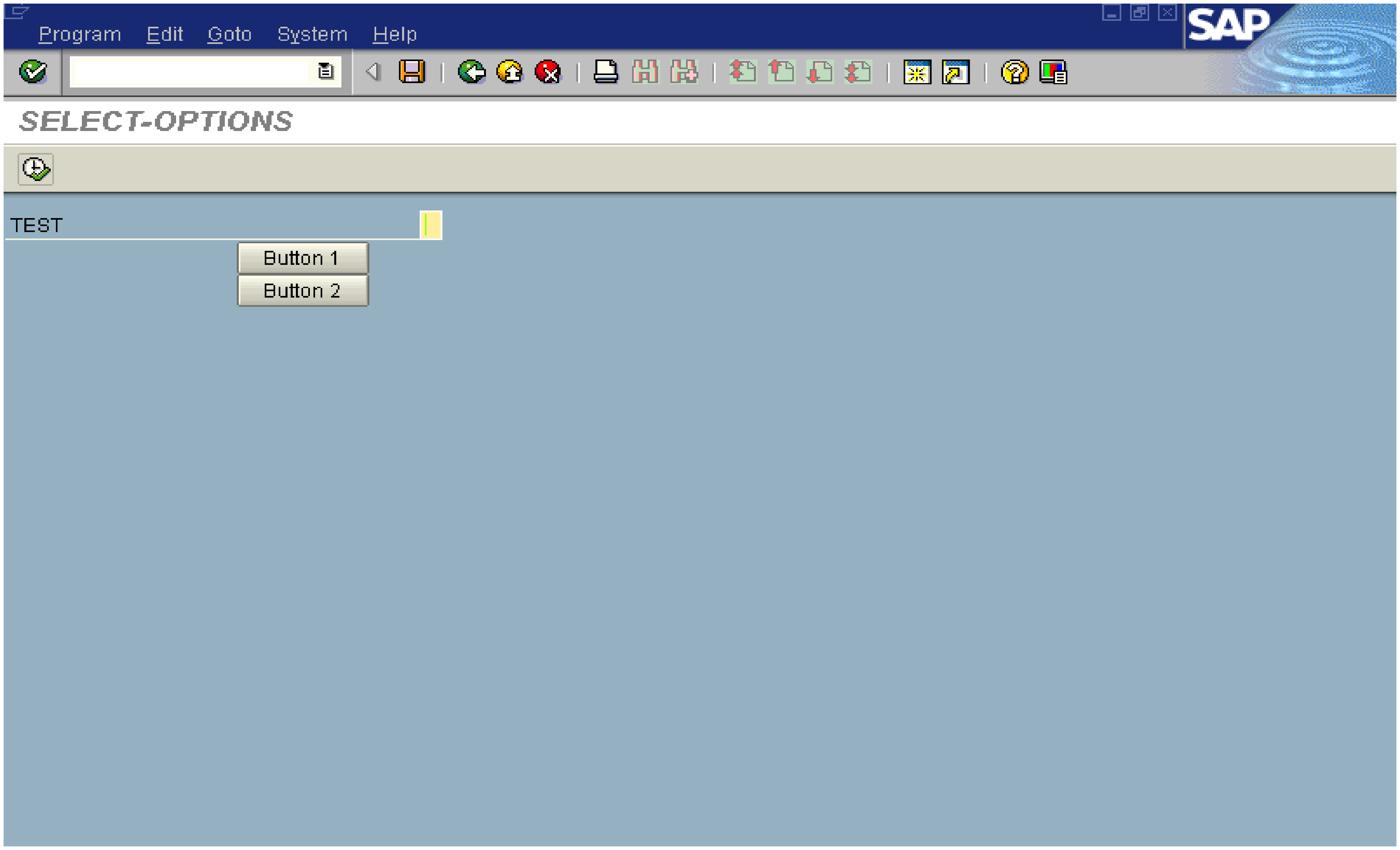
Example:

```
TABLES SSCRFIELDS.
DATA FLAG.
PARAMETERS TEST:
SELECTION-SCREEN PUSHBUTTON /20(10) BUT1
                                        USER-COMMAND CLI1.
SELECTION-SCREEN PUSHBUTTON /20(10) TEXT-020
                                        USER-COMMAND CLI2.
INITIALIZATION.
BUT1 =  ‘Button 1’.
AT SELECTION-SCREEN.
 IF SSCRFIELDS-UCOMM = ‘FCI1’.
       FLAG = ‘1’.
 ELSEIF SSCRFIELDS-UCOMM = ‘CLI2’.
FLAG = ‘2’.
ENDIF.
```

```
START-OF-SELECTION.

IF FLAG = ‘1’.
WRITE : / ’Button 1 was clicked’.
ELSEIF FLAG = ‘2’.
Write : / ‘Button 2 was clicked’.
ENDIF.
```

If the text symbol TEXT-020 is defined as ‘button 2’, this example causes two pushbuttons with the texts ‘Button 1’ and ‘Button 2’ to appear on the selection screen.



Creating Pushbuttons on the Selection Screen

What is a Variant?

•A variant is an interface to the selection screen.

•If you want to run the same report program with the same selections at regular intervals (for example, for monthly sales statistics), you would not want to enter the same values each time. ABAP/4 offers you a possibility to combine the desired values for all these selections in one selection set. Such a selection set is called a variant.

•You can create as many different selection sets as you like for each report program and they remain assigned only to the report program in question.

•Variants you use online may have different functions than those you use in background processing.

•**Using Variants Online:**

•**Online, starting a report via variant saves the user work, since he does not have to enter the same selection set again and again each time the selection screen appears. In addition, using a variant minimizes input errors.**

•**Using Variants in Background Processing:**

•**In background processing, a variant is the only possibility you have to pass values for the selections. Therefore, report programs executed in the background must be started via a variant .**

Variable in variant

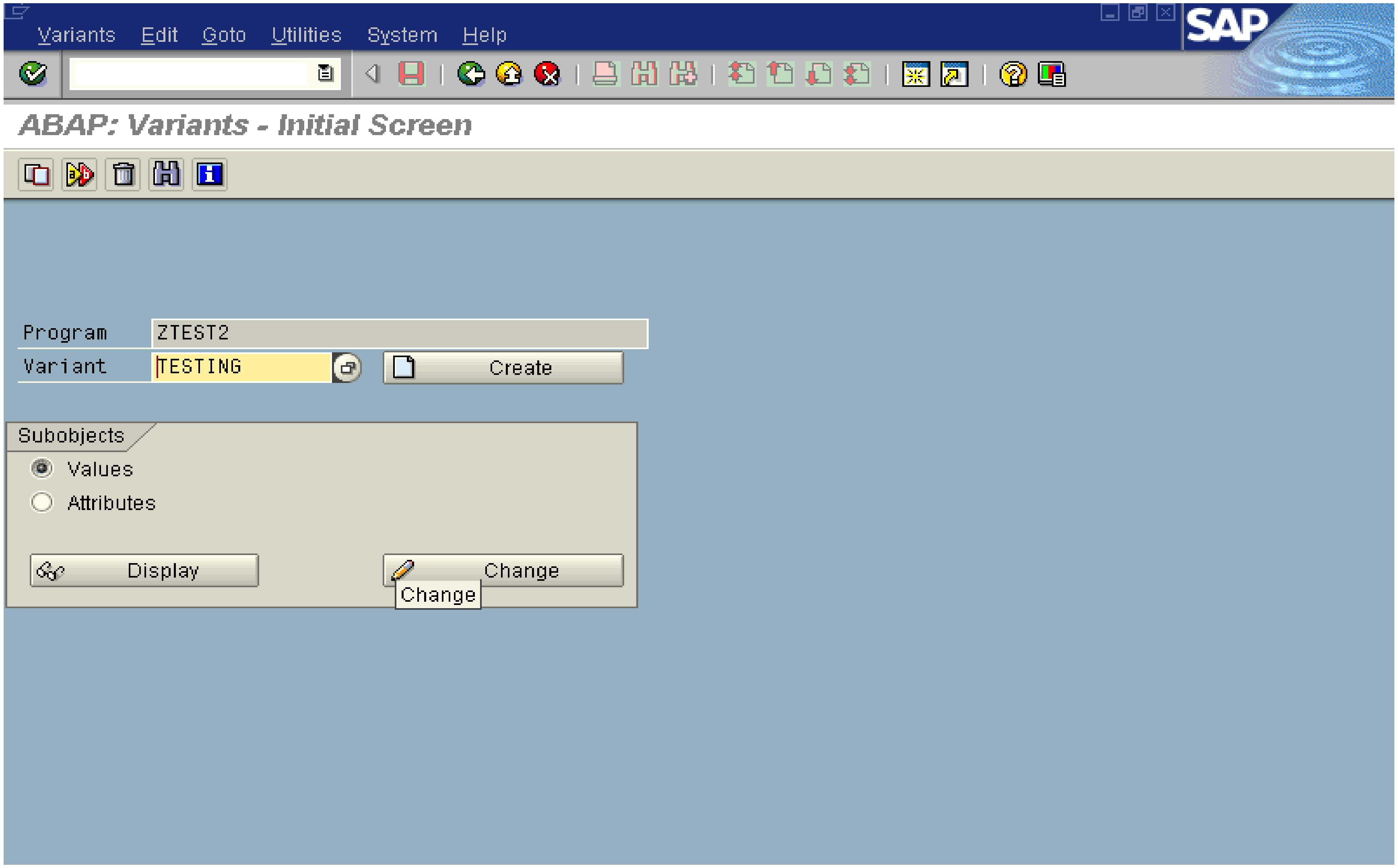
While creating a variant , you can create it as a variable variant.
This setting is done in the attributes screen while creating variant.

You can do this in the following three ways :

- variable date calculation
- user-specific values
- values from Table TVARV

CREATING AND CHANGING VARIANTS:

To create or change a variant, start from the ABAP/4 Editor: Initial screen. Enter the name of the program you want to maintain a variant for, mark Variants and choose Change. The ABAP/4: Variants - Initial Screen appears. You can now display a list of existing variants for the program, create a new variant, or modify an existing one.



ABAP: Variants - Initial Screen

Program ZTEST2

Variant TESTING

Subobjects

[✓ Values]

Attributes

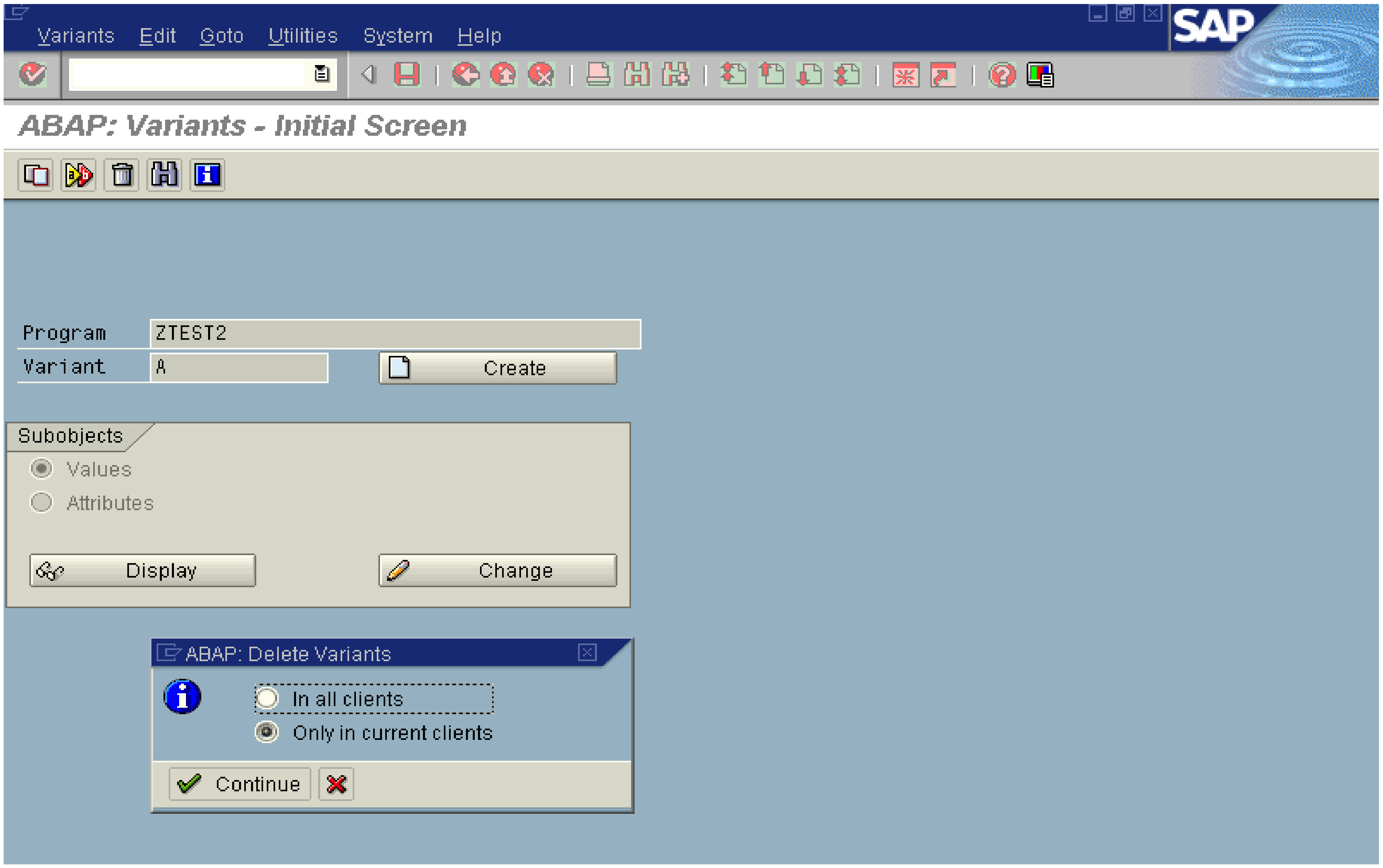
Creating / Changing a Variant

DELETING VARIANTS:

To delete a variant, enter the report name and the variant name in the variants initial screen. Then, choose Variants -> Delete.

PRINTING VARIANTS:

To print a variant, enter the name of the variant in the variants initial screen, choose the values for display and click print. Note that you cannot print the values if you are in change mode.

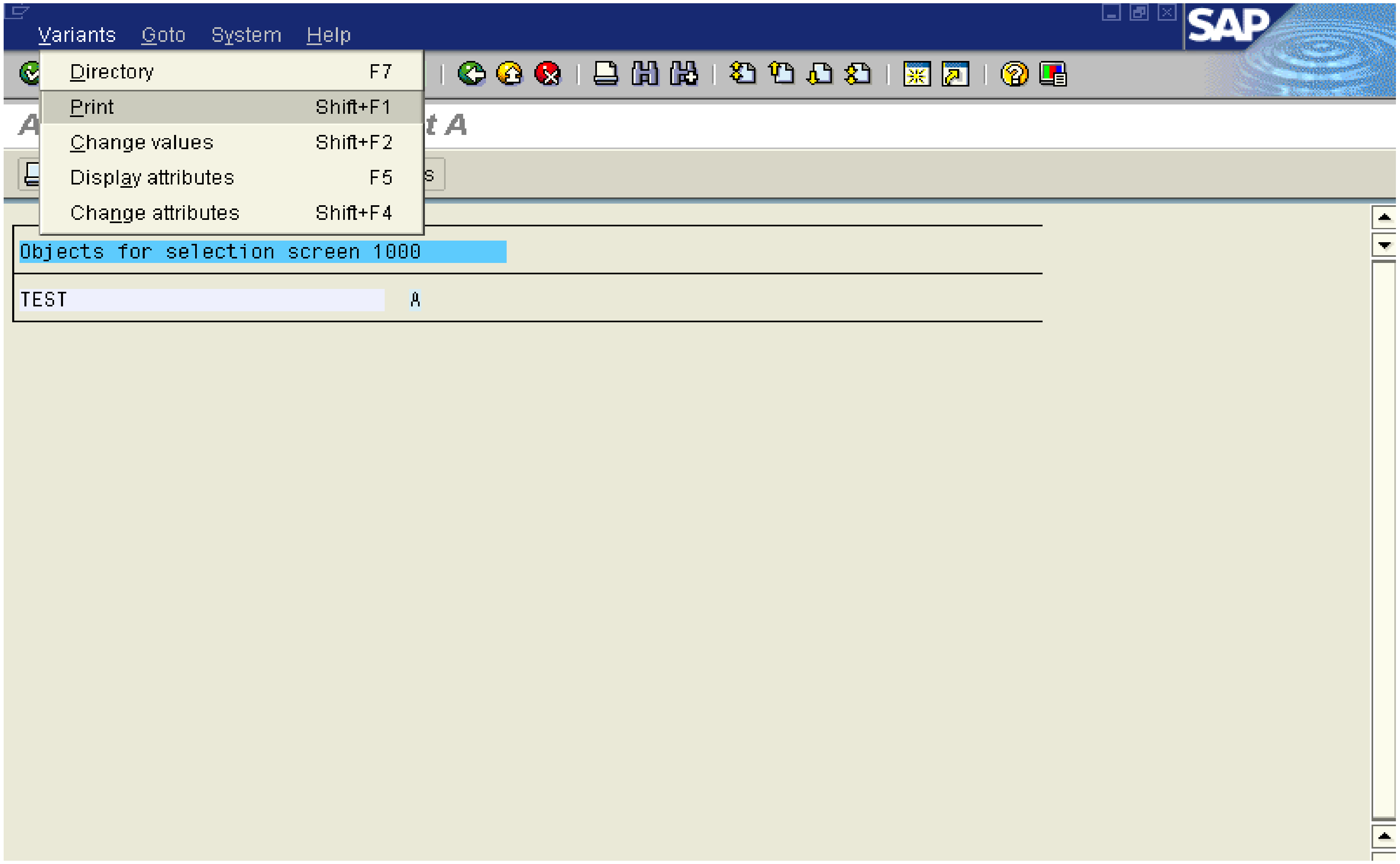

Variants Edit Goto Utilities System Help
SAP
ABAP: Variants - Initial Screen
Create
Subobjects
Display
Change
ABAP: Delete Variants
Continue

Program: ZTEST2
Variant: A

Subobjects: [✓ Values] Attributes

ABAP: Delete Variants: In all clients [✓ Only in current clients]

Deleting Variants



```
Objects for selection screen 1000

TEST                              A
```

Printing Variant in Display mode

Controlling the Flow of ABAP/4 Programs by Events

ABAP/4 is an event-driven language. This means that the general flow of an ABAP/4 program is controlled by external events. Parts of a program form processing blocks, which are assigned to a certain event. The system always starts a processing block when the corresponding event occurs.

What is a processing block ?

You can define processing blocks in your ABAP/4 program by using event keywords. All statements between two event keywords or between an event keyword and a FORM statement form a processing block. When an event occurs, the system processes the processing block after the corresponding event keyword. Each statement in an ABAP/4 report program is part of a processing block or a subroutine.

EVENTS

INITIALIZATION	Point before the selection screen is displayed
AT SELECTION-SCREEN	Point after processing user input on the selection screen while the selection screen is still active
START-OF-SELECTION	Point after processing the selection screen
GET <table>	Point at which the logical database offers a line of the database table <table>.
END-OF-SELECTION	Point after processing all lines offered by the logical database.

Events occurring during the processing of the output list of a report.

TOP-OF-PAGE	Point during list processing when a new page is started.
END-OF-PAGE	Point during list processing when a page is ended.

Event keywords to write a program for interactive reporting:

AT LINE-SELECTION	Point at which the user selects a line
AT USER-COMMAND	Point at which the user presses a function key or enters a command in the command field.
AT PF<nn>	Point at which the user presses the function key with the function code PF<n>

MESSAGES

They are grouped by language, a two-character ID, and a three-digit number. From your program, you can send a message with different qualifications:

A	Abend; the current transaction is stopped
E	Error; the system waits for new input data
I	Information; after pressing ENTER, the system continues processing
S	Confirmation; the message appears on the next screen
W	Warning; you can change the input data or continue by pressing ENTER

You must specify the MESSAGE-ID behind the REPORT statement of your program.

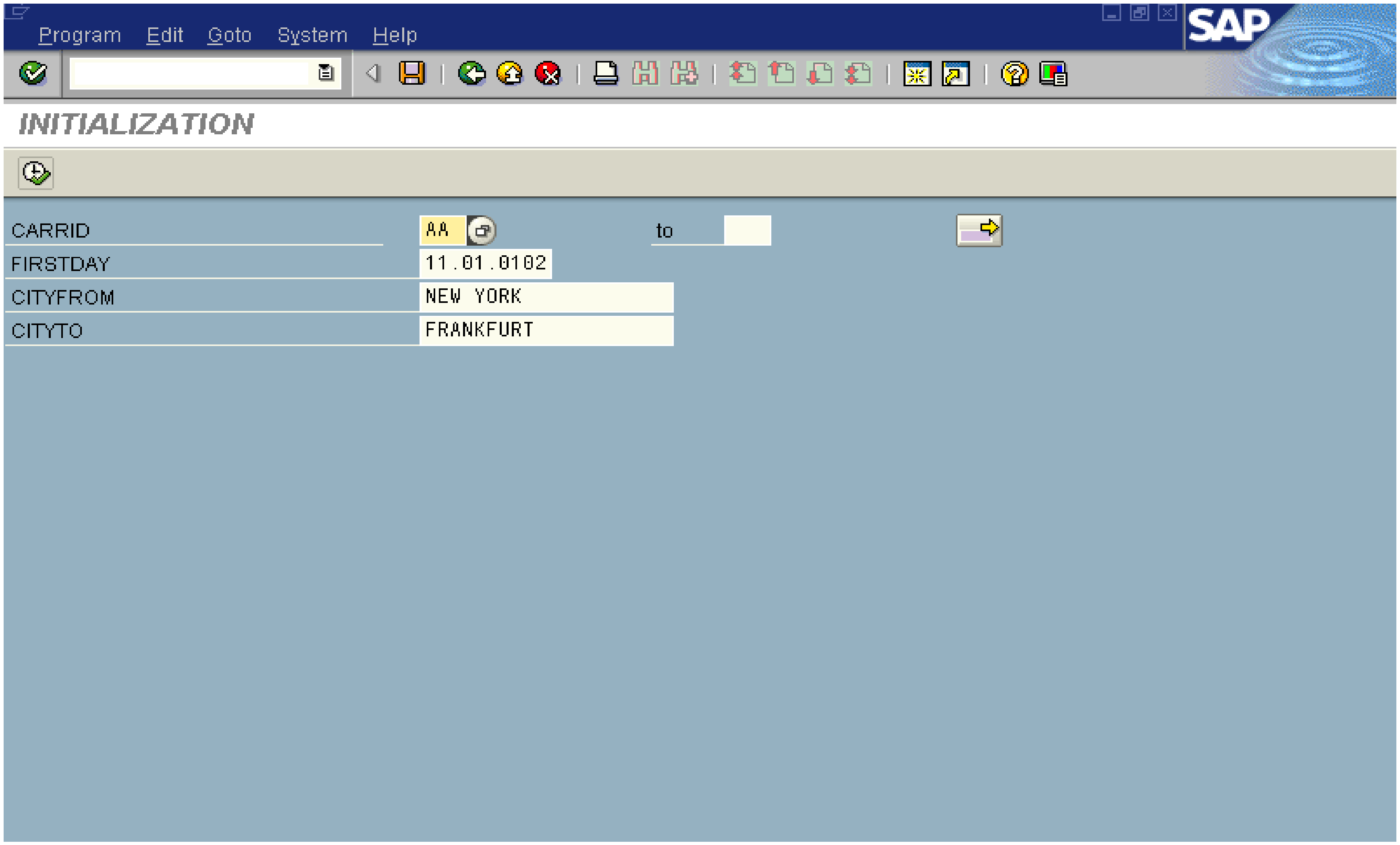
INITIALIZATION

When you start a program in which a selection screen is defined (either in the program itself or in the linked logical database program), the system normally processes this selection screen first. If you want to execute a processing block before the selection screen is processed, you can assign it to the event keyword INITIALIZATION.

EXAMPLE:

```
REPORT SAPMZTST.
TABLES SPFLI.

SELECT-OPTIONS : CARRID FOR SPFLI-CARRID.
PARAMETERS : FIRSTDAY LIKE SY-DATUM DEFAULT SY-DATUM,
             CITYFROM LIKE SPFLI-CITYFROM,
             CITYTO LIKE SPFLI-CITYTO.
INITIALIZATION.
CITYFROM = 'NEW YORK'.
CITYTO = 'FRANKFURT'.
CARRID-SIGN = 'I'.
CARRID-OPTION = 'EQ'.
CARRID-LOW = 'AA'.
APPEND CARRID.
FIRSTDAY+(2) = '01'.
```



INITIALIZATION

CARRID	AA	to	
FIRSTDAY	11.01.0102		
CITYFROM	NEW YORK		
CITYTO	FRANKFURT		

Example of Initialization

AT SELECTION-SCREEN

The event keyword AT SELECTION-SCREEN provides you with several possibilities to carry out processing blocks while the system is processing the selection screen. To react on different events, that can occur when the selection screen is processed, the keyword AT SELECTION-SCREEN has various options:

- **AT SELECTION-SCREEN ON <field>.**
- **AT SELECTION-SCREEN ON END OF <seltab>.**
- **AT SELECTION-SCREEN ON VALUE-REQUEST FOR <field>.**
- **AT SELECTION-SCREEN ON HELP-REQUEST FOR <field>.**
- **AT SELECTION-SCREEN ON RADIOBUTTON GROUP <radi>.**
- **AT SELECTION-SCREEN ON BLOCK <block>.**
- **AT SELECTION-SCREEN OUTPUT.**

START-OF-SELECTION

The event START-OF-SELECTION gives you the possibility of creating a processing block after processing the selection screen and before accessing database tables.

If you do not specify any event keywords in your program, all statements of the program before a FORM statement form the START-OF-SELECTION processing block.

NOTE:

- **All statements between an ENDFORM statement and an event keyword or between an ENDFORM statement and the end of the program form a processing block that is never processed.**

•Do not place any statements there. Place all subroutines at the end of your program.

•Statements which do not follow an event keyword or a FORM-ENDFORM block are automatically part of the processing block of the default event START-OF-SELECTION . This has the following consequences:

•If you write statements between the REPORT statements and the first event keyword or FORM statement, these statements are included in the START-OF-SELECTION processing block.

•If no START-OF-SELECTION keyword is included in your report, these statements form the entire START-OF-SELECTION processing block.

•If a START-OF-SELECTION keyword is included in your report, these statements are inserted at the beginning of this block.

END-OF-SELECTION

To define a processing block after the system has read and processed all database tables of a logical database, use the keyword END-OF-SELECTION.