Copyright

© Copyright 2001 SAP AG. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

Some software products marketed by SAP AG and its distributors contain proprietary software components of other software vendors.

Microsoft®, WINDOWS®, NT®, EXCEL®, Word®, PowerPoint® and SQL Server® are registered trademarks of Microsoft Corporation.

IBM®, DB2®, OS/2®, DB2/6000®, Parallel Sysplex®, MVS/ESA®, RS/6000®, AIX®, S/390®, AS/400®, OS/390®, and OS/400® are registered trademarks of IBM Corporation.

ORACLE® is a registered trademark of ORACLE Corporation.

INFORMIX®-OnLine for SAP and Informix® Dynamic Server™ are registered trademarks of Informix Software Incorporated.

UNIX®, X/Open®, OSF/1®, and Motif® are registered trademarks of the Open Group.

HTML, DHTML, XML, XHTML are trademarks or registered trademarks of W3C®, World Wide Web Consortium, Massachusetts Institute of Technology.

JAVA® is a registered trademark of Sun Microsystems, Inc.

JAVASCRIPT® is a registered trademark of Sun Microsystems, Inc., used under license for technology invented and implemented by Netscape.

SAP, SAP Logo, R/2, R/IV, R/3, ABAP, SAP ArchiveLink, SAP Business Workflow, WebFlow, SAP EarlyWatch, BAPI, SAPPHIRE, Management Cockpit, mySAP.com Logo and mySAP.com are trademarks or registered trademarks of SAP AG in Germany and in several other countries all over the world. All other products mentioned are trademarks or registered trademarks of their respective companies.
## Icons

<table>
<thead>
<tr>
<th>Icon</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Caution Icon]</td>
<td>Caution</td>
</tr>
<tr>
<td>![Example Icon]</td>
<td>Example</td>
</tr>
<tr>
<td>![Note Icon]</td>
<td>Note</td>
</tr>
<tr>
<td>![Recommendation Icon]</td>
<td>Recommendation</td>
</tr>
<tr>
<td>![Syntax Icon]</td>
<td>Syntax</td>
</tr>
</tbody>
</table>
## Contents

**BC SAP Graphics: Programming Interfaces** .................................................... 6
  Basics ............................................................................................................... 7
  SAP Graphics Programs and Function Modules ............................................... 8
  Providing Interactive Capabilities .................................................................. 10
  **Calling SAP Business Graphics** ................................................................. 11
    Overview ....................................................................................................... 12
    The GRAPH_2D Function Module ................................................................. 14
    The GRAPH_3D Function Module ................................................................. 16
    The GRAPH_MATRIX_2D Function Module .................................................... 18
    The GRAPH_MATRIX_3D Function Module .................................................... 20
    The GRAPH_MATRIX_4D Function Module .................................................... 22
    The GRAPH_MATRIX Function Module ....................................................... 24
    The GRAPH_BUSG_MENU_SET Function Module .......................................... 26
    The GET_BUSG_PARAM Function Module ..................................................... 27
  Sample Programs ............................................................................................. 28
  **Calling SAP Hierarchy Graphics** ................................................................. 29
    Overview ....................................................................................................... 30
    The GRAPH_HIERARCHY Function Module ................................................... 32
    The GRAPH_HIERARCHY_MENUE Function Module ..................................... 34
    The GET_HIER_PARAM Function Module ..................................................... 37
  **Calling SAP Pushbutton Graphics** ............................................................... 38
    Overview ....................................................................................................... 39
    The GRAPH_BUTTON_MATRIX Function Module ......................................... 41
    The GRAPH_BUTTON_MENUE Function Module .......................................... 43
    The GET_BMAT_PARAM Function Module ................................................... 45
  Sample Programs ............................................................................................. 46
  **Calling SAP Gantt Chart** ............................................................................. 47
    Overview ....................................................................................................... 48
    The GANTT_DIAGRAMM Function Module .................................................... 50
    The GRAPH_GANTT Function Module .......................................................... 52
    The GET_GANTT_PARAM Function Module ................................................... 54
  **Calling SAP HPGL Display** ........................................................................ 55
    Overview ....................................................................................................... 56
    The GRAPH_HPGL Function Module ............................................................ 58
    The GET_HPGL_PARAM Function Module .................................................... 59
  **Calling SAP Text Matrix** ............................................................................ 60
    Overview ....................................................................................................... 61
    The TEXT_MATRIX Function Module .......................................................... 63
  **Calling SAP Network Graphics** ................................................................. 64
    Overview ....................................................................................................... 65
  Confirm-Mode Processing of User Actions ...................................................... 67
BC SAP Graphics: Programming Interfaces
Basics

The following topics deal with basic mechanisms of SAP graphics programming.

*SAP Graphics Programs and Function Modules [Page 8]*

*Providing Interactive Capabilities [Page 10]*
SAP Graphics Programs and Function Modules

The following table lists the SAP Graphics programs, their graphing capabilities and the available function modules for calling each of these programs. The table also contains the utility function modules.

The graphics function modules call the relevant graphics program directly (for example GRAPH_MATRIX_4D).

The other function modules perform utility functions and do not call a specific graphics program (GRAPH_ACTION, GRAPH_RECEIVE, and the GET_xxxx_PARAM modules).

The distinction is important because the utility function modules interact closely with the functions modules that call a graphics program.

This guide contains sample programs [Page 114] for all applications. These programs demonstrate function module calls and the use of the dialog parameters. You can use these programs as models for programming your own graphics applications.

For further sample programs calling graphics applications, use transaction code gral in the R/3 System.

SAP Graphics Function Modules Overview

<table>
<thead>
<tr>
<th>Graphics program</th>
<th>Capabilities</th>
<th>Function modules</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAP Business Graphics</td>
<td>Graphing diagrams, charts, etc</td>
<td>GRAPH_2D</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GRAPH_3D</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GRAPH_MATRIX_2D</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GRAPH_MATRIX_3D</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GRAPH_MATRIX_4D</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GRAPH_MATRIX</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GRAPH_BUSG_MENU_SET</td>
</tr>
<tr>
<td>SAP Hierarchy Graphics</td>
<td>Graphing hierarchical graphs</td>
<td>GRAPH_HIERARCHY</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GRAPH_HIERARCHY_MENUE</td>
</tr>
<tr>
<td>SAP Pushbutton Graphics</td>
<td>Graphing pushbutton pads,</td>
<td>GRAPH_BUTTON_MATRIX</td>
</tr>
<tr>
<td></td>
<td>time planning information</td>
<td>GRAPH_BUTTON_MENUE</td>
</tr>
<tr>
<td>SAP Statistical Graphics</td>
<td>Graphing statistical data in</td>
<td>STAT_GRAPH</td>
</tr>
<tr>
<td></td>
<td>curves</td>
<td>STAT_GRAPH_REF</td>
</tr>
<tr>
<td>SAP Gantt Chart</td>
<td>Graphing time planning</td>
<td>GANTT_DIAGRAMM</td>
</tr>
<tr>
<td></td>
<td>information</td>
<td>GRAPH_GANTT</td>
</tr>
<tr>
<td>SAP HPGL Display</td>
<td>Graphing graphic data in HPGL</td>
<td>GRAPH_HPGL</td>
</tr>
<tr>
<td></td>
<td>format</td>
<td></td>
</tr>
<tr>
<td>SAP Graphics Programs and Function Modules</td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------------------------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SAP Text Matrix</td>
<td>Graphing text windows</td>
<td>TEXT_MATRIX</td>
</tr>
<tr>
<td>SAP Network Graphics</td>
<td>Graphing network and hierarchy graphs</td>
<td>CNET_GRAPHIC_NETWORK, CNET_GRAPHIC_HIERARCHY, CNET_GRAPHIC_CLUSTER, CNET__...</td>
</tr>
<tr>
<td>SAP Portfolio Graphics</td>
<td>Graphing comparative business trends</td>
<td>GRAPH_PORT</td>
</tr>
<tr>
<td>SAP Bar Chart Graphics</td>
<td>Graphing bar charts</td>
<td>BARC_GRAPHIC_PBO, BARC_GRAPHIC_PAIX, BARC__...</td>
</tr>
<tr>
<td>SAP Structural Graphics</td>
<td>Graphing hierarchical structures</td>
<td>GRAPH_STRUCTURAL, and others for controlling SAP Structural Graphics</td>
</tr>
<tr>
<td>(none)</td>
<td>Defining CUA items (using Menu Painter)</td>
<td>GRAPH_SET_CUA_STATUS</td>
</tr>
<tr>
<td>(none)</td>
<td>Defining additional CUA items (For new programs, use GRAPH_SET_CUA_STATUS)</td>
<td>GRAPH_ACTION</td>
</tr>
<tr>
<td>(none)</td>
<td>Defining additional CUA items (for SAP Network Graphics only, for new programs, use GRAPH_SET_CUA_STATUS)</td>
<td>CNET_GRAPH_ACTION</td>
</tr>
<tr>
<td>(none)</td>
<td>Closing windows, performing waiting (For new programs, use GRAPH_RECEIVE)</td>
<td>GRAPH_DIALOG</td>
</tr>
<tr>
<td>(none)</td>
<td>Performing waiting, receiving messages about user input from the graphics program</td>
<td>GRAPH_RECEIVE</td>
</tr>
<tr>
<td>(none)</td>
<td>Obtaining export parameters set to their proper values</td>
<td>GET_xxxx_PARAM</td>
</tr>
</tbody>
</table>
Providing Interactive Capabilities

ABAP programs calling the graphics function modules and the utility function modules can provide the online user with extensive interactive capabilities. The online user can open multiple windows, each with a different graphics program running in it, and enter input to any of them.

To make these interactions possible, the function modules perform all the communications tasks needed to carry on the dialog between the application program and the graphics program. In addition, the modules execute the following operations:

- open a window on the user's desktop
- start up the graphics program in the window
- send data to the running graphics program
- receive messages from the graphics program
- report the messages back to the ABAP program

Thus, the function module is the mediator (see the following figure) in all communication between the ABAP program and the graphics program.

When reading this guide, it is important to keep the above diagram in mind. The sections in this guide make continual reference to the three participants (ABAP program, function module, and graphics program) in the story. Keeping firmly in mind which programs are responsible for which actions will greatly ease your programming efforts.
Calling SAP Business Graphics

An ABAP program specifies the data and calls up an appropriate function module. The SAP Business Graphics program generates a two-dimensional or three-dimensional graph from the data supplied.

**Overview [Page 12]**

The GRAPH_2D Function Module [Page 14]

The GRAPH_3D Function Module [Page 16]

The GRAPH_MATRIX_2D Function Module [Page 18]

The GRAPH_MATRIX_3D Function Module [Page 20]

The GRAPH_MATRIX_4D Function Module [Page 22]

The GRAPH_MATRIX Function Module [Page 24]

The GRAPH_BUSG_MENU_SET Function Module [Page 26]

The GET_BUSG_PARAM Function Module [Page 27]

Sample Programs [Page 28]
Overview

The SAP Business Graphics program generates a business graph (bar chart, pie chart, and so on) in two-dimensional and three-dimensional views. The online user can modify the display using menu options.

A typical business graph looks as shown in the following figure.

Functions for Drawing Business Graphics

You can call SAP Business Graphics from an ABAP program, and generate a business graphic at the user's workstation terminal. Your program specifies the data and calls one of the following function modules:

- GRAPH_2D
- GRAPH_3D
- GRAPH_MATRIX_2D
- GRAPH_MATRIX_3D
- GRAPH_MATRIX_4D
- GRAPH_MATRIX

The basic difference between these modules is ease of use. All the GRAPH_MATRIX function modules are more flexible but also more complicated to use.
Overview

- You can call all of the GRAPH_MATRIX modules with large data tables, and indicate which parts of the table you want graphed.
  
  By contrast, the GRAPH_2D and GRAPH_3D function modules graph all the data you send.

- You can set the initial display options with the GRAPH_MATRIX modules. These are the same display options that an online user sets using menus.

- With the GRAPH_2D and GRAPH_3D function modules, you get initial displays with the default display options. However, the online user can reset these options as desired.

Special Purpose Functions

To define additional pushbuttons, use the following function module:

- GRAPH_BUSG_MENU_SET

To use the function module, you must call it before calling one of the function modules listed above. GRAPH_BUSG_MENU_SET does not open a graphics window but stores information on defined pushbuttons in static memory. (See the topic on GRAPH_BUSG_MENU_SET for more information.)

In SAP Business Graphics, the utility function modules GRAPH_SET_CUA_STATUS or GRAPH_ACTION are not available for defining pushbuttons.

Programming Interactive Applications

If your ABAP program allows the online user to open multiple windows or enter data with the mouse or keyboard, the application is interactive.

You are provided with two mechanisms for controlling the interaction between ABAP program and online user:

- Dialog programming in Releases prior to 2.1
  
  Using graphics function modules with dialog parameters STAT, M_TYP, PWDID, SUPER, WINID and RWNID)

- Current dialog programming
  
  Using graphics function modules with dialog parameters STAT, PWDID, SUPER, and WINID)
  
  plus utility function modules:
  - GRAPH_RECEIVE (with dialog parameters ACTIVATE_RAISE, ERRORCODE, MCODE, and RWNID)
  - GET_xxxx_PARAM

For new programs, you should use only the second mechanism, graphics function modules with the utility function modules.

Do not use both mechanisms together in a program.

For more information, see the section Programming Interactive Applications. [Page 86]
The GRAPH_2D Function Module

The GRAPH_2D function module calls the SAP Business Graphics program. You send it a list of values, which are displayed in two-dimensional view.

The following table contains a list of all possible parameters you can use with this function module. Of this list, only the DATA table parameter is required.

For a detailed description of parameters and exceptions, see the R/3 system documentation (Tools → ABAP Workbench, Function library button, or transaction code SE37).

Parameters used with GRAPH_2D

<table>
<thead>
<tr>
<th>Name</th>
<th>Status</th>
<th>Initial Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>INBUF **</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>INFORM</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>MAIL_ALLOW</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>PWDID **</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>STAT **</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>SUPER **</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>TIMER</td>
<td>Import</td>
<td>(Do Not Use)</td>
</tr>
<tr>
<td>TITL</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>TYPE</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>VALT</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>WDID</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>WINID **</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>WINPOS</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>WINSZX</td>
<td>Import</td>
<td>'50'</td>
</tr>
<tr>
<td>WINSZY</td>
<td>Import</td>
<td>'50'</td>
</tr>
<tr>
<td>X_OPT</td>
<td>Import</td>
<td>(Do Not Use)</td>
</tr>
<tr>
<td>B_KEY</td>
<td>Export</td>
<td></td>
</tr>
<tr>
<td>B_TYP</td>
<td>Export</td>
<td></td>
</tr>
<tr>
<td>M_TYP **</td>
<td>Export</td>
<td></td>
</tr>
<tr>
<td>MOD_COL</td>
<td>Export</td>
<td></td>
</tr>
<tr>
<td>MOD_ROW</td>
<td>Export</td>
<td></td>
</tr>
<tr>
<td>MOD_VAL</td>
<td>Export</td>
<td></td>
</tr>
<tr>
<td>RBUFF **</td>
<td>Export</td>
<td></td>
</tr>
<tr>
<td>RWNID **</td>
<td>Export</td>
<td></td>
</tr>
</tbody>
</table>
** DATA | Internal Table **

Note: ** denotes a dialog parameter
The GRAPH_3D Function Module

The GRAPH_3D function module calls the SAP Business Graphics program. You send it a table of values, which are displayed in three-dimensional view.

The following table contains a list of all possible parameters you can use with this function module. Of this list, only the DATA table parameter is required.

For a detailed description of parameters and exceptions, see the R/3 system documentation (Tools → ABAP Workbench, Function library button, or transaction code SE37).

Parameters used with GRAPH_3D

<table>
<thead>
<tr>
<th>Name</th>
<th>Status</th>
<th>Initial Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIM1,DIM2</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>INBUF **</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>INFORM</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>MAIL_ALLOW</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>PWDID **</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>STAT **</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>SUPER **</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>TIMER</td>
<td>Import</td>
<td>(Do Not Use)</td>
</tr>
<tr>
<td>TITL</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>TYPE</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>VALT</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>WDid</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>WINID **</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>WINPOS</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>WINSZx</td>
<td>Import</td>
<td>'50'</td>
</tr>
<tr>
<td>WINSZY</td>
<td>Import</td>
<td>'50'</td>
</tr>
<tr>
<td>X_OPT</td>
<td>Import</td>
<td>(Do Not Use)</td>
</tr>
<tr>
<td>B_KEY</td>
<td>Export</td>
<td></td>
</tr>
<tr>
<td>B_TYP</td>
<td>Export</td>
<td></td>
</tr>
<tr>
<td>M_TYP **</td>
<td>Export</td>
<td></td>
</tr>
<tr>
<td>MOD_COL</td>
<td>Export</td>
<td></td>
</tr>
<tr>
<td>MOD_ROW</td>
<td>Export</td>
<td></td>
</tr>
<tr>
<td>MOD_VAL</td>
<td>Export</td>
<td></td>
</tr>
<tr>
<td>RBUFF **</td>
<td>Export</td>
<td></td>
</tr>
</tbody>
</table>
The GRAPH_3D Function Module

<table>
<thead>
<tr>
<th>RWNID **</th>
<th>Export</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATA</td>
<td>Internal Table</td>
</tr>
</tbody>
</table>

Note: ** denotes a dialog parameter

GRAPH_3D provides the means to name individual rows in a graph (the DATA parameter), but not individual columns. Parameters DIM1 and DIM2 are available for naming the row and column axes, but if this is not enough, use the GRAPH MATRIX 3D function module.
The GRAPH_MATRIX_2D Function Module

The GRAPH_MATRIX_2D function module calls SAP Business Graphics and generates a two-dimensional view of your data. You can send a full table of values, and further parameters telling which part of the table to display.

The following table contains a list of all possible parameters you can use with this function module.

For a detailed description of parameters and exceptions, see the R/3 system documentation (Tools → ABAP Workbench, Function library button, or transaction code SE37).

Parameters used with GRAPH_MATRIX_2D

<table>
<thead>
<tr>
<th>Name</th>
<th>Status</th>
<th>Initial Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>INBUF **</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>INFORM</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>MAIL_ALLOW</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>NCOL</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>NROW</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>PWID **</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>STAT **</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>SUPER **</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>TIMER</td>
<td>Import</td>
<td>(Do Not Use)</td>
</tr>
<tr>
<td>TITL</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>VALT</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>WIDID</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>WINID **</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>WINPOS</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>WINSZX</td>
<td>Import</td>
<td>'50'</td>
</tr>
<tr>
<td>WINSZY</td>
<td>Import</td>
<td>'50'</td>
</tr>
<tr>
<td>X_OPT</td>
<td>Import</td>
<td>(Do Not Use)</td>
</tr>
<tr>
<td>B_KEY</td>
<td>Export</td>
<td></td>
</tr>
<tr>
<td>B_TYP</td>
<td>Export</td>
<td></td>
</tr>
<tr>
<td>M_TYP **</td>
<td>Export</td>
<td></td>
</tr>
<tr>
<td>MOD_COL</td>
<td>Export</td>
<td></td>
</tr>
<tr>
<td>MOD_ROW</td>
<td>Export</td>
<td></td>
</tr>
<tr>
<td>MOD_VAL</td>
<td>Export</td>
<td></td>
</tr>
<tr>
<td>RBUFF **</td>
<td>Export</td>
<td></td>
</tr>
</tbody>
</table>
### The GRAPH_MATRIX_2D Function Module

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RWNID **</td>
<td>Export</td>
</tr>
<tr>
<td>DATA</td>
<td>Internal Table</td>
</tr>
<tr>
<td>OPTS</td>
<td>Internal Table</td>
</tr>
<tr>
<td>TCOL</td>
<td>Internal Table</td>
</tr>
</tbody>
</table>

Note: ** denotes a dialog parameter

Of the above list, only the DATA, OPTS and TCOL parameters are required.
The GRAPH_MATRIX_3D Function Module

The GRAPH_MATRIX_3D function module calls SAP Business Graphics with data for display in a three-dimensional graph. You can send a full table of values, and further parameters telling which part of the table to display.

The following table contains a list of all possible parameters you can use with this function module.

For a detailed description of parameters and exceptions, see the R/3 system documentation (Tools → ABAP Workbench, Function library button, or transaction code SE37).

Parameters used with GRAPH_MATRIX_3D

<table>
<thead>
<tr>
<th>Name</th>
<th>Status</th>
<th>Initial Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>COL1,COL2,...COL6</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>DIM1,DIM2</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>INBUF **</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>INFORM</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>MAIL_ALLOW</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>PWDID **</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>STAT **</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>SUPER **</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>TIMER</td>
<td>Import</td>
<td>(Do Not Use)</td>
</tr>
<tr>
<td>TITL</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>VALT</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>WDID</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>WINID **</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>WINPOS</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>WINSZX</td>
<td>Import</td>
<td>'50'</td>
</tr>
<tr>
<td>WINSZY</td>
<td>Import</td>
<td>'50'</td>
</tr>
<tr>
<td>X_OPT</td>
<td>Import</td>
<td>(Do Not Use)</td>
</tr>
<tr>
<td>B_KEY</td>
<td>Export</td>
<td></td>
</tr>
<tr>
<td>B_TYP</td>
<td>Export</td>
<td></td>
</tr>
<tr>
<td>M_TYP **</td>
<td>Export</td>
<td></td>
</tr>
<tr>
<td>MOD_COL</td>
<td>Export</td>
<td></td>
</tr>
<tr>
<td>MOD_ROW</td>
<td>Export</td>
<td></td>
</tr>
<tr>
<td>MOD_VAL</td>
<td>Export</td>
<td></td>
</tr>
<tr>
<td>RBUFF **</td>
<td>Export</td>
<td></td>
</tr>
</tbody>
</table>
The GRAPH_MATRIX_3D Function Module

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RWNID **</td>
<td>Export</td>
</tr>
<tr>
<td>DATA</td>
<td>Internal Table</td>
</tr>
<tr>
<td>OPTS</td>
<td>Internal Table</td>
</tr>
</tbody>
</table>

Note: ** denotes a dialog parameter

Of the above list, only the DATA, OPTS and at least one of the COLn (COL1, COL2,..., COL6) parameters are required.
The GRAPH_MATRIX_4D Function Module

The GRAPH_MATRIX_4D function module calls SAP Business Graphics with "four-dimensional" data for display in a graph. This four-dimensional graph is actually a series of three-dimensional graphs you can page through on the screen. Each 3D graph is identified by page, and the elements in it are identified by row and column.

The chief parameter (DATA) is a table of values, and further parameters tell which part of the table to display. You can use other parameters to set display options.

The following table contains a list of all possible parameters you can use with this function module.

For a detailed description of parameters and exceptions, see the R/3 system documentation (Tools → ABAP Workbench, Function library button, or transaction code SE37).

### Parameters used with GRAPH_MATRIX_4D

<table>
<thead>
<tr>
<th>Name</th>
<th>Status</th>
<th>Initial Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Status</td>
<td>Initial Value</td>
</tr>
<tr>
<td>DIM1,DIM2,DIM3</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>DIM1_1...DIM1_6</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>DIM2_1...DIM2_6</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>INBUF **</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>INFORM</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>MAIL_ALLOW</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>PWDID **</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>STAT **</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>SUPER **</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>TIMER</td>
<td>Import</td>
<td>(Do Not Use)</td>
</tr>
<tr>
<td>TITL</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>VALT</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>WDID</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>WINID **</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>WINPOS</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>WINSZX</td>
<td>Import</td>
<td>'50'</td>
</tr>
<tr>
<td>WINSZY</td>
<td>Import</td>
<td>'50'</td>
</tr>
<tr>
<td>X_OPT</td>
<td>Import</td>
<td>(Do Not Use)</td>
</tr>
<tr>
<td>B_KEY</td>
<td>Export</td>
<td></td>
</tr>
<tr>
<td>B_TYP</td>
<td>Export</td>
<td></td>
</tr>
</tbody>
</table>
### The GRAPH_MATRIX_4D Function Module

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>M_TYP **</td>
<td>Export</td>
</tr>
<tr>
<td>MOD_COL</td>
<td>Export</td>
</tr>
<tr>
<td>MOD_ROW</td>
<td>Export</td>
</tr>
<tr>
<td>MOD_VAL</td>
<td>Export</td>
</tr>
<tr>
<td>RBUFF **</td>
<td>Export</td>
</tr>
<tr>
<td>RWNID **</td>
<td>Export</td>
</tr>
<tr>
<td>DATA</td>
<td>Internal Table</td>
</tr>
<tr>
<td>OPTS</td>
<td>Internal Table</td>
</tr>
</tbody>
</table>

Note: ** denotes a dialog parameter

Of the above list, the DATA, OPTS, at least one of the DIM1_1...DIM1_6 parameters are required and at least one of the DIM2_1...DIM2_6 parameters are required.
The GRAPH_MATRIX Function Module

The GRAPH_MATRIX function module is the most general-purpose of the SAP Business Graphics function modules.

With this module, you can send data in the same "four-dimensional" format described for GRAPH_MATRIX_4D. A graph of four-dimensional data is actually a series of three-dimensional graphs you can page through on the screen. Each 3D graph is identified by page, and the elements in it are identified by row and column.

The chief parameter (DATA) is a table of values, and further parameters tell which part of the table to display. You can use other parameters to set display options.

The following table contains a list of all possible parameters you can use with this function module.

For a detailed description of parameters and exceptions, see the R/3 system documentation (Tools → ABAP Workbench, Function library button, or transaction code SE37).

Parameters used with GRAPH_MATRIX

<table>
<thead>
<tr>
<th>Name</th>
<th>Status</th>
<th>Initial Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIM1,DIM2,DIM3</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>INBUF **</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>INFORM</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>MAIL_ALLOW</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>MAX1,MAX2,MAX3</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>PWIDID **</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>SET_FOCUS</td>
<td>Import</td>
<td>Do Not Use</td>
</tr>
<tr>
<td>STAT **</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>SUPER **</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>TIMER</td>
<td>Import</td>
<td>(Do Not Use)</td>
</tr>
<tr>
<td>TITL</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>VALT</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>WDID</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>WINID **</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>WINPOS</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>WINSZX</td>
<td>Import</td>
<td>'50'</td>
</tr>
<tr>
<td>WINSZY</td>
<td>Import</td>
<td>'50'</td>
</tr>
<tr>
<td>X_OPT</td>
<td>Import</td>
<td>(Do Not Use)</td>
</tr>
<tr>
<td>B_KEY</td>
<td>Export</td>
<td>(Do Not Use)</td>
</tr>
<tr>
<td>B_TYP</td>
<td>Export</td>
<td>(Do Not Use)</td>
</tr>
</tbody>
</table>
The GRAPH_MATRIX Function Module

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>M_TYP **</td>
<td>Export</td>
</tr>
<tr>
<td>MOD_COL</td>
<td>Export</td>
</tr>
<tr>
<td>MOD_ROW</td>
<td>Export</td>
</tr>
<tr>
<td>MOD_VAL</td>
<td>Export</td>
</tr>
<tr>
<td>RBUFF **</td>
<td>Export</td>
</tr>
<tr>
<td>RWNID **</td>
<td>Export</td>
</tr>
<tr>
<td>DATA</td>
<td>Internal Table</td>
</tr>
<tr>
<td>OPTS</td>
<td>Internal Table</td>
</tr>
<tr>
<td>TDIM1,TDIM2,TDIM3</td>
<td>Internal Table</td>
</tr>
</tbody>
</table>

Note: ** denotes a dialog parameter

Of the above list, the DATA, OPTS, TDIM1, TDIM2, and TDIM3 parameters are required.
The GRAPH_BUSG_MENU_SET Function Module

The GRAPH_BUSG_MENU_SET function module is a special-purpose routine that lets you define a graphics toolbox. A toolbox is a row of buttons that appears in the business graphics window with specified labels. The user can click on a toolbox button, and SAP Business Graphics reports the selection back to the ABAP program.

In SAP Business Graphics, the utility function modules GRAPH_SET_CUA_STATUS or GRAPH_ACTION are not available for defining pushbuttons.

An ABAP program must call GRAPH_BUSG_MENU_SET before calling the function module for which the toolbox is to be defined. GRAPH_BUSG_MENU_SET itself does not open a graphics window; it merely records the toolbox information in static storage. The next SAP Business Graphics function module you call (that opens a window or triggers a screen update) will find this information and display toolbox buttons in the window.

You can only use GRAPH_BUSG_MENU_SET to place buttons in a single window. If you want in other BUSG windows (existing or not yet opened), you must call GRAPH_BUSG_MENU_SET again.

To call GRAPH_BUSG_MENU_SET, your program provides a single table parameter containing labels for the toolbox buttons. If the online user clicks on one of these buttons, the function module reports this to the ABAP program in the B_TYP and B_KEY parameters. (B_TYP='C' and B_KEY contains the index (in MENU_TAB) for the button pressed.)

You cannot use GRAPH_BUSG_MENU_SET to wait for input: the STAT parameter is not allowed with this function module.

The following table contains a list of all possible parameters you can use with this function module.

For a detailed description of parameters and exceptions, see the R/3 system documentation (Tools → ABAP Workbench, Function library button, or transaction code SE37).

Parameters used with GRAPH_MATRIX

<table>
<thead>
<tr>
<th>Name</th>
<th>Status</th>
<th>Initial Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>MENU_TAB</td>
<td>Internal Table</td>
<td></td>
</tr>
</tbody>
</table>
The GET_BUSG_PARAM Function Module

The GET_BUSG_PARAM function module calls SAP Business Graphics to get export parameters set to their proper values. Use GET_BUSG_PARAM when the ABAP program has received return values of MCODE='I' and RWNID='BUSGrn'. See Controlling the Interaction Using Parameters and Utility Function Modules [Page 109] for more information.

The following figure contains a list of the parameters you can use with this function module.

For a detailed description of parameters and exceptions, see the R/3 system documentation (Tools → ABAP Workbench, Function library button, or transaction code SE37).

Parameters used with GET_BUSG_PARAM

<table>
<thead>
<tr>
<th>Name</th>
<th>Status</th>
<th>Initial Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>B_KEY</td>
<td>Export</td>
<td></td>
</tr>
<tr>
<td>B_TYP</td>
<td>Export</td>
<td></td>
</tr>
<tr>
<td>CUA_ID</td>
<td>Export</td>
<td></td>
</tr>
<tr>
<td>MOD_COL</td>
<td>Export</td>
<td></td>
</tr>
<tr>
<td>MOD_ROW</td>
<td>Export</td>
<td></td>
</tr>
<tr>
<td>MOD_VAL</td>
<td>Export</td>
<td></td>
</tr>
</tbody>
</table>
Sample Programs

The following ABAP programs contain function modules that call up SAP Business Graphics. For further sample programs calling SAP Graphics applications, use transaction code gral in the R/3 System.

- **Calling SAP Business Graphics, Without Dialog** [Page 115]
- **Calling SAP Pushbutton, Statistical and Business Graphics, With Dialog** [Page 117]
- **Calling SAP Pushbutton and Business Graphics, With Dialog, Using GRAPH_RECEIVE and GET_xxxx_PARAM** [Page 129]
Calling SAP Hierarchy Graphics

An ABAP program specifies hierarchy data and calls up an appropriate function module. The SAP Hierarchy Graphics program generates a graphical hierarchy of nodes from the data supplied.

Overview [Page 30]

The GRAPH_HIERARCHY Function Module [Page 32]

The GRAPH_HIERARCHY_MENUE Function Module [Page 34]

The GET_HIER_PARAM Function Module [Page 37]

Sample Program [Page 138]
Overview

The SAP Hierarchy graphics program generates a graphical hierarchy of nodes that online users can click on. A typical hierarchy looks as shown in the following figure.

You can call SAP Hierarchy from an ABAP program, and generate such a hierarchy at the user's workstation screen. Your program specifies the nodes of the hierarchy and calls one of the following function modules:

- **GRAPH_HIERARCHY**: presents a simple hierarchy.
  
  Your ABAP program responds to user node selection with particular actions.

- **GRAPH_HIERARCHY_MENUE**: presents a hierarchy with menu capabilities.
  
  The function module responds to user node selection by presenting a menu of functions. Your ABAP program must tell the function module what menu to display, and respond when menu items have been selected.

Your program can also determine the appearance (text and color of individual nodes) of the hierarchy. Nodes can also be assigned numeric values and colored according to the size of these values.
Programming Interactive Applications

If your ABAP program allows the online user to open multiple windows or enter data with the mouse or keyboard, the application is interactive.

You are provided with two mechanisms for controlling the interaction between ABAP program and online user:

- **Dialog programming in Releases prior to 2.1**
  Using graphics function modules with dialog parameters STAT, M_TYP, PWDID, SUPER, WINID and RWNID)

- **Current dialog programming**
  Using graphics function modules with dialog parameters STAT, PWDID, SUPER, and WINID)
  
  plus utility function modules :
  - GRAPH_RECEIVE (with dialog parameters ACTIVATE_RAISE, ERRORCODE, MCODE, and RWNID)
  - GET_xxxx_PARAM

For new programs, you should use only the second mechanism, graphics function modules with the utility function modules.

💡 Do not use both mechanisms together in a program.

For more information, see the section Programming Interactive Applications [Page 93].

You can also change the standard graphics display and define additional graphics functions for interactive users. This means when you call up SAP Hierarchy, it appears with changed or extra menu names or pushbuttons. For more information, see the section Changing the Standard Graphics Display [Page 86]. To define additional functions, call the appropriate function module before you call an SAP Hierarchy function module.

If your ABAP program calls GRAPH_ACTION, it must also use the CUA_ID and H_MES parameters (when calling SAP Hierarchy function modules) in order to learn when the user has selected these additional functions.
The GRAPH_HIERARCHY Function Module

The GRAPH_HIERARCHY function module executes a call to the SAP Hierarchy graphics program.

The following table contains a list of all possible parameters you can use with this function module.

For a detailed description of parameters and exceptions, see the R/3 system documentation (Tools → ABAP Workbench, Function library button, or transaction code SE37).

### Parameters specific to GRAPH_HIERARCHY

<table>
<thead>
<tr>
<th>Name</th>
<th>Status</th>
<th>Initial Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOXLN</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>COL11...COL16</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>COL21...COL26</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>COL31...COL36</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>COL_BACK</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>COL_LINE</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>COL_NODE</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>CONF</td>
<td>Import</td>
<td>Do Not Use</td>
</tr>
<tr>
<td>DMODE</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>INBUF **</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>LMODE</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>MAIL_ALLOW</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>MODIF</td>
<td>Import</td>
<td>Do Not Use</td>
</tr>
<tr>
<td>NDIMS</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>NLOOK</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>NLVAL</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>NTEXT</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>NTXHXL</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>ORDER</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>PWID **</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>RELBOXHT</td>
<td>Import</td>
<td>0</td>
</tr>
<tr>
<td>RELBOXWD</td>
<td>Import</td>
<td>0</td>
</tr>
<tr>
<td>RELWD_HT</td>
<td>Import</td>
<td>0</td>
</tr>
<tr>
<td>RMODE</td>
<td>Import</td>
<td>SPACE</td>
</tr>
</tbody>
</table>
### The GRAPH_HIERARCHY Function Module

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Direction</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>SHAPE</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>SMODE</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>STAT **</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>SUPER **</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>TINDEX</td>
<td>Import</td>
<td>'0'</td>
</tr>
<tr>
<td>TTEXT</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>VALD1...VALD3</td>
<td>Import</td>
<td>0</td>
</tr>
<tr>
<td>VALU1...VALU3</td>
<td>Import</td>
<td>100</td>
</tr>
<tr>
<td>VAL11...VAL15</td>
<td>Import</td>
<td>0</td>
</tr>
<tr>
<td>VAL21...VAL25</td>
<td>Import</td>
<td>0</td>
</tr>
<tr>
<td>VAL31...VAL35</td>
<td>Import</td>
<td>0</td>
</tr>
<tr>
<td>VINDEX</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>VTXT1...VTXT3</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>WINID **</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>WINPOS</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>WINSZX</td>
<td>Import</td>
<td>'50'</td>
</tr>
<tr>
<td>WINSZY</td>
<td>Import</td>
<td>'50'</td>
</tr>
<tr>
<td>ZOOMFAC</td>
<td>Import</td>
<td>0</td>
</tr>
<tr>
<td>CUA_ID</td>
<td>Export</td>
<td></td>
</tr>
<tr>
<td>H_MES</td>
<td>Export</td>
<td></td>
</tr>
<tr>
<td>M_TYP **</td>
<td>Export</td>
<td></td>
</tr>
<tr>
<td>NODES</td>
<td>Export</td>
<td></td>
</tr>
<tr>
<td>RBUFF **</td>
<td>Export</td>
<td></td>
</tr>
<tr>
<td>RWNID **</td>
<td>Export</td>
<td></td>
</tr>
<tr>
<td>DATA</td>
<td>Internal Table</td>
<td></td>
</tr>
</tbody>
</table>

Note: ** denotes a dialog parameter

For GRAPH_HIERARCHY, only the DATA parameter is required.
The GRAPH_HIERARCHY_MENUE Function Module

The GRAPH_HIERARCHY_MENUE function module executes a call to the SAP Hierarchy graphics program. Unlike the GRAPH_HIERARCHY function module, this module allows you to program hierarchy displays with menus. The user who clicks on a node gets a menu of options.

The following table contains a list of all possible parameters you can use with this function module.

For a detailed description of parameters and exceptions, see the R/3 system documentation (Tools → ABAP Workbench, Function library button, or transaction code SE37).

Parameters specific to GRAPH_HIERARCHY_MENUE

<table>
<thead>
<tr>
<th>Name</th>
<th>Status</th>
<th>Initial Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOXLN</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>COL11...COL16</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>COL21...COL26</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>COL31...COL36</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>COL_BACK</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>COL_LINE</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>COL_NODE</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>CONF</td>
<td>Import</td>
<td>Do Not Use</td>
</tr>
<tr>
<td>DMODE</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>INBUF **</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>LMODE</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>MAIL_ALLOW</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>MENPOS</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>MODIF</td>
<td>Import</td>
<td>Do Not Use</td>
</tr>
<tr>
<td>MTITL</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>NDIMS</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>NLOOK</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>NLVAL</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>NTEXT</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>NTXHL</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>ORDER</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>PWDID **</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>REGION</td>
<td>Import</td>
<td>Do Not Use</td>
</tr>
<tr>
<td>RELBOXHT</td>
<td>Import</td>
<td>0</td>
</tr>
</tbody>
</table>
The GRAPH_HIERARCHY_MENUE Function Module

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>RELBOXWD</td>
<td>Import</td>
<td>0</td>
</tr>
<tr>
<td>RELWD_HT</td>
<td>Import</td>
<td>0</td>
</tr>
<tr>
<td>RMODE</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>SHAPE</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>SMODE</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>STAT **</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>SUPER **</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>TINDEX</td>
<td>Import</td>
<td>'0'</td>
</tr>
<tr>
<td>TTTEXT</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>USEHLE</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>VALD1...VALD3</td>
<td>Import</td>
<td>0</td>
</tr>
<tr>
<td>VALU1...VALU3</td>
<td>Import</td>
<td>100</td>
</tr>
<tr>
<td>VAL11...VAL15</td>
<td>Import</td>
<td>0</td>
</tr>
<tr>
<td>VAL21...VAL25</td>
<td>Import</td>
<td>0</td>
</tr>
<tr>
<td>VAL31...VAL35</td>
<td>Import</td>
<td>0</td>
</tr>
<tr>
<td>VINDEX</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>VTXT1...VTXT3</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>WINID **</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>WINPOS</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>WINSZX</td>
<td>Import</td>
<td>'50'</td>
</tr>
<tr>
<td>WINSZY</td>
<td>Import</td>
<td>'50'</td>
</tr>
<tr>
<td>ZOOMFAC</td>
<td>Import</td>
<td>0</td>
</tr>
<tr>
<td>CUA_ID</td>
<td>Export</td>
<td></td>
</tr>
<tr>
<td>H_MES</td>
<td>Export</td>
<td></td>
</tr>
<tr>
<td>M_TYP **</td>
<td>Export</td>
<td></td>
</tr>
<tr>
<td>MENNR</td>
<td>Export</td>
<td></td>
</tr>
<tr>
<td>NODES</td>
<td>Export</td>
<td></td>
</tr>
<tr>
<td>RBUFF **</td>
<td>Export</td>
<td></td>
</tr>
<tr>
<td>RWNID **</td>
<td>Export</td>
<td></td>
</tr>
<tr>
<td>DATA</td>
<td>Internal Table</td>
<td></td>
</tr>
<tr>
<td>MENU</td>
<td>Internal Table</td>
<td></td>
</tr>
</tbody>
</table>

Note: ** denotes a dialog parameter

For GRAPH_HIERARCHY_MENUE, only the DATA and MENU parameters are required.
The GRAPH_HIERARCHY_MENUE Function Module

An ABAP program that calls this function module must be sure to use the STAT parameter. At least one of the calls to a graphics function module (not necessarily to GRAPH_HIERARCHY_MENUE) must be made with a Wait-for-input STAT value.

After the hierarchy is displayed, the online user can either close the window or select a node. If a node is selected, the SAP Hierarchy displays the requested menu. If the user then selects a menu entry, SAP Hierarchy places a message reporting user input on the message queue. The waiting function module receives this message and returns to the ABAP program with the number of the entry. (See the descriptions of the MENU and MENNR parameters for more information.)
The GET_HIER_PARAM Function Module

The GET_HIER_PARAM function module calls SAP Hierarchy to get export parameters set to their proper values. Use GET_HIER_PARAM when the ABAP program has received return values of MCODE='I' and RWNID='HIERn'. See Controlling the Interaction Using Parameters and Utility Function Modules [Page 109].

The following table contains a list of the parameters you can use with this function module.

For a detailed description of parameters and exceptions, see the R/3 system documentation (Tools → ABAP Workbench, Function library button, or transaction code SE37).

Parameters used with GET_HIER_PARAM

<table>
<thead>
<tr>
<th>Name</th>
<th>Status</th>
<th>Initial Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CUA_ID</td>
<td>Export</td>
<td></td>
</tr>
<tr>
<td>H_MES</td>
<td>Export</td>
<td></td>
</tr>
<tr>
<td>H_NODE_2</td>
<td>Export</td>
<td></td>
</tr>
<tr>
<td>H_TEXT</td>
<td>Export</td>
<td></td>
</tr>
<tr>
<td>H_VAL1</td>
<td>Export</td>
<td></td>
</tr>
<tr>
<td>H_VAL2</td>
<td>Export</td>
<td></td>
</tr>
<tr>
<td>H_VAL3</td>
<td>Export</td>
<td></td>
</tr>
<tr>
<td>H_VALCOUNT</td>
<td>Export</td>
<td></td>
</tr>
<tr>
<td>MENNR</td>
<td>Export</td>
<td></td>
</tr>
<tr>
<td>NODES</td>
<td>Export</td>
<td></td>
</tr>
</tbody>
</table>
Calling SAP Pushbutton Graphics

An ABAP program specifies the data and calls up an appropriate function module. The SAP Pushbutton Graphics program generates a pad with pushbuttons providing information or functions.

Overview [Page 39]

The GRAPH_BUTTON_MATRIX Function Module [Page 41]

The GRAPH_BUTTON_MENUE Function Module [Page 43]

The GET_BMAT_PARAM Function Module [Page 45]

Sample Programs [Page 46]
Overview

The SAP Pushbutton Graphics program generates a pushbutton pad that the online users can click on. A typical pushbutton pad looks as shown in the following figure.

You can call SAP Pushbutton Graphics from an ABAP program, and generate a pushbutton pad at the user's workstation terminal. Your program specifies the buttons on the pad and calls one of the following function modules:

- **GRAPH_BUTTON_MATRIX**: presents a simple pushbutton pad.
  - Your ABAP program responds to user button selection with particular actions.

- **GRAPH_BUTTON_MENUE**: presents a pushbutton pad with menu capabilities.
  - The function module responds to user button selection by presenting a menu of functions. Your ABAP program must tell the function module what menu to display, and respond when menu items have been selected.

Your program can also determine the appearance (text and color of individual buttons) of the pushbutton pad. Buttons can also be assigned numeric values and colored according to the size of these values.

Programming Interactive Applications

An ABAP program that allows the online user to open multiple windows or enter data with the mouse or keyboard is interactive. When you use the function modules in this section, your application is always interactive, since SAP Pushbutton Graphics always reports user-clicks (selections) back to the ABAP program.

You are provided with two mechanisms for controlling the interaction between ABAP program and online user:

- Dialog programming in Releases prior to 2.1
  - Using graphics function modules with dialog parameters STAT, M_TYP, PWDID, SUPER, WINID and RWNID)

- Current dialog programming
Overview

Using graphics function modules with dialog parameters STAT, PWDID, SUPER, and WINID)
  plus utility function modules:
  - GRAPH_RECEIVE (with dialog parameters ACTIVATE_RAISE, ERRORCODE, MCODE, and RWNID)
  - GET_xxxx_PARAM

For new programs, you should use only the second mechanism, graphics function modules with the utility function modules.

Do not use both mechanisms together in a program.

For more information, see Programming Interactive Applications [Page 93]

You can also change the standard graphics display and define additional graphics functions for interactive users. This means when you call up SAP Pushbutton Graphics, it appears with changed or extra menu names or pushbuttons.

For more information, see the section Changing the Standard Graphics Display [Page 86]

To define additional functions, call the appropriate function module before you call the SAP Pushbutton Graphics function module.

If your ABAP program calls GRAPH_ACTION, it must also use the CUA_ID and B_TYP parameters (when calling GRAPH_BUTTON_MATRIX or GRAPH_BUTTON_MENU) in order to learn when the user has selected these additional functions.
The GRAPH_BUTTON_MATRIX Function Module

The GRAPH_BUTTON_MATRIX function module executes a call to the SAP Pushbutton Graphics program.

The following table contains a list of all possible parameters you can use with this function module.

For a detailed description of parameters and exceptions, see the R/3 system documentation (Tools → ABAP Workbench, Function library button, or transaction code SE37).

Parameters used with GRAPH_BUTTON_MATRIX

<table>
<thead>
<tr>
<th>Name</th>
<th>Status</th>
<th>Initial Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>COLLN</td>
<td>Import</td>
<td>15</td>
</tr>
<tr>
<td>COL11...COL16</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>CTXT1...CTXT9</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>INBUF **</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>MODIF</td>
<td>Import</td>
<td>'X'</td>
</tr>
<tr>
<td>OUTFT</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>PWIDID **</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>ROWLN</td>
<td>Import</td>
<td>1</td>
</tr>
<tr>
<td>STAT **</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>STITL</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>SUPER **</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>TTTEXT</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>VALU1</td>
<td>Import</td>
<td>0</td>
</tr>
<tr>
<td>VAL11...VAL15</td>
<td>Import</td>
<td>0</td>
</tr>
<tr>
<td>VTXT1</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>WINID **</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>WINPOS</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>WINSZX</td>
<td>Import</td>
<td>'50'</td>
</tr>
<tr>
<td>WINSZY</td>
<td>Import</td>
<td>50’</td>
</tr>
<tr>
<td>B_TYP</td>
<td>Export</td>
<td></td>
</tr>
<tr>
<td>COLNR</td>
<td>Export</td>
<td></td>
</tr>
<tr>
<td>CUA_ID</td>
<td>Export</td>
<td></td>
</tr>
<tr>
<td>M_TYP **</td>
<td>Export</td>
<td></td>
</tr>
</tbody>
</table>
The `GRAPH_BUTTON_MATRIX` Function Module

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>RBUFF **</td>
<td>Export</td>
</tr>
<tr>
<td>ROWNR</td>
<td>Export</td>
</tr>
<tr>
<td>RWNID **</td>
<td>Export</td>
</tr>
<tr>
<td>ATTR</td>
<td>Internal Table</td>
</tr>
<tr>
<td>DATA</td>
<td>Internal Table</td>
</tr>
<tr>
<td>SIZE</td>
<td>Internal Table</td>
</tr>
</tbody>
</table>

Note: ** denotes a dialog parameter

Of this list, only the DATA, ATTR and at least one of the CTXT1 through CTXT9 parameters is required.
The GRAPH_BUTTON_MENUE Function Module

The GRAPH_BUTTON_MENUE function module executes a call to the SAP Pushbutton Graphics program. Unlike the GRAPH_BUTTON_MATRIX function module, this module allows you to program pushbutton displays with menus. The user who clicks on a button gets a menu of options rather than a static display.

Parameters for GRAPH_BUTTON_MENUE include all those for GRAPH_BUTTON_MATRIX, as well as the MTITL, MCOLN, MROWN, and MENU parameters. (See the following table.)

For a detailed description of parameters and exceptions, see the R/3 system documentation (Tools → ABAP Workbench, Function library button, or transaction code SE37).

Parameters used with GRAPH_BUTTON_MENUE

<table>
<thead>
<tr>
<th>Name</th>
<th>Status</th>
<th>Initial Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>COLLN</td>
<td>Import</td>
<td>15</td>
</tr>
<tr>
<td>COL11...COL16</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>COL21...COL26</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>COL31...COL36</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>CTXT1...CTXT9</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>INBUF **</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>MODIF</td>
<td>Import</td>
<td>'X'</td>
</tr>
<tr>
<td>MTITL</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>OUTFT</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>PWDID **</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>ROWLN</td>
<td>Import</td>
<td>1</td>
</tr>
<tr>
<td>STAT **</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>STITL</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>SUPER **</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>TTEXT</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>VALD1</td>
<td>Import</td>
<td>0</td>
</tr>
<tr>
<td>VALU1</td>
<td>Import</td>
<td>100</td>
</tr>
<tr>
<td>VAL11...VAL15</td>
<td>Import</td>
<td>0</td>
</tr>
<tr>
<td>VTXT1</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>WINID **</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>WINPOS</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>WINSZX</td>
<td>Import</td>
<td>'50'</td>
</tr>
<tr>
<td>WINSZY</td>
<td>Import</td>
<td>'50'</td>
</tr>
</tbody>
</table>
The GRAPH_BUTTON_MENUE Function Module

<table>
<thead>
<tr>
<th>B_TYP</th>
<th>Export</th>
</tr>
</thead>
<tbody>
<tr>
<td>COLNR</td>
<td>Export</td>
</tr>
<tr>
<td>CUA_ID</td>
<td>Export</td>
</tr>
<tr>
<td>M_TYP **</td>
<td>Export</td>
</tr>
<tr>
<td>MCOLN</td>
<td>Export</td>
</tr>
<tr>
<td>MROWN</td>
<td>Export</td>
</tr>
<tr>
<td>RBUFF **</td>
<td>Export</td>
</tr>
<tr>
<td>ROWNR</td>
<td>Export</td>
</tr>
<tr>
<td>RWIND **</td>
<td>Export</td>
</tr>
<tr>
<td>ATTR</td>
<td>Internal Table</td>
</tr>
<tr>
<td>DATA</td>
<td>Internal Table</td>
</tr>
<tr>
<td>MENU</td>
<td>Internal Table</td>
</tr>
<tr>
<td>SIZE</td>
<td>Internal Table</td>
</tr>
</tbody>
</table>

Note: ** denotes a dialog parameter

An ABAP program that calls this function module must be sure to use the STAT parameter. At least one of the calls to a graphics function module (not necessarily the one to GRAPH_BUTTON_MENUE) must be made with a Wait-for-input STAT value.

After the pushbutton pad is displayed, the online user can either close the window or click on a graph item (a pushbutton, or, if present, application-defined menu items or function keys). If the user clicks on an button, the SAP Pushbutton Graphics displays the requested menu. If the user then selects a menu entry, SAP Pushbutton Graphics places a message reporting user input on the message queue. The waiting function module receives this message and returns to the ABAP program with the row and column of the menu entry. (See the descriptions of the MENU, MROWN, and MCOLN parameters for more information.)

SAP Pushbutton Graphics responds in a similar manner for application-defined menu items or function keys. (The ABAP program defines these by calling GRAPH_ACTION.) If the user clicks on one of these items, SAP Pushbutton Graphics places a message reporting this on the message queue. The waiting function module receives the message and returns to the ABAP program with information identifying the selected item in the CUA_ID parameter field.
The GET_BMAT_PARAM Function Module

The GET_BMAT_PARAM function module calls SAP Pushbutton Graphics to get export parameters set to their proper values. Use GET_BMAT_PARAM when the ABAP program has received return values of MCODE='I' and RNID='BMATn'. See Controlling the Interaction Using Parameters and Utility Function Modules [Page 109] for more information.

The following table contains a list of the parameters you can use with this function module.

For a detailed description of parameters and exceptions, see the R/3 system documentation (Tools → ABAP Workbench, Function library button, or transaction code SE37).

Parameters used with GET_BMAT_PARAM

<table>
<thead>
<tr>
<th>Name</th>
<th>Status</th>
<th>Initial Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>B_KEY</td>
<td>Export</td>
<td></td>
</tr>
<tr>
<td>B_TYP</td>
<td>Export</td>
<td></td>
</tr>
<tr>
<td>COLNR</td>
<td>Export</td>
<td></td>
</tr>
<tr>
<td>CUA_ID</td>
<td>Export</td>
<td></td>
</tr>
<tr>
<td>MCOLN</td>
<td>Export</td>
<td></td>
</tr>
<tr>
<td>MROWN</td>
<td>Export</td>
<td></td>
</tr>
<tr>
<td>ROWNR</td>
<td>Export</td>
<td></td>
</tr>
</tbody>
</table>
Sample Programs

The following ABAP programs contain function modules that call up SAP Pushbutton Graphics.

For further sample programs calling SAP Graphics applications, use transaction code gral in the R/3 System.

- Calling SAP Pushbutton, Statistical and Business Graphics, With Dialog [Page 117]
- Calling SAP Pushbutton and Business Graphics, With Dialog, Using GRAPH_RECEIVE and GET_xxxx_PARAM [Page 129]
Calling SAP Gantt Chart

An ABAP program specifies planning data and calls up an appropriate function module. The SAP Gantt Chart program generates a planning board at the user's workstation terminal.

Overview [Page 48]

The GANTT_DIAGRAMM Function Module [Page 50]

The GRAPH_GANTT Function Module [Page 52]

The GET_GANTT_PARAM Function Module [Page 54]

Sample Program [Page 169]
Overview

SAP Gantt Chart generates a planning board on which scheduling information is displayed. The online user can browse the planning-board in a variety of formats, or modify it to re-schedule events.

You can call SAP Gantt Chart from an ABAP program, and generate a planning board at the user's workstation terminal. Your program provides scheduling information and calls one of the SAP Gantt Chart function modules:

- GRAPH_GANTT
- GANTT_DIAGRAMM

NOTE: GANTT_DIAGRAMM is an older function module that should not be used for new developments.

The SAP Gantt Chart planning board looks as shown in the following figure.

![SAP Gantt Chart Planning Board](image)

Programming Interactive Applications

If your ABAP program allows the online user to open multiple windows or enter data with the mouse or keyboard, the application is interactive.

You are provided with two mechanisms for controlling the interaction between ABAP program and online user:

- Dialog programming in Releases prior to 2.1
  Using graphics function modules with dialog parameters STAT, M_TYP, PWID, SUPER, WINID and RWID)
- Current dialog programming
Using graphics function modules with dialog parameters STAT, PWDID, SUPER, and WINID)

plus utility function modules:

- GRAPH_RECEIVE (with dialog parameters ACTIVATE_RAISE, ERRORCODE, MCODE, and RWNID)
- GET_xxxx_PARAM

For new programs, you should use only the second mechanism, graphics function modules with the utility function modules.

Do not use both mechanisms together in a program.

For more information, see the section Programming Interactive Applications [Page 93].

You can also change the standard graphics display and define additional graphics functions for interactive users using GRAPH_SET_CUA_STATUS or GRAPH_GANTT. This means when you call up SAP Gantt Chart, it appears with changed or extra menu names or pushbuttons.

For more information, see the section Changing the Standard Graphics Display [Page 86]. To define additional functions, call the appropriate function module before you call an SAP Gantt Chart function module.

If your ABAP program calls GRAPH_ACTION, it must also use the CUA_ID and G_TYP parameters (when calling GRAPH_GANTT) in order to learn when the user has selected these additional functions.
The GANTT_DIAGRAMM Function Module

GANTT_DIAGRAMM is an older function module that should not be used for new development. Use GRAPH_GANTT instead.

The GANTT_DIAGRAMM function module executes a call to the SAP Gantt Chart graphics program.

The following table contains a list of all possible parameters you can use with this function module.

For a detailed description of parameters and exceptions, see the R/3 system documentation (Tools → ABAP Workbench, Function library button, or transaction code SE37).

Parameters used with GANTT_DIAGRAMM

<table>
<thead>
<tr>
<th>Name</th>
<th>Status</th>
<th>Initial Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>COLOR</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>MODIF</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>PWDID **</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>STAT **</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>SUPER **</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>TLENGTH</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>TTEXT</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>TTITLE</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>TUNIT</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>WHEADER</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>WIDTH</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>WINID **</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>M_TYP **</td>
<td>Export</td>
<td></td>
</tr>
<tr>
<td>RBUFF **</td>
<td>Export</td>
<td></td>
</tr>
<tr>
<td>RWNID **</td>
<td>Export</td>
<td></td>
</tr>
<tr>
<td>ELEM</td>
<td>Internal Table and Export</td>
<td></td>
</tr>
<tr>
<td>ITEM</td>
<td>Internal Table</td>
<td></td>
</tr>
</tbody>
</table>

Note: ** denotes a dialog parameter.

Of the parameters listed above, only ELEM and ITEM are required.
The GANTT_DIAGRAMM Function Module

The function module GANTT_DIAGRAMM differs from other SAP Graphics function modules in that it allows modification of an internal table. This table, ELEM, contains planning board information: information in ELEM is provided by the ABAP program, but can be modified by the online user.
The GRAPH_GANTT Function Module

The GRAPH_GANTT function module executes a call to the SAP Gantt Chart graphics program. The following table contains a list of all possible parameters you can use with this function module.

For a detailed description of parameters and exceptions, see the R/3 system documentation (Tools → ABAP Workbench, Function library button, or transaction code SE37).

Parameters used with GRAPH_GANTT

<table>
<thead>
<tr>
<th>Name</th>
<th>Status</th>
<th>Ref.Field</th>
<th>Initial Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMMAND_OK</td>
<td>IMPORT</td>
<td>SPACE</td>
<td>SPACE</td>
</tr>
<tr>
<td>CONFIRM</td>
<td>IMPORT</td>
<td>SPACE</td>
<td>SPACE</td>
</tr>
<tr>
<td>DFEL</td>
<td>IMPORT</td>
<td>GGAEL</td>
<td>SPACE</td>
</tr>
<tr>
<td>DFMS</td>
<td>IMPORT</td>
<td>GGAMI</td>
<td>SPACE</td>
</tr>
<tr>
<td>GLENGTH</td>
<td>IMPORT</td>
<td>SPACE</td>
<td>SPACE</td>
</tr>
<tr>
<td>HGRID</td>
<td>IMPORT</td>
<td>SPACE</td>
<td>SPACE</td>
</tr>
<tr>
<td>LEGEND</td>
<td>IMPORT</td>
<td>SPACE</td>
<td>SPACE</td>
</tr>
<tr>
<td>MODIF</td>
<td>IMPORT</td>
<td>SPACE</td>
<td>SPACE</td>
</tr>
<tr>
<td>NOTXT</td>
<td>IMPORT</td>
<td>SPACE</td>
<td>SPACE</td>
</tr>
<tr>
<td>NO_EL_POPUP</td>
<td>IMPORT</td>
<td>SPACE</td>
<td>SPACE</td>
</tr>
<tr>
<td>NO_EX_POPUP</td>
<td>IMPORT</td>
<td>SPACE</td>
<td>SPACE</td>
</tr>
<tr>
<td>NO_IT_POPUP</td>
<td>IMPORT</td>
<td>SPACE</td>
<td>SPACE</td>
</tr>
<tr>
<td>NO_MS_POPUP</td>
<td>IMPORT</td>
<td>SPACE</td>
<td>SPACE</td>
</tr>
<tr>
<td>NO_SORT</td>
<td>IMPORT</td>
<td>SPACE</td>
<td>SPACE</td>
</tr>
<tr>
<td>PWDID **</td>
<td>IMPORT</td>
<td>SPACE</td>
<td>SPACE</td>
</tr>
<tr>
<td>SET_FOCUS</td>
<td>IMPORT</td>
<td>'X'</td>
<td></td>
</tr>
<tr>
<td>STAT **</td>
<td>IMPORT</td>
<td>SPACE</td>
<td>SPACE</td>
</tr>
<tr>
<td>SUPER **</td>
<td>IMPORT</td>
<td>SPACE</td>
<td>SPACE</td>
</tr>
<tr>
<td>TIMER</td>
<td>IMPORT</td>
<td>GGAEL-BEG</td>
<td>Do Not Use</td>
</tr>
<tr>
<td>TLENGTH</td>
<td>IMPORT</td>
<td>SPACE</td>
<td></td>
</tr>
<tr>
<td>TLINE</td>
<td>IMPORT</td>
<td>GGAEL-BEG</td>
<td>SPACE</td>
</tr>
<tr>
<td>TMBEG</td>
<td>IMPORT</td>
<td>GGAEL-BEG</td>
<td>SPACE</td>
</tr>
<tr>
<td>TMEND</td>
<td>IMPORT</td>
<td>GGAEL-BEG</td>
<td>SPACE</td>
</tr>
<tr>
<td>TMVIEW</td>
<td>IMPORT</td>
<td>GGAEL-BEG</td>
<td>SPACE</td>
</tr>
<tr>
<td>TTEXT</td>
<td>IMPORT</td>
<td>SPACE</td>
<td></td>
</tr>
</tbody>
</table>
Of the parameters listed above, only the table parameters are required, although they may be empty.
The GET_GANTT_PARAM Function Module

The GET_GANTT_PARAM function module calls SAP Gantt Chart to get export parameters set to their proper values. Use GET_GANTT_PARAM when the ABAP program has received return values of MCODE='I' and RWNID='GANTn'. See Controlling the Interaction Using Parameters and Utility Function Modules [Page 109] for more information.

The following table contains a list of the parameters you can use with this function module.

For a detailed description of parameters and exceptions, see the R/3 system documentation (Tools → ABAP Workbench, Function library button, or transaction code SE37).

Parameters used with GET_GANTT_PARAM

<table>
<thead>
<tr>
<th>Name</th>
<th>Status</th>
<th>Ref.Field</th>
<th>Initial Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>AKT_SVIEW</td>
<td>EXPORT</td>
<td>GGAEL-BEG</td>
<td></td>
</tr>
<tr>
<td>AKT_TMBEG</td>
<td>EXPORT</td>
<td>GGAEL-BEG</td>
<td></td>
</tr>
<tr>
<td>AKT_TMEND</td>
<td>EXPORT</td>
<td>GGAEL-BEG</td>
<td></td>
</tr>
<tr>
<td>AKT_TUNIT</td>
<td>EXPORT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CUA_ID</td>
<td>EXPORT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>G_TYP</td>
<td>EXPORT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MSGT</td>
<td>Table</td>
<td>GGABA</td>
<td></td>
</tr>
</tbody>
</table>

54 April 2001
Calling SAP HPGL Display

An ABAP program specifies vector graphic data and calls up an appropriate function module. The SAP HPGL Display program generates a HPGL display at the user's workstation terminal.

Overview [Page 56]

The GRAPH_HPGl Function Module [Page 58]

The GET_HPGl_PARAM Function Module [Page 59]

Sample Program [Page 189]
Overview

The SAP HPGL Display program displays vector graphic data coded in HPGL (Hewlett Packard Graphic Language) format. A sample SAP HPGL Display graphic is shown in the following figure.

You can call SAP HPGL Display from an ABAP program, and generate a HPGL display at the user's workstation terminal. Your program provides the vector graphic information (control codes in HPGL format) to the following function module:

- GRAPH_HPGL: displays a vector graphic.

Programming Interactive Applications

If your ABAP program allows the online user to open multiple windows or enter data with the mouse or keyboard, the application is interactive.

You are provided with two mechanisms for controlling the interaction between ABAP program and online user:

- Dialog programming in Releases prior to 2.1
  Using graphics function modules with dialog parameters STAT, M_TYP, PWDID, SUPER, WINID and RWNID)

- Current dialog programming
  Using graphics function modules with dialog parameters STAT, PWDID, SUPER, and WINID)
plus utility function modules:

- GRAPH_RECEIVE (with dialog parameters ACTIVATE_RAISE, ERRORCODE, MCODE, and RWNID)
- GET_xxxx_PARAM

For new programs, you should use only the second mechanism, graphics function modules with the utility function modules.

Do not use both mechanisms together in a program.

For more information, see Programming Interactive Applications [Page 93].

You can also change the standard graphics display and define additional graphics functions for interactive users. This means when you call up SAP HPGL Display, it appears with changed or extra menu names or pushbuttons.

For more information, see the section Changing the Standard Graphics Display [Page 86]. To define additional functions, call the appropriate function module before you call GRAPH_HPGL.

If your ABAP program calls GRAPH_ACTION, it must also use the CUA_ID and H_TYP parameters (when calling GRAPH_HPGL) in order to learn when the user has selected these additional functions.
The GRAPH_HPGL Function Module

The GRAPH_HPGL function module executes a call to the SAP HPLG Display program.

The following table contains a list of all possible parameters you can use with this function module.

For a detailed description of parameters and exceptions, see the R/3 system documentation (Tools → ABAP Workbench, Function library button, or transaction code SE37).

Parameters used in GRAPH_HPGL

<table>
<thead>
<tr>
<th>Name</th>
<th>Status</th>
<th>Initial Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>FILE</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>INBUF **</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>INFORM</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>PWVID **</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>STAT **</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>SUPER **</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>WINID **</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>WINPOS</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>WINSZXX</td>
<td>Import</td>
<td>'50'</td>
</tr>
<tr>
<td>WINSZYY</td>
<td>Import</td>
<td>'50'</td>
</tr>
<tr>
<td>CUA_ID</td>
<td>Export</td>
<td></td>
</tr>
<tr>
<td>H_KEY</td>
<td>Export</td>
<td></td>
</tr>
<tr>
<td>H_TYP</td>
<td>Export</td>
<td></td>
</tr>
<tr>
<td>M_TYP **</td>
<td>Export</td>
<td></td>
</tr>
<tr>
<td>RBU **</td>
<td>Export</td>
<td></td>
</tr>
<tr>
<td>RWNID **</td>
<td>Export</td>
<td></td>
</tr>
<tr>
<td>DATA</td>
<td>Export</td>
<td></td>
</tr>
</tbody>
</table>

Note: ** denotes a dialog parameter

Of the parameters listed above, only the DATA parameter is required. DATA is required even if you use the FILE parameter.
The GET_HPGL_PARAM Function Module

The GET_HPGL_PARAM function module calls SAP HPGL Display to get export parameters set to their proper values. Use GET_HPGL_PARAM when the ABAP program has received return values of MCODE='I' and RWNID='HPGLn'. See Controlling the Interaction Using Parameters and Utility Function Modules [Page 109] for more information.

The following table contains a list of the parameters you can use with this function module.

For a detailed description of parameters and exceptions, see the R/3 system documentation (Tools → ABAP Workbench, Function library button, or transaction code SE37).

Parameters used with GET_HPGL_PARAM

<table>
<thead>
<tr>
<th>Name</th>
<th>Status</th>
<th>Initial Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CUA_ID</td>
<td>Export</td>
<td></td>
</tr>
<tr>
<td>H_KEY</td>
<td>Export</td>
<td></td>
</tr>
<tr>
<td>H_TYP</td>
<td>Export</td>
<td></td>
</tr>
</tbody>
</table>
Calling SAP Text Matrix

An ABAP program specifies text and calls up an appropriate function module. The SAP Text Matrix program generates a text window at the user's workstation terminal.

Overview [Page 61]

The TEXT_MATRIX Function Module [Page 63]
Overview

The SAP Text Matrix graphics program generates a text window in which the user can scroll (vertically and horizontally). This program allows the programmer to offer text displays (such as help windows) to the online user.

A typical text window looks as shown in the following figure.

You can call SAP Text Matrix from an ABAP program, and generate a text window at the user’s workstation screen. Your program specifies the text to be displayed and calls the following function module:

- TEXT_MATRIX: presents a simple text window

The TEXT_MATRIX function module actually implements text windows by a call to SAP Pushbutton Graphics. In contrast to pushbutton pads, however, text windows offer the online user no functionality other than scrolling.

Your program can also determine the size of the display window.

Programming Interactive Applications

If your ABAP program allows the online user to open multiple windows or enter data with the mouse or keyboard, the application is interactive.

You are provided with two mechanisms for controlling the interaction between ABAP program and online user:

- Dialog programming in Releases prior to 2.1
  Using graphics function modules with dialog parameters STAT, M_TYP, PWDID, SUPER, WINID and RWNID)
Overview

- Current dialog programming
  Using graphics function modules with dialog parameters STAT, PWDID, SUPER, and WINID
  plus utility function modules:
  - GRAPH_RECEIVE (with dialog parameters ACTIVATE_RAISE, ERRORCODE, MCODE, and RWNID)
  - GET_xxxx_PARAM

For new programs, you should use only the second mechanism, graphics function modules with the utility function modules.

💡
Do not use both mechanisms together in a program.

For more information, see Programming Interactive Applications [Page 93].

You can also change the standard graphics display and define additional graphics functions for interactive users. This means when you call up SAP Text Matrix, it appears with changed or extra menu names or pushbuttons.

For more information, see the section Changing the Standard Graphics Display [Page 86]. To define additional functions, call the appropriate function module before you call the SAP Text Matrix function module.

If your ABAP program calls GRAPH_ACTION, it must also use the CUA_ID and B_TYP parameters (when calling TEXT_MATRIX) in order to learn when the user has selected these additional functions.
The TEXT_MATRIX Function Module

The TEXT_MATRIX function module executes a call to the SAP Pushbutton Graphics program. The following table contains a list of all possible parameters you can use with this function module.

For a detailed description of parameters and exceptions, see the R/3 system documentation (Tools → ABAP Workbench, Function library button, or transaction code SE37).

Parameters used in TEXT_MATRIX

<table>
<thead>
<tr>
<th>Name</th>
<th>Status</th>
<th>Initial Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSIZE</td>
<td>Import</td>
<td>‘15’</td>
</tr>
<tr>
<td>INBUF  **</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>PWDID **</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>RSIZE</td>
<td>Import</td>
<td>‘0’</td>
</tr>
<tr>
<td>STAT **</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>SUPER **</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>TTEXT</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>WINID **</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>WINPOS</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>WINSZX</td>
<td>Import</td>
<td>‘50’</td>
</tr>
<tr>
<td>WINSZY</td>
<td>Import</td>
<td>‘50’</td>
</tr>
<tr>
<td>B_TYP **</td>
<td>Export</td>
<td></td>
</tr>
<tr>
<td>CUA_ID</td>
<td>Export</td>
<td></td>
</tr>
<tr>
<td>M_TYP **</td>
<td>Export</td>
<td></td>
</tr>
<tr>
<td>RBUFF **</td>
<td>Export</td>
<td></td>
</tr>
<tr>
<td>RWNNID **</td>
<td>Export</td>
<td></td>
</tr>
<tr>
<td>DATA</td>
<td>Internal Table</td>
<td></td>
</tr>
</tbody>
</table>

Note: ** denotes a dialog parameter

Of the parameters listed above, only the DATA parameter is required.
Calling SAP Network Graphics

An ABAP program specifies node data and calls up an appropriate function module. The SAP Network Graphics program generates a network display from the data supplied.

Overview [Page 65]

Confirm-Mode Processing of User Actions [Page 67]

General and Specific Function Modules [Page 69]

Sample Program [Page 200]
Overview

The SAP Network Graphics function modules let you generate graphs of network or hierarchy structures.

Networks structures contain nodes (with links between the nodes), clusters of nodes, and other objects.

Networks structures have properties (attributes) at all different levels. Nodes, for example, can have colors, fields with text labels, numerical values, and so on. Lines can have colors and fields, while clusters can have colors.

A typical SAP Network Graphics display looks as shown in the following figure.

You can call SAP Network Graphics from an ABAP program, and generate a graphical representation at the user’s workstation terminal.

The following function modules call up SAP Network Graphics on the user’s workstation, display the data there, and report some user actions back to the your ABAP program.

- CNET_GRAPHIC_NETWORK
  This function module is for displaying all networks.
- CNET_GRAPHIC_HIERARCHY
Overview

This function module is specialized for displaying hierarchies. However, you can represent hierarchies with CNET_GRAPHIC_NETWORK, if desired.

- CNET_GRAPHIC_CLUSTER
  This function module calls the Cluster Editor.
  The Cluster Editor contains groups of boxes with frames around them. You can specify text to appear in the boxes. The online user can move boxes within a frame.

You can customize your graphics application using a profile (see parameter PROFILE) to be defined in the R/3 Customizing facility.

Reporting Back User Actions

Each of the above function modules provides interactive capabilities. If your ABAP program allows the online user to open multiple windows or enter data with the mouse or keyboard, the application is interactive. In confirm-firm mode, the graphics program notifies the ABAP program every time the user attempts certain actions.

For more information on interactive applications, see Offering Interactive Capabilities [Page 98].

Changing or Adding CUA Items

You can customize the SAP Network Graphics window by changing or removing existing CUA items (menu names or pushbuttons) or defining additional graphics functions.

To add functions or rename menu names, you must call the appropriate function module. For more information, see the section Changing the Standard Graphics Display [Page 86].

Networks and Hierarchies

Networks are structures of nodes connected by lines. A given node can have any number of predecessor nodes and successor nodes. Predecessor nodes are neighbors linked directly to the given node, from the left side. Successor nodes are neighbors linked directly to the given node, from the right side.

Hierarchy structures are special cases of networks. In a hierarchy, each node can have any number of successor nodes, but only a single number of predecessor nodes. Hierarchies are usually pictured in a top to bottom fashion ("downward branching"), so that the predecessor lies above a given node, and the children lie beneath. With hierarchies, predecessors and successors are often called "parent" and "child" nodes.
Confirm-Mode Processing of User Actions

The ABAP program can call SAP Network Graphics function modules in "confirm-mode". In confirm-firm mode, the graphics program notifies the ABAP program every time the user attempts certain actions.

The ABAP program can then confirm or deny the request. In addition, when confirming, the ABAP program can:

• provide further information for the graphics program
  For example, if the user wants to create a new node, the ABAP program can generate names, perform database look-ups, and pass the additional data back to the graphics program.

  If the user wants to delete three nodes in the network, the graphics program sends a deletion message and a table of items to be deleted. The ABAP program can check authorization, update the table of desired deletions (by removing any items for which deletion is not allowed), and pass it back to the graphics program.

• specify secondary actions that must be performed as a result of the request
  This is important when a user action entails secondary actions not explicitly requested.

• initiate its own queries to the graphics program
  The ABAP program can send queries to the graphics program, which the graphics program can answer.

(Note that this confirm-mode has a more extensive meaning than with SAP Gantt Chart programming. There, confirm-mode means the ABAP program can only permit or deny actions. It cannot itself initiate actions. In addition, in GANTT, the confirm-mode parameter is turned on and off during processing. With SAP Network Graphics, confirm-mode should be either on or off throughout the session. Turning the confirm-mode parameter on and off repeatedly is not recommended.)

In general, when the graphics program sends a message, the function module returns the message type in GR_MES. When the ABAP program sends a message, it should place the message type in ABAP_CMD. In confirm-mode, the ABAP program merely copies GR_MES into ABAP_CMD as part of confirmation. It also updates any message-related data as needed, and sends it back to the graphic program. Thus the table parameters that carry this data are updated in both the import and export directions.

The graphics program can send many of the same messages in non-confirm-mode that it can in confirm-mode. The difference is that in confirm-mode, the message is a request for permission. If the ABAP program grants it, the action will take place. In non-confirm-mode, the action has already happened. The message functions simply as a notification: the ABAP is not expected to send back permission or update the tables parameters in response.

Updating Networks in Confirm Mode

SAP Network Graphics provides several table parameters for passing data and data changes between the ABAP program and the graphics program. These are all the table parameters:

NODES  Node objects
Confirm-Mode Processing of User Actions

NVALS  Attribute settings for nodes
LINES  Line objects (node-links)
LVALS  Attribute settings for lines
CLUSTERS  Cluster objects (collections of nodes and lines)
CVALS  Attribute settings for clusters
POSITIONS  Node/line positions

LINES and LVALS are valid only with CNET_GRAPH_NETWORK."

Normally, in confirm-mode, a request to update the graph would be passed with certain message types: 'AINS', 'ADUP', 'ADEL', 'ACON', 'AMOD'. For each of these messages, SAP Network Graphics should logically check the following parameters for relevant data:

'AINS' (Insert request)  NODES, LINES, CLUSTERS, POSITIONS
'ADUP' (Copy request)  NODES, LINES, CLUSTERS, POSITIONS
'ADEL' (Delete request)  DELETIONS
'ACON' (Connect request)  LINES, LVALS
'AMOD' (Modify request)  NVALS, LVALS, CVALS

However, in confirm mode, SAP Network Graphics must assume that both primary actions and secondary actions may be requested.

Primary actions are those specified in the message parameter (ABAP_CMD or GR_MES). Secondary actions are those that accompany the primary one, but actually imply a different action code.

For example, the graphics program may return a request from the online user to insert a node. To confirm this action, the ABAP program returns a message of ABAP_CMD='AINS', with complete node information in NODES. Logically, SAP Network Graphics should then check NODES, and insert the nodes specified.

However, perhaps the application also requires that when inserting a node, some other objects (lines or clusters) be added, or their attributes be set, or perhaps even that other objects be deleted. The ABAP program may specify these other updates as secondary actions in any of the other tables listed above. They are all still returned with a message code of 'AINS', although modifications, deletions, and so on, may also be implied.

As a result, in confirm mode, when SAP Network Graphics receives any of the update-related message codes (listed above), it checks all table parameters for possible information. Any data in NODES, LINES, or CLUSTERS implies an add or change; any data in NVALS, LVAL, or CVALS implies an attribute update; any data in DELETIONS implies a deletion.

This is why, when the ABAP program wishes to allow an action, it need only update these tables and return the same message code. If it wishes to deny permission, it need only clear the tables of data. (Partial permission is possible by deleting the disallowed data from the table.)
# General and Specific Function Modules

The following function modules are available for SAP Network Graphics.

For a detailed description of parameters and exceptions, see the R/3 system documentation (Tools → ABAP Workbench, Function library button, or transaction code SE37).

## General Function Modules

<table>
<thead>
<tr>
<th>Function module</th>
<th>Used for</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNET_FIELD_TO_INDEX_NVALS</td>
<td>Supplying values for nodes</td>
</tr>
<tr>
<td>CNET_FIELD_TO_INDEX_LVALS</td>
<td>Supplying values for links</td>
</tr>
<tr>
<td>CNET_FIELD_TO_INDEX_CVALS</td>
<td>Converting field names to text indexes for frames</td>
</tr>
<tr>
<td>CNET_GET_PROFILE_CONTENTS</td>
<td>Obtaining profile contents for customizing a chart</td>
</tr>
<tr>
<td>CNET_SET_OPTIONS</td>
<td>Setting options</td>
</tr>
<tr>
<td>CNET_SET_TIMEEGG_TEXT</td>
<td>Entering text in the &quot;time egg&quot;</td>
</tr>
<tr>
<td>CNET_MAKE_BOXES</td>
<td>Creating boxes</td>
</tr>
<tr>
<td>CNET_SET_LABEL</td>
<td>Entering graphics search text</td>
</tr>
<tr>
<td>CNET_SET_LEGEND</td>
<td>Creating graphics legend</td>
</tr>
<tr>
<td>CNET_GRAPH_ACTION</td>
<td>Adding CUA items for Network Graphics</td>
</tr>
<tr>
<td>CNET_GRAPHIC_PAI</td>
<td>Analyzing data returned by the graphic</td>
</tr>
</tbody>
</table>

## Hierarchy-specific Function Modules

<table>
<thead>
<tr>
<th>Function module</th>
<th>Used for</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNET_GRAPHIC_HIERARCHY*</td>
<td>Starting Hierarchy Graphics</td>
</tr>
<tr>
<td></td>
<td>Combination of the following two modules plus CNET_GRAPHIC_PAI and GRAPH_RECEIVE [Page 109]</td>
</tr>
<tr>
<td>CNET_GRAPHIC_HIERARCHY_PBO</td>
<td>Starting Hierarchy Graphics at PBO time, using a graphic profile (parameter PROFILE)</td>
</tr>
<tr>
<td>CNET_CONVERT_TO_HIERARCHY</td>
<td>Converting links to node attributes</td>
</tr>
<tr>
<td>CNET_GET_NODES_ORDERED</td>
<td>Obtaining the order of hierarchy nodes</td>
</tr>
</tbody>
</table>

## Network-specific Function Modules

<table>
<thead>
<tr>
<th>Function module</th>
<th>Used for</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNET_GRAPHIC_NETWORK*</td>
<td>Starting Network Graphics</td>
</tr>
<tr>
<td></td>
<td>Combination of CNET_GRAPHIC_NETWORK_PBO plus CNET_GRAPHIC_PAI and GRAPH_RECEIVE [Page 109]</td>
</tr>
</tbody>
</table>
**General and Specific Function Modules**

<table>
<thead>
<tr>
<th>Function module</th>
<th>Used for</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNET_GRAPHIC_NETWORK_PBO</td>
<td>Starting Network Graphics at PBO time</td>
</tr>
</tbody>
</table>

* The only interface differences between CNET_GRAPHIC_HIERARCHY and CNET_GRAPHIC_NETWORK are two parameters (LINES and LVALS) in CNET_GRAPHIC_NETWORK that do not occur in CNET_GRAPHIC_HIERARCHY. Also, the NODES tables for the two modules use different reference tables (CNG_NODES and CHG_NODES).

**Cluster Editor-specific Function Modules**

<table>
<thead>
<tr>
<th>Function module</th>
<th>Used for</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNET_GRAPHIC_CLUSTER</td>
<td>Starting the Cluster Editor</td>
</tr>
<tr>
<td></td>
<td>Combination of CNET_GRAPHIC_CLUSTER_PBO plus CNET_GRAPHIC_PAI and GRAPH_RECEIVE [Page 109]</td>
</tr>
<tr>
<td>CNET_GRAPHIC_CLUSTER_PBO</td>
<td>Starting the Cluster Editor at PBO time, using a graphic profile (parameter PROFILE)</td>
</tr>
</tbody>
</table>
Calling SAP Portfolio Graphics

An ABAP program specifies portfolio data and calls up an appropriate function module. The SAP Portfolio Graphics program generates a portfolio representations from the data supplied.

Overview [Page 72]

The GRAPH_PORT Function Module [Page 74]

The GET_PORT_PARAM Function Module [Page 76]

Sample Program [Page 190]
Overview

SAP Portfolio Graphics generates graphs of portfolio representations. These graphs can display business relationships (for example, a comparison of product lines): you can depict objects in relative sizes, their growth trends, and so on. To show such relationships, SAP Portfolio Graphics lets you specify objects for display, their sizes and positions in the graph, connections between objects, graph quadrants, and so on. The online user can browse and zoom the graph, format it with display options, modify numerical values, and so on.

The SAP Portfolio Graphics display looks as shown in the following figure.

You can call SAP Portfolio Graphics from an ABAP program, and generate a portfolio display at the user's workstation terminal. Your program provides the graphical data and calls the SAP Portfolio Graphics function module:

- GRAPH_PORT

Programming Interactive Applications

An ABAP program that allows the online user to open multiple windows or enter data with the mouse or keyboard is interactive. When you use GRAPH_PORT, your application is always...
interactive, since SAP Portfolio always reports user-clicks (selections) back to the ABAP program.

You are provided with two mechanisms for controlling the interaction between ABAP program and online user:

- Dialog programming in Releases prior to 2.1
  Using graphics function modules with dialog parameters STAT, M_TYP, PWDID, SUPER, WINID and RWNID)
- Current dialog programming
  Using graphics function modules with dialog parameters STAT, PWDID, SUPER, and WINID)
  plus utility function modules:
  - GRAPH_RECEIVE (with dialog parameters ACTIVATE_RAISE, ERRORCODE, MCODE, and RWNID)
  - GET_PORT_PARAM

For new programs, you should use only the second mechanism, graphics function modules with the utility function modules.

Do not use both mechanisms together in a program.

For more information, see Programming Interactive Applications [Page 93].

You can also also change the standard graphics display and define additional graphics functions for interactive users. This means when you call up SAP Portfolio Graphics, it appears with changed or extra menu names or pushbuttons.

For more information, see the section Changing the Standard Graphics Display [Page 86]. To define these additional functions, call the appropriate function module before you call an SAP Portfolio Graphics function module.

If your ABAP program calls GRAPH_ACTION, it must also use the CUA_ID and P_TYP parameters (when calling GRAPH_PORT) in order to learn when the user has selected these additional functions.
The GRAPH_PORT Function Module

The GRAPH_PORT Function Module

The GRAPH_PORT function module executes a call to the SAP Portfolio Graphics program. The following table contains a list of all possible parameters you can use with this function module.

For a detailed description of parameters and exceptions, see the R/3 system documentation (Tools → ABAP Workbench, Function library button, or transaction code SE37).

Parameters used with GRAPH_PORT

<table>
<thead>
<tr>
<th>Name</th>
<th>Status</th>
<th>Ref.Field</th>
<th>Initial Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>FRM_BACKCLR</td>
<td>IMPORT</td>
<td>SPACE</td>
<td></td>
</tr>
<tr>
<td>FRM_FRAMCLR</td>
<td>IMPORT</td>
<td>SPACE</td>
<td></td>
</tr>
<tr>
<td>FRM_GRIDX</td>
<td>IMPORT</td>
<td>SPACE</td>
<td></td>
</tr>
<tr>
<td>FRM_GRIDY</td>
<td>IMPORT</td>
<td>SPACE</td>
<td></td>
</tr>
<tr>
<td>FRM_SCALCLR</td>
<td>IMPORT</td>
<td>SPACE</td>
<td></td>
</tr>
<tr>
<td>FRM_SCALMOD</td>
<td>IMPORT</td>
<td>SPACE</td>
<td></td>
</tr>
<tr>
<td>FRM_SHADOW</td>
<td>IMPORT</td>
<td>SPACE</td>
<td></td>
</tr>
<tr>
<td>INBUF **</td>
<td>IMPORT</td>
<td>SPACE</td>
<td></td>
</tr>
<tr>
<td>OBJ_SIZE</td>
<td>IMPORT</td>
<td>SPACE</td>
<td></td>
</tr>
<tr>
<td>OBJ_SVAL</td>
<td>IMPORT</td>
<td>SPACE</td>
<td></td>
</tr>
<tr>
<td>OBJ_XARR</td>
<td>IMPORT</td>
<td>SPACE</td>
<td></td>
</tr>
<tr>
<td>OBJ_XVAL</td>
<td>IMPORT</td>
<td>SPACE</td>
<td></td>
</tr>
<tr>
<td>OBJ_YARR</td>
<td>IMPORT</td>
<td>SPACE</td>
<td></td>
</tr>
<tr>
<td>OBJ_YVAL</td>
<td>IMPORT</td>
<td>SPACE</td>
<td></td>
</tr>
<tr>
<td>PWDID **</td>
<td>IMPORT</td>
<td>SPACE</td>
<td></td>
</tr>
<tr>
<td>STAT **</td>
<td>IMPORT</td>
<td>SPACE</td>
<td></td>
</tr>
<tr>
<td>SUPER **</td>
<td>IMPORT</td>
<td>SPACE</td>
<td></td>
</tr>
<tr>
<td>TITLE_BACKCLR</td>
<td>IMPORT</td>
<td>SPACE</td>
<td></td>
</tr>
<tr>
<td>TITLE_INFO</td>
<td>IMPORT</td>
<td>SPACE</td>
<td></td>
</tr>
<tr>
<td>TITLE_SIZE</td>
<td>IMPORT</td>
<td>SPACE</td>
<td></td>
</tr>
<tr>
<td>TITLE_TEXTCLR</td>
<td>IMPORT</td>
<td>SPACE</td>
<td></td>
</tr>
<tr>
<td>TITLE_TITLE</td>
<td>IMPORT</td>
<td>SPACE</td>
<td></td>
</tr>
<tr>
<td>WINID **</td>
<td>IMPORT</td>
<td>SPACE</td>
<td></td>
</tr>
<tr>
<td>CUA_ID</td>
<td>EXPORT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>M_TYP **</td>
<td>EXPORT</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### The GRAPH_PORT Function Module

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OBJ_ID</td>
<td>EXPORT</td>
<td></td>
</tr>
<tr>
<td>P_TYP</td>
<td>EXPORT</td>
<td></td>
</tr>
<tr>
<td>RBUFF **</td>
<td>EXPORT</td>
<td></td>
</tr>
<tr>
<td>RWNID **</td>
<td>EXPORT</td>
<td></td>
</tr>
<tr>
<td>AREA_TAB</td>
<td>Table</td>
<td>GPOAREA</td>
</tr>
<tr>
<td>AXIS</td>
<td>Table</td>
<td>GPOAXIS</td>
</tr>
<tr>
<td>COL_TEXT</td>
<td>Table</td>
<td></td>
</tr>
<tr>
<td>OBJT</td>
<td>Table</td>
<td>GPOOBJT</td>
</tr>
<tr>
<td>VALUES</td>
<td>Table</td>
<td></td>
</tr>
</tbody>
</table>
The GET_PORT_PARAM Function Module

The GET_PORT_PARAM function module calls SAP Portfolio Graphics to get export parameters set to their proper values. Use GET_PORT_PARAM when the ABAP program has received return values of MCODE='I' and RWNID='PORTn'. See Controlling the Interaction Using Parameters and Utility Function Modules [Page 109] for more information.

The following table contains a list of the parameters you can use with this function module.

For a detailed description of parameters and exceptions, see the R/3 system documentation (Tools → ABAP Workbench, Function library button, or transaction code SE37).

Parameters used with GET_PORT_PARAM

<table>
<thead>
<tr>
<th>Name</th>
<th>Status</th>
<th>Initial Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CUA_ID</td>
<td>EXPORT</td>
<td></td>
</tr>
<tr>
<td>OBJ_ID</td>
<td>EXPORT</td>
<td></td>
</tr>
<tr>
<td>P_TYP</td>
<td>EXPORT</td>
<td></td>
</tr>
</tbody>
</table>
Calling SAP Bar Chart Graphics

An ABAP program specifies event data and calls up an appropriate function module. The SAP Bar Chart Graphics program generates a table of events from the data supplied.

Overview [Page 78]
The Function Modules [Page 80]
Sample Program [Page 206]
Overview

The SAP Bar Chart Graphics program displays a table of events over a given time period and allows the online user to reschedule them.

Apart from the standard elements (menu bar, symbols bar and pushbutton bar), the SAP Bar Chart Graphics window contains part of or all of the following elements:

- **Graph title in the window title bar**
  The graph title is defined in the ABAP program that called the graphics program.

- **Time line**
  The time line sets the time span for the display areas.

- **Display areas (8 max.)**
  You can display a maximum of eight bar charts in a graphics window. Each display contains a title which is defined in the calling ABAP program. You can resize the displays by dragging the dividing line between the displays.

  Each display consists of the following elements:
  - **Time units line**
    For each display, the online user can choose one of the time units provided. The time span is set by the time line.
  - **Tables area**
    The tables section on the left contains the table heading and the table fields for each graph area. The online user can adjust the width of the columns. The online user can also change the width of the tables section and with it the width of the graph display area.
  - **Graph display area**
    The graph display area contains objects (event blocks) depicting the duration of events. Each of the graph areas contains its own time units line within the limits of the time line. The online user can choose among a number of time units (day, week, etc.).
The standard pushbuttons in the pushbutton bar let you change the size of the graph display and the tables section.

The calling ABAP program determines which elements are to contained in the graphics window. You can define more menus with additional functions or leave out standard functions and symbols. For detailed information, see your application-specific SAP Bar Charts Graphics documentation.

For sample programs calling SAP Bar Chart Graphics, use transaction code `gral` in the R/3 System.
The Function Modules

The available function modules are listed in the table below.

For a detailed description of parameters and exceptions, see the R/3 system documentation (Tools → ABAP Workbench, Function library button, or transaction code SE37).

You can customize your graphics application using a profile (see parameter PROFILE) to be defined in the R/3 Customizing facility.

To change the standard graphics display, edit reference screen SAPLBARC using Menu Painter [Page 89] and call function module GRAPH_SET_CUA_STATUS [Page 90] before calling BARC_GRAPHIC_PBO. GRAPH_SET_CUA_STATUS sends CUA information to the graphics program.

For a detailed description of parameters and exceptions, see the R/3 system documentation (Tools → ABAP Workbench, Function library button, or transaction code SE37).

SAP Bar Chart Function Modules and what they are used for

<table>
<thead>
<tr>
<th>Function module</th>
<th>Used for</th>
</tr>
</thead>
<tbody>
<tr>
<td>BARC_GRAPHIC_PBO</td>
<td>Starting bar chart at PBO time, using a graphic profile (parameter PROFILE)</td>
</tr>
<tr>
<td>BARC_GRAPHIC_PA1</td>
<td>Analyzing data returned by the graphic</td>
</tr>
<tr>
<td>BARC_SET_TIME_AXIS</td>
<td>Setting start and end of time axis</td>
</tr>
<tr>
<td>BARC_SET_OPTIONS</td>
<td>Setting options</td>
</tr>
<tr>
<td>BARC_ADD_CHART</td>
<td>Creating a chart</td>
</tr>
<tr>
<td>BARC_SET_CHART_ATTRIB</td>
<td>Setting chart attributes</td>
</tr>
<tr>
<td>BARC_ADD_SECTION</td>
<td>Creating a section on the time axis</td>
</tr>
<tr>
<td>BARC_SET_SECTION_ATTRIB</td>
<td>Setting section attributes</td>
</tr>
<tr>
<td>BARC_ADD_RIBBON</td>
<td>Adding a ribbon to the time axis</td>
</tr>
<tr>
<td>BARC_SET_RIBBON_ATTRIB</td>
<td>Setting attributes for ribbons in the chart</td>
</tr>
<tr>
<td>BARC_ADD_GRID</td>
<td>Adding a time grid</td>
</tr>
<tr>
<td>BARC_SET_GRID_ATTRIB</td>
<td>Setting grid attributes</td>
</tr>
<tr>
<td>BARC_ADD_LAYER</td>
<td>Adding a layer (graphic elements)</td>
</tr>
<tr>
<td>BARC_SET_LAYER_ATTRIB</td>
<td>Setting layer attributes</td>
</tr>
<tr>
<td>BARC_ADD_LINE</td>
<td>Adding a line</td>
</tr>
<tr>
<td>BARC_ADDCALENDAR</td>
<td>Creating a calendar</td>
</tr>
<tr>
<td>BARC_SETCALENDAR_ATTRIB</td>
<td>Setting attributes for a calendar</td>
</tr>
<tr>
<td>BARC_ADDTIME_PROFILE</td>
<td>Creating time profiles</td>
</tr>
<tr>
<td>BARC_SETTIME_PROFILE_ATTRIB</td>
<td>Setting attributes for time profile</td>
</tr>
<tr>
<td>BARC_ADD_INTERVAL</td>
<td>Adding a time interval</td>
</tr>
<tr>
<td>Function Module</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>----------------------------------------------------------</td>
</tr>
<tr>
<td>BARC_SET_INTERVAL_ATTRIB</td>
<td>Setting time interval attributes</td>
</tr>
<tr>
<td>BARC_ADD_TIME_OBJECT</td>
<td>Creating a time object</td>
</tr>
<tr>
<td>BARC_CONVERT_DATE</td>
<td>Creating a date string in bar chart format</td>
</tr>
<tr>
<td>BARC_REVERT_DATE</td>
<td>Converting a date string in bar chart format to date and time</td>
</tr>
<tr>
<td>BARC_ADD_DATELINE</td>
<td>Creating a date line</td>
</tr>
<tr>
<td>BARC_SET_DATELINE_ATTRIB</td>
<td>Setting dateline attributes</td>
</tr>
<tr>
<td>BARC_GET_PROFILE_CONTENTS</td>
<td>Obtaining profile contents for customizing a chart</td>
</tr>
<tr>
<td>BARC_GET_COLUMN_WIDTH</td>
<td>Selecting new column width</td>
</tr>
<tr>
<td>BARC_SET_COLUMN_WIDTH</td>
<td>Setting the column width</td>
</tr>
<tr>
<td>BARC_GET_TEXTINDEX</td>
<td>Obtaining the text index of a field</td>
</tr>
<tr>
<td>BARC_SET_LABELS</td>
<td>Positioning the chart display</td>
</tr>
<tr>
<td>BARC_SET_COLUMN_ATTRIB</td>
<td>Setting column attributes</td>
</tr>
<tr>
<td>BARC_SET_ROW_ATTRIB</td>
<td>Setting row attributes</td>
</tr>
<tr>
<td>BARC_SET_ROW_HEIGHT</td>
<td>Setting the line height</td>
</tr>
<tr>
<td>BARC_SET_MAXCHARTS</td>
<td>Setting the maximum number of charts sent</td>
</tr>
</tbody>
</table>
Calling SAP Structural Graphics

Calling SAP Structural Graphics
An ABAP program specifies hierarchy data and calls up an appropriate function module. The SAP Structural Graphics program generates a graphical hierarchy of nodes from the data supplied.

Overview [Page 83]
The Function Modules [Page 84]
Sample Program [Page 217]
Overview

SAP Structural Graphics allows you to display and edit any structures that can be represented hierarchically.

You can call SAP Structural Graphics from an ABAP program, and generate a graphic at the user's workstation terminal. Your program specifies the nodes in the hierarchy and calls the following function module:

- GRAPH STRUCTURAL

In addition to GRAPH STRUCTURAL, other function modules are available for controlling SAP Structural Graphics.

💡

All screen elements of SAP Structural Graphics are static. You cannot use the function modules GRAPH_ACTION and GRAPH_SET_CUA_STATUS.

Programming Interactive Applications

If your ABAP program allows the online user to open multiple windows or enter data with the mouse or keyboard, the application is interactive.

To make SAP Structural Graphics interactive, you must use the function module GRAPHSTRU_SET_TOOLBOX.
The Function Modules

The function module GRAPH_STRUCTUREAL calls the SAP Structural Graphics program.

The only mandatory parameter is the object table which also contains the links between objects. In addition, a few standard dialog parameters that enable simultaneous control of several graphics and determine the initial layout of the structural graphic are available.

Instead of displaying the graphic on the screen, you can also send it to another user as an SAPoffice object. In the process, a document of the type 'GRA' is generated. As a result, when you select this document in the inbox, the graphic is opened together with the data sent.

The function module is exited when a message is received from the structural graphic (e.g. editing function) or from another graphic that has also been started. Parameters returned contain the handle of the graphic that issued the message and the message type.

The function module GET_STRU_PARAM lets you further evaluate the message received for the structural graphic. This is also available for other graphic types with the function modules GET_xxxx_PARAM.

An exception is triggered when the object table is empty when starting or reloading or if accessing the database structure GSUOBJ has failed. The latter only applies when the toolbox is being used.

The GRAPH_STRUCTUREAL Function Module

The function module GRAPH_STRUCTUREAL serves as the graphics driver for the SAP Structural Graphics.

The following table contains a list of all possible parameters you can use with this function module.

For a detailed description of parameters and exceptions, see the R/3 system documentation (Tools → ABAP Workbench, Function library button, or transaction code SE37).

Parameters for GRAPH_STRUCTUREAL

<table>
<thead>
<tr>
<th>Name</th>
<th>Status</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADDSHAPES</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>EDITABLE</td>
<td>Import</td>
<td>X</td>
</tr>
<tr>
<td>EXTACTBAR</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>FOCUS</td>
<td>Import</td>
<td>X</td>
</tr>
<tr>
<td>NEWFORMAT</td>
<td>Import</td>
<td>X</td>
</tr>
<tr>
<td>PATTERN</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>PWDID **</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>REFRESH **</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>SO_SEND</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>TBOXOPEN</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>TBOXPOSX</td>
<td>Import</td>
<td>SPACE</td>
</tr>
</tbody>
</table>
The Function Modules

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>TBOXPOSY</td>
<td>Import</td>
</tr>
<tr>
<td>TBOXSIZEX</td>
<td>Import</td>
</tr>
<tr>
<td>TBOXSIZEY</td>
<td>Import</td>
</tr>
<tr>
<td>WINPOSX</td>
<td>Import</td>
</tr>
<tr>
<td>WINPOSXY</td>
<td>Import</td>
</tr>
<tr>
<td>WINSIZEX</td>
<td>Import</td>
</tr>
<tr>
<td>WINSIZEY</td>
<td>Import</td>
</tr>
<tr>
<td>M_TYP **</td>
<td>Export</td>
</tr>
<tr>
<td>RWNID **</td>
<td>Export</td>
</tr>
<tr>
<td>OBJECTS</td>
<td>Internal Table</td>
</tr>
</tbody>
</table>

Note: ** denotes a dialog parameter

For GRAPH_STRUCTURAL, only the OBJECTS parameter is required.

**Additional Function Modules**

In addition to GRAPH_STRUCTURAL, there are function modules that control SAP Structural Graphics:

**GRAPHSTRU_SET_DISPLAY**

This function lets you display any text in a small information window. The function module is useful when, for example, information has to be displayed for objects being selected in a graphic.

**GRAPHSTRU_SET_MESSAGE**

This function module enables you to send a message to a graphic and to display it in the status line there.

**GRAPHSTRU_SET_OPTIONS**

Structural Graphics has many options for settings that cannot all be managed by the GRAPH_STRUCTURAL function module. These settings can be defined using this function module before starting the graphic.

**GRAPHSTRU_SET_PROFILES**

Two design profiles are available in Structural Graphics to set the color and shape of objects and linking lines. One profile is type oriented (depends on the object type or type of linking), the other is status oriented (depends on the status of the objects and links). Before Structural Graphics starts, it is informed of the profiles by this function module.

**GRAPHSTRU_SET_TOOLBOX**

An important feature of structural graphics enables you to edit the object or the structure. The desired editing functions are provided by a toolbox in an amodal window or an extended menu bar.

Before you start the graphic, you can use this function to define the toolbox.
Changing the Standard Graphics Display

You can use special function modules to change or extend the CUA items of the graphics display.

**Overview [Page 87]**

**Calling GRAPH_SET_CUA_STATUS [Page 90]**

**Calling GRAPH_ACTION (CNET_GRAPH_ACTION) [Page 91]**
Overview

The following function modules let you change the standard graphics display.

- **GRAPH_SET_CUA_STATUS**
  Lets you change existing CUA Items [Page 88] or add new items to the graphics display using Menu Painter [Page 89].
  Used before calling any of the graphics program, except SAP Business Graphics, SAP HPGL Display and SAP Structural Graphics.

  ![Note]

  You should use this function module when writing new programs. However, it is **not** available for SAP Business Graphics and SAP HPGL Display.

- **GRAPH_ACTION**
  Lets you add your own CUA items to the standard graphics display.
  Used before calling any of the graphics function modules except those that call SAP Network Graphics and SAP Bar Chart Graphics.
  For SAP Network Graphics, use CNET_GRAPH_ACTION before calling the appropriate function module.

  ![Warning]

  Do not use GRAPH_SET_CUA_STATUS and GRAPH_ACTION (or CNET_GRAPH_ACTION) together in a program.
  CUA information would be overwritten.

What the Function Modules Do

The function modules let you change or extend the CUA items of the graphics display:

- **Menus (in the menu bar)**
  You can define up to eight levels of sub-menus for each menu. In addition, the user can press F1 while selecting a menu to request a help text.

- **Pushbuttons**
  The function modules store the information in static storage, where it is used by the next-called graphics program to extend its window. (The graphics program also deletes the information from static storage to prevent any other calls from using it.)

  When the user later clicks on one of these items (a given button or menu entry), the waiting function module returns the corresponding message to the ABAP program. Your program can then respond to the user requests as needed.
CUA Items

User interface items such as menus and pushbuttons complying with the CUA standard (Common User Access).

An ABAP program can modify a standard graphics window by renaming existing menus and pushbuttons or defining additional graphics functions. These changes to the normal graphics functions can be made with all graphics programs except SAP Business Graphics and SAP HPGL Display.
Menu Painter

Tool provided with the ABAP Development Workbench

The Menu Painter is used to assign the functions implemented in a program to certain menus, function keys and pushbuttons.

The call the tool, use transaction code SE41.
Calling GRAPH_SET_CUA_STATUS

Calling GRAPH_SET_CUA_STATUS

This function module sends CUA information created with Menu Painter to the graphics program. It is not available for SAP Business Graphics, SAP HPGL Display and SAP Structural Graphics.

Menu entries that trigger direct action in the graphics program are defined through standardized function codes.

All of these function codes are defined in the reference screen and may be referenced via include menus.

The following reference screens are available:

- SAP Hierarchy Graphics: SAPLHIER / MAIN
- SAP Network Graphics: SAPLCNET / GRAFIK
- SAP Bar Chart Graphics: SAPLBARC / GRAFIK
- SAP Statistical Graphics: SAPLSTAT / MAIN
- SAP Gantt Chart Graphics: SAPLGNAT / MAIN
- SAP Pushbutton Graphics: SAPLBMAT / MAIN
- SAP Text Matrix Graphics: SAPLBMAT / MAIN
- SAP Portfolio Graphics: SAPLPORT / MAIN

When you create application-specific menu entries, you can use function codes beginning with X, Y or Z.

As the online user selects a menu function, the function code is reported back directly to the ABAP program, where it is analyzed using the following return parameters of the respective function module:

For a detailed description of parameters and exceptions, see the R/3 system documentation (Tools → ABAP Workbench, Function library button, or transaction code SE37).

- SAP Network Graphics: GR_MES
- SAP Bar Chart Graphics: GRAPH_CMD
- All other graphics programs: M_TYP = 'I', <x> = '0' → CUA_ID = <FCode>

If a modified user interface is to be used, you must set the CUA status using the appropriate parameters with GRAPH_SET_CUA_STATUS before each call to the graphics program.

Do not use GRAPH_SET_CUA_STATUS and GRAPH_ACTION (or CNET_GRAPH_ACTION) together in a program.

CUA information would be overwritten.
Calling GRAPH_ACTION (CNET_GRAPH_ACTION)

This function module sends additional CUA information to the graphics program.

GRAPH_ACTION is **not** available for SAP Business Graphics and SAP Structural Graphics.

You can define additional CUA items as follows:

1. Call GRAPH_ACTION (or CNET_GRAPH_ACTION for SAP Network Graphics) with the desired CUA information.
2. Call the function module for the graphics program running in the window to be extended.
   Your call to this function module must request a screen refresh (with STAT='3' or '5') in order to display the new CUA items in the window.

**When using GRAPH_ACTION:** The ABAP program should ensure that no calls to other graphics programs intervene between the call to GRAPH_ACTION and the call to the graphics program for the window being extended. If another call does intervene, the CUA buttons will appear in the wrong window.

For example, if you want to extend an SAP Statistics window with CUA items, but call GRAPH_GANTT before the statistics function module, the items will appear in the GANTT window.

An exception to the above rule is SAP Business Graphics, which does not respond to the CUA information. You can call as many BUSG function modules as you like without causing the button information to be deleted from static storage. (Note, however, that this may change in future releases.)

**When using CNET_GRAPH_ACTION:** Only SAP Network Editor function modules react to CUA information registered with CNET_GRAPH_ACTION. No function calls to other graphics programs will interfere.

**In general:** You may call GRAPH_ACTION or CNET_GRAPH_ACTION to update a new window or one that is already open. CUA items can be added or changed anytime you call a function module for that graphics program. If you have multiple windows open for a single graphics program, only the very next window updated is affected.

GRAPH_ACTION and CNET_GRAPH_ACTION use the same parameters. The following table contains a list of these parameters.

For a detailed description of parameters and exceptions, see the R/3 system documentation (*Tools → ABAP Workbench, Function library* button, or transaction code SE37).

**Parameters used with GRAPH_ACTION and CNET_GRAPH_ACTION**

<table>
<thead>
<tr>
<th>Name</th>
<th>Status</th>
<th>Initial Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACTBAR</td>
<td>Internal table</td>
<td></td>
</tr>
<tr>
<td>ACTION</td>
<td>Internal table</td>
<td></td>
</tr>
<tr>
<td>HELP</td>
<td>Internal table</td>
<td></td>
</tr>
</tbody>
</table>

These three table parameters do not use any reference tables defined in the ABAP Dictionary. However, your declarations for these parameters must match exactly the structures described in the parameter description.
Calling GRAPH_ACTION (CNET_GRAPH_ACTION)

After Calling GRAPH_ACTION

Graphics function modules called after GRAPH_ACTION report user selection of CUA items in a type parameter (called B_TYP, G_TYP, S_TYP, and so on, depending on the graphics program) and the CUA_ID parameter. If the type parameter contains:

'0': The user selected a menu-entry. CUA_ID contains the contents of the field ACTBAR-<Action-ID>.

'1': The user requested help for a menu-entry. CUA_ID contains the contents of the field ACTBAR-<Help-ID>.

'2': The user clicked on a function key. CUA_ID contains the contents of the field ACTION-<Action-Text>.

After Calling CNET_GRAPH_ACTION

SAP Network Graphics function modules report user selection of CUA items in the GR_MES export parameter.

GR_MES can report back many kinds of user actions, both fixed and variable. (Fixed values are listed in "The SAP Network Graphics Function Modules"). If GR_MES contains anything other than one of the fixed values, it must be one of the CUA-related values declared when calling CNET_GRAPH_ACTION. The CUA-related values are those found in the parameter fields:

ACTBAR-<Action-ID>

ACTBAR-<Help-ID>

ACTION-<Action-Text>
Programming Interactive Applications

If an application is to allow user input, dialog programming is required.

Introduction to Dialog Programming [Page 94]

Dialog Programming in Releases Prior to 2.1 [Page 102]

Current Dialog Programming [Page 108]
Introduction to Dialog Programming

The following topics explain the basics of dialog programming.

Overview [Page 95]  
Which ABAP Programs Require Dialog Programming [Page 96]  
Offering Interactive Capabilities [Page 98]  
Managing Multiple Windows [Page 99]  
Summary of Dialog Programming Features [Page 100]
Overview

If your ABAP program is to allow the online user to open multiple windows or enter data with the mouse or keyboard, the application is interactive and requires dialog programming.

ABAP programs that call graphics function modules may be simple or complex.

If your application is very simple, you may not need to do any dialog programming at all. The simplest applications may only use a single graphics program in a single window: they call the function module to open the window, display some data in it, and wait for the online user to close the window.

A slightly more complex application uses the same single window, but allows some interaction between the ABAP program and the online user.

Even more complicated applications may perform the same interactive functions, but from more than a single window.

If you are unsure of whether your application requires dialog programming or not, read the topic "Which ABAP programs Require Dialog Programming" in this section. This topic defines "simple" programs, and how these can avoid dialog programming.

Two mechanisms are provided for controlling the interactions between ABAP program and online user. Explaining these two mechanisms is the chief task of this section.

- **Dialog programming in Releases prior to 2.1**
  
  In existing ABAP programs, dialog parameters used to be used in calls to graphics function modules to tell the function module to wait for and receive user input.

  Do not use this method any longer, except for existing programs.

- **Current dialog programming**

  In new ABAP programs with calls to graphics function modules, you must also call GRAPH_RECEIVE, which will wait for messages. When a message has been received, you must call the appropriate GET_xxxx_PARAM to get the details on that message.

  ![](image)

  The function module GET_xxxx_PARAM is not available in SAP Network Graphics and Bar Chart Graphics.
Which ABAP Programs Require Dialog Programming

The ABAP programmer is provided with two mechanisms for controlling the interaction between ABAP program and online user:

- Graphics function modules with dialog parameters (Parameters STAT, M_TYP, PWDID, SUPER, WINID and RWNID)
- Graphics function modules with dialog parameters (Parameters STAT, PWDID, SUPER, and WINID) plus utility function modules:
  - GRAPH_RECEIVE (with dialog parameters ACTIVATE_RAISE, ERRORCODE, MCODE, and RWNID)
  - GET_xxxx_PARAM

For new programs, you should use only the second mechanism, graphics function modules with the utility function modules.

Your program may not require dialog programming at all, depending on whether your application is simple, more complex or complex.

- "Very simple" applications:
  - Your program will not open more than one window at a time. This means your program does not need to manage multiple windows. Hence it does not need to use any of the WINID, PWDID, or SUPER dialog parameters.
  - Your program will not respond to any mouse clicks or keyboard input from the online user. It merely starts up the graphics program for the user without regard to what happens after that.

Even though your program will not respond to user input, it must ask the function module to wait for input. This is because at least one function module call in each ABAP program must be a waiting function module. ("Waiting for user input" under the relevant topics "Dialog Programming Using the Dialog Parameters" and "Dialog Programming With GRAPH_RECEIVE and GET_xxxx_PARAM" explains why.)

You do not need any dialog parameters for this kind of program. The default value of STAT specifies waiting, and thus can be omitted.

If your program doesn't respond to user input, neither of the dialog parameters that report this input are needed.

- "More complex" applications:
  - Your program opens only one window, and waits for the user to select menu items. (The GRAPH_HIERARCHY_MENUE and GRAPH_BUTTON_MENUE function modules generate menus and report back what the user selected. Menu parameters returned include the menu entry and the hierarchy node or pushbutton to which the entry applies.)

For this scenario, you don't need the STAT parameter, since waiting is the default. You do, however, need message code parameter (MCODE or M_TYP) in order to know when the user has closed the window. You can ignore the other dialog parameters.

- "Complex" applications:
Which ABAP Programs Require Dialog Programming

- Your program opens multiple windows and will wait for user input from these windows.
  You must use the STAT, MCODE (M_TYP), WINID, and PWDID (and possibly also SUPER) parameters.

- Your program responds to other user input beyond mere menu selection. (Such input includes pressing the Enter or other keyboard keys.) You must use STAT, MCODE (M_TYP), and RWNID.

Your program might allow the online user to select a menu item in the first window, which then starts up a second window. The graphics program running in the second window might be the same as the program in the first, or they might be different.

For example, SAP Hierarchy might run in the first window, and might allow the user to select a node. Node selection might then trigger a second window running SAP Statistics, where a graph is displayed for the selected node.
Offering Interactive Capabilities

ABAP applications can allow the online user to enter several kinds of input. Each graphics function module can report on some or all of the following user actions:

- Select items in the graphic window: hierarchy nodes, buttons in a pushbutton pad, bars in a bar chart, points in a curve, and so on. The function module reports the identity of the selected object back to the application program.

- Select items in the graphic window. This is the same as the above, except that the function module presents the online user with a menu. The user then selects a menu item. The function module reports the identity of the selected object and the menu item back to the application program.

- Edit the numerical value of graph items (such as bars in a bar chart). The function module reports the identity of the edited object and its new value back to the application program.

- Enter keyboard input. The function module reports the key pressed back to the application program.

- Close the graphics window, using menu options or the F3 key at the bottom of the window. The function module reports back to the application program that the graphics program has been killed.

The ABAP program can then respond to these requests as required by the application. These forms of user input should not be confused with the use of menus and function keys supported by the graphics program itself. Two kinds of user input should be distinguished:

- User actions requesting functions not local to the graphics program. These are all the actions listed above. Note that selection of menu items (mentioned in the second point above) refers to menus invoked by clicking on a graph object. These functions cannot be performed by the graphics program alone. The function module and the ABAP program are always involved.

- User actions requesting functions entirely local to the graphics program. These functions are performed by the graphics program without reporting back to the function module. The actions consist of:
  - Clicking on function key buttons shown in the window
  - Clicking on menu items, when the menu is offered in the menu bar of the graphics window

The only exception is ending the graphics program, which the user can request with a menu option or a function key. This request is fulfilled by the graphics program, but is also reported back to the ABAP program.

Your ABAP program can safely ignore all actions in the latter category except requests to end the graphics program.
Managing Multiple Windows

If a window is open, the graphics program running in it is alive on the workstation. This is true even if the function module that called it has returned.

If two windows running the same graphics program are open, there are two copies of the underlying program running. Each copy handles requests from its own window. If the user closes one of the windows, the other one is still alive.

The ABAP program must be able to handle all of the following window-related functions:

- Provide window ID's, when the window is opened.
  
  If you do not do this, the window is named by default. (See the WINID parameter.)

- Receive user input from any window.
  
  This means the program must recognize window ID's and which function modules created them. (See the RWNID parameter.)

- Send new data to any window. (See the WINID parameter.)

- Keep track of windows that are closed.
  
  An ABAP program must not send data to a window that has been closed. The closed window can however be re-opened with the same name.

- Provide a function module with the name of a parent window when creating its child. (See the PWDID parameter.)

For a detailed description of parameters and exceptions, see the R/3 system documentation (Tools → ABAP Workbench, Function library button, or transaction code SE37).

The ABAP program need not maintain the tree of parent-child relationships between windows. This information is maintained on the graphics program side, and does not depend on which graphics programs are running or which function module is waiting for input.

The ABAP program tells the function module how windows are to be closed using the SUPER or PWDID parameters. This information is transmitted to the graphics program. When the online user later exits from the window, the graphics program terminates, closing the window (and possibly its children) as instructed.
Summary of Dialog Programming Features

The ABAP programmer is provided with two mechanisms for controlling the interaction between ABAP program and online user. Your program needs some of the features provided, but may not need all. Note that you must never use both mechanisms together in a program.

The two mechanisms are as follows:

- Dialog programming using the dialog parameters
  
  In calls to graphics function modules, dialog parameters used to be used to tell the function module to wait for and receive user input. The waiting function module intercepts messages stored in a queue, regardless of their origin, and sets the values of all export parameters. Thus, it only makes sense to ask one function module at a time to wait for input.

  For compatibility with older ABAP programs, the non-utility graphics functions can still be used to return their own export parameters. (This requires using the RWNID and M_TYP parameter, and more values for STAT parameter.) This interface is still available, but is not recommended for developing new programs.

- Dialog Programming With GRAPH_RECEIVE and GET_xxxx_PARAM
  
  For new programs, you should use this mechanism only.

  After calling a graphics function module, you must also call GRAPH_RECEIVE, which will wait for messages. ("Waiting for user input" in the section Controlling the Interaction Using Parameters and Utility Function Modules [Page 109] explains why.)

  Every ABAP program that responds to user input (as reported by GRAPH_RECEIVE) must call the GET_xxxx_PARAM function for the corresponding graphics program.

  The normal graphics functions offer many export parameters that have no meaningful values until activated by calling the utility GET_xxxx_PARAM. In essence, the normal graphics functions have only importing and table parameters, and the GET_xxxx_PARAM functions only exporting and table parameters.

  GET_xxxx_PARAM is a collection of function modules, one for each graphic program. The letters xxxx can stand for: BMAT, BUSG, GANTT, HIER, HPGL, PORT, or STAT.

  You can use the following dialog parameters with the function module that calls your graphics program.

  - STAT
    
    You must use the STAT parameter whenever your program will transfer data to a window that is already open.

  - WINID
    
    In general, WINID is necessary whenever your program opens several windows and needs to transfer data to them selectively.

  - PWDID and SUPER
PWDID and SUPER specify how to close windows when your program has several open at a time. Neither of these parameters is ever required, but are mutually exclusive if you do use them.

You need not use the dialog parameters in your application program. However, if you don't, the function modules assume particular default values for them. These default values are the same, regardless of which function module you call.

For a detailed description of parameters and exceptions, see the R/3 system documentation (Tools → ABAP Workbench, Function library button, or transaction code SE37).
Dialog Programming in Releases Prior to 2.1

In Releases prior to 2.1, the interplay between the online user and the ABAP program is exclusively controlled by a set of dialog parameters.

Controlling the Interaction Using Dialog Parameters [Page 103]

Closing Windows from the ABAP Program Using GRAPH_DIALOG [Page 106]
Controlling the Interaction Using Dialog Parameters

To control the interplay between online user and the ABAP program, the graphics function modules offer a set of dialog parameters:

**Telling the function module what to do**
- STAT: status with which function module called

**Receiving input from the user**
- M_TYP: type of message returned from graphics program
- RWNID: ID of window in which user input occurred

**Opening and closing windows**
- PWDID: parent window for the window being opened
- WINID: ID of current window (existing or to be created)
- SUPER: marks current window as "exit window"

STAT is the central parameter. This parameter tells the function module which functions to perform on a given call:
- open a window and start the graphics program in it
- transfer data to the graphic window
- wait for input from the window

The most common pattern for programs allowing multiple windows is to place the various function module calls in a loop. A pass through the loop is made each time new input is received from the user. (See the programming scheme in "Waiting for User Input").

The STAT value for calling a function module is changed depending on which of the three functions above are needed at a particular time. Usually, for example, a window must be opened on the first call, but not thereafter.

You need not use the dialog parameters in your application program. However, if you don't, the function modules assume particular default values for them. The default values for these parameters are the same, regardless of which function module you call.

**Waiting for User Input**

You use the STAT parameter to tell a function module to wait for user input. A waiting function module does not return to the ABAP program until it has received input from a graphics program. This is true even if the input is only a report that a graphic window has been closed.

Every ABAP program that calls graphics function modules must ask one of the function modules to wait. This is because calling a Wait function module has the effect of "realizing" actions requested in No-wait function calls:
- Windows created in No-wait calls do not appear on the screen until the call to the waiting function module takes place.
- Data sent to existing windows in No-wait calls are not actually displayed until a Wait call to a function module happens.
Controlling the Interaction Using Dialog Parameters

A waiting function module receives input from all graphics programs, not just its own. All messages returned from any running graphics program are written to a common queue. A waiting function module intercepts all messages using this queue, regardless of their origin. (However, the function module returns to the ABAP program with a single message at a time.)

Thus it is only sensible for one function module at a time to be asked to wait for input. The waiter picks up all messages in the queue, and makes them available to the program at large.

All other function modules are called with a non-wait STAT value, and should be regarded as "output only" calls. They merely execute their other functions (opening new graphics windows and transferring data to them) and return to the ABAP program. Graphics programs called by non-waiting function modules are still running, however, after the module returns to the ABAP program.

The following example is a simple scheme for combining multiple calls to function modules in a program.

```
LOOP UNTIL (user input available)
  ...
  CALL FunctMod1
    EXPORTING STAT = STAT1.    " A No-wait call
  CALL FunctMod2
    EXPORTING STAT = STAT2.    " A No-wait call
  CALL FunctMod3
    EXPORTING STAT = STAT3.    " A No-wait call
  CALL FunctMod4
    EXPORTING STAT = STAT4   " A Wait call
  ...
  [Set STAT values and conditions controlling how the function modules are to be called on subsequent iterations]
  ...
ENDLOOP
```

In the above example, only the use of the STAT parameter is illustrated. (Other dialog parameters such as window parameters have been omitted.)

Here the programmer wanted to open several windows simultaneously, but not take input from the user until the last one is open.

The STAT values should change after the each time through the loop, depending on the input reported by FunctMod4.

In the above scheme, the windows created in the first (No-wait) calls do not appear on the screen until the last (Wait) function module has been called. As a result, no input to any window by the user can get lost.

How the Waiting Function Module Handles Return Parameters

The waiting function module handles received messages differently, depending on which graphics program sends them. If the messages are from its own graphics program, (for example,
if GRAPH_3D receives a message from SAP Business Graphics, the function module sets the values of all export parameters and returns to the ABAP program.

In the example mentioned above, GRAPH_3D can set return parameters for all messages from SAP Business Graphics. This is true for all SAP Business Graphics parameters, regardless of whether GRAPH_3D itself uses them or not. That is, any function module for a given graphics program can set all the return parameters for that graphics program, even parameters for other function modules, as long as the other function modules call the same graphics program.

However, when the waiter receives messages from graphics programs not its own (for example, GRAPH_3D receives a message from SAP Statistics), it has no information on the relevant export parameters and how to set them. In this case, the waiting function module has no choice but to return with invalid export parameters.

**How the ABAP Program Should Handle Return Parameters**

When the waiting function module returns, the ABAP program needs to know whether the parameters returned have valid values in them or not. (This can be determined by inspecting the M_TYP parameter.) If not, special processing is necessary to get the needed parameter values.

To learn which graphics program sent the message received, the ABAP program should check the RWNID parameter. RWNID tells the name of the window where the user input occurred. This window name ("window ID") also identifies the graphics program running in the window. (See the WINID and RWNID parameter descriptions for information on window ID formats.)

If (based on the value of M_TYP), the exported parameters already contain the returned information, the ABAP program can use them right away.

However, if the exported parameters are invalid, the ABAP program must call *any function module associated with the graphics program*, sending it a STAT value of 'R' (Reuse). This asks the function module to take parameter information out of static storage and set the appropriate return parameters.

For example, when the message comes from SAP Business Graphics, the ABAP program can call any of the Business Graphics function modules. It need not be the function module that opened the window in the first place.)
Closing Windows from the ABAP Program Using GRAPH_DIALOG

The GRAPH_DIALOG function module is a feature of the old waiting mechanism, which is not recommended for new programs.

The GRAPH_DIALOG function module does not call a specific graphics program. Instead, GRAPH_DIALOG offers specific utility functions:

- closing windows
- performing waiting

Parameters used

The following table contains a list of all possible parameters you can use with this function module.

For a detailed description of parameters and exceptions, see the R/3 system documentation (Tools → ABAP Workbench, Function library button, or transaction code SE37).

Parameters used in GRAPH_DIALOG

<table>
<thead>
<tr>
<th>Name</th>
<th>Status</th>
<th>Initial Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLOSE</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>INBUF **</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>KWIDID</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>STAT **</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>M_TYP **</td>
<td>Export</td>
<td></td>
</tr>
<tr>
<td>RBUFF **</td>
<td>Export</td>
<td></td>
</tr>
<tr>
<td>RNID **</td>
<td>Export</td>
<td></td>
</tr>
</tbody>
</table>

Note: ** denotes a dialog parameter

If you merely want to close windows, you only need the KWNID or CLOSE parameters.

If you merely want the function module to perform waiting, you only need the M_TYP and RNID parameters. A STAT value of SPACE, a Wait value, is assumed by default. (The INBUF and RBUFF parameters are old and no longer needed. Do not use them when updating an existing programs.)

Closing Windows with GRAPH_DIALOG

The primary reason to call GRAPH_DIALOG is to kill a window from the ABAP program. The program can close one or all windows at a time.

Normally, windows are closed by the online user. This action is merely reported back to the ABAP program by the function module waiting for user input.
Occasionally, however, there may be reasons why the ABAP program needs to close the window itself. In this case, the ABAP program should call GRAPH_DIALOG with the KWDID parameter set to a valid window ID.

**Using GRAPH_DIALOG as the Function Module That Waits**

You can also use GRAPH_DIALOG as the function module that waits for input from the user. This can simplify how you handle return parameters.

GRAPH_DIALOG doesn't call any graphics program of its own. This means that all messages received from graphics programs can be handled consistently. There is no need to distinguish between messages sent by the waiter's graphics program, and those sent by other graphics programs. An M_TYP (message type) parameter of 'I' is not possible when GRAPH_DIALOG is the waiter.

For a detailed discussion of how return parameters are handled, see Controlling the Interaction Using Dialog Parameters [Page 103].

**Programming Interactive Applications**

If your ABAP program allows the online user to open multiple windows or enter data with the mouse or keyboard, the application is interactive.

To control the interplay between online user and the ABAP program, you must use a special set of dialog parameters: STAT, INBUF, M_TYP, RBUFF, RWNID, PWDID, WINID, and SUPER. (The INBUF and RBUFF parameters are old and no longer needed. Do not use them for new programs.) For more information, see Dialog Programming in Releases Prior to 2.1 [Page 102].

Note that you should use the dialog programming mechanism with GRAPH_RECEIVE and GET_xxxx_PARAM in new programs.
Current Dialog Programming

As of Release 2.1, the interplay between the online user and the ABAP program is controlled by a set of dialog parameters and special function modules.

- Controlling the Interaction Using Parameters and Utility Function Modules [Page 109]
- Comparison with Waiting in Releases Prior to 2.1 [Page 113]
Controlling the Interaction Using Parameters and Utility Function Modules

To control the interplay between online user and the ABAP program, the graphics function modules offer the following mechanisms:

**Telling the function module what to do (parameter)**

- **STAT:** status with which function module called

**Receiving input from the user (utility function modules)**

- **GRAPH_RECEIVE:** waits for user input
- **GET_xxxx_PARAM:** returns the export parameters needed by the graphics program to respond to user input

**Opening and closing windows (parameters)**

- **PWDID:** parent window for the window being opened
- **WINID:** ID of current window (existing or to be created)
- **SUPER:** marks current window as "exit window"

STAT, the central parameter, tells the function module which task to perform on a given call:

- open a window and start the graphics program in it
- transfer data to the graphic window

The GRAPH_RECEIVE function module lets you wait for user input. The graphics program running in a given window reports back user activity (mouse clicks, menu selections, keyboard input, or window closure) as a message to GRAPH_RECEIVE. The ABAP program must then call the relevant GET_xxxx_PARAM module to activate the return parameter information for the given graphic program.

The most common pattern for programs allowing multiple windows is to place the various function module calls in a loop. A pass through the loop is made each time GRAPH_RECEIVE reports new input from the user. (See the example scheme in "Waiting for user input.")

Older ABAP programs that use previous waiting mechanisms will still run correctly. However, for new program development, use GRAPH_RECEIVE. Do not mix the old and the new waiting mechanisms in a single program.

**Waiting for User Input**

Every ABAP program that uses graphics function modules must call the GRAPH_RECEIVE utility function module to perform waiting. Even if your program does not respond to user input, waiting is important because it "realizes" actions requested with the STAT parameter:

- Newly-created windows do not appear on the screen until the call to GRAPH_RECEIVE takes place.
- Data sent to existing windows are not actually displayed until GRAPH_RECEIVE is called.
Controlling the Interaction Using Parameters and Utility Function Modules

Messages from the graphics program report user actions (keyboard and mouse input) that select graph objects, choose menu options, press functions keys, and so on.

GRAPH_RECEIVE receives messages from all graphics programs running on the user's workstation. The function module does not return until it has received an input message, even if the message only reports that a graphic window has been closed.

The ABAP program uses GRAPH_RECEIVE together with the GET_xxxx_PARAM function modules. These are a collection of function modules, one for each graphic program. (The letters xxxx stand for graphics program acronyms: BMAT, BUSG, GANTT, HIER, HPGL, PORT, or STAT).

GRAPH_RECEIVE is implicitly used by the SAP Network Graphics function modules CNET_GRAPHIC_NETWORK, CNET_GRAPHIC_HIERARCHY, and CNET_GRAPHIC_CLUSTER. These function modules are a combination of a number of function modules.

Parameters used

The following table contains a list of all possible parameters you can use with this function module.

For a detailed description of parameters and exceptions, see the R/3 system documentation (Tools → ABAP Workbench, Function library button, or transaction code SE37).

Parameters used with GRAPH_RECEIVE

<table>
<thead>
<tr>
<th>Name</th>
<th>Status</th>
<th>Initial Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACTIVATE_RAISE</td>
<td>Import</td>
<td>SPACE</td>
</tr>
<tr>
<td>ERRORCODE</td>
<td>Export</td>
<td></td>
</tr>
<tr>
<td>MCODE **</td>
<td>Export</td>
<td></td>
</tr>
<tr>
<td>RWNID **</td>
<td>Export</td>
<td></td>
</tr>
</tbody>
</table>

Note: ** denotes a dialog parameter

When GRAPH_RECEIVE reports that a message has been received, the ABAP program calls the relevant GET_xxxx_PARAM module to get the details on that message. For each graphics program, GRAPH_xxxx_PARAM returns the export parameters needed to respond to the user input.

To learn which graphics program sent the message received, the ABAP program must check the RWNID parameter returned by GRAPH_RECEIVE. The RWNID parameter provides the window ID for the window where the user input occurred. This window ID also identifies the graphics program running in the window. (For more information on window ID's, see the WINID parameter description.)

As in previous releases of SAP Graphics, it is possible to wait without using GRAPH_RECEIVE or the GET_xxxx_PARAM functions.

Function modules that actually call the graphics program can perform waiting and thus return their own export parameters.

(This requires using the RWNID and M_TYP parameter, and more values for STAT, when calling any graphics function module.)
This method is still available for compatibility with older programs, but is not recommended for new programs. Instead, use GRAPH_RECEIVE as described here. The function modules that call the graphics program should be viewed as "output-only" functions, while the GET_xxxx_PARAM functions serve as "input-only".

The following example is a possible scheme for combining multiple calls to function modules in a program. Here the programmer wanted to open several windows simultaneously, but not take input from the user until the last one is open. The windows created in the first calls do not appear on the screen until GRAPH_RECEIVE has been called. Similarly, newly-transferred data does not appear in the display until GRAPH_RECEIVE has been called.

```
CALL FUNCTION 'GRAPH_MATRIX_4D'
    EXPORTING STAT = '2'    "open window and send data
    WINID = 'BUSG01'
    CALL FUNCTION 'GRAPH_BUTTON_MENUE'
    EXPORTING STAT = '2'    "open-window and send data
    WINID = 'BMAT03'

CALL FUNCTION 'GRAPH_RECEIVE'
    IMPORTING MCODE = MCODE
    RWNID = RWNID.

WHILE MCODE NE 'D'.
    CASE RWNID(4).
      WHEN 'BUSG'.
        IF MCODE EQ 'Q'......  ENDIF."window closed?
      ELSE.
        CALL FUNCTION 'GET_BUSG_PARAM'
        IMPORTING....
      .....)
      CALL FUNCTION 'GRAPH_MATRIX_4D'
        EXPORTING STAT = '5'    "send new data
        WINID = 'BUSG01'
        .....)
    ENDIF.

WHEN 'BMAT'.
    IF MCODE EQ 'Q'......  ENDIF."window closed?
    ELSE.
```

April 2001  111
Controlling the Interaction Using Parameters and Utility Function Modules

CALL FUNCTION 'GET_BMAT_PARAM'
  IMPORTING....

.....

CALL FUNCTION 'GRAPH_BUTTON_MENUE'
  EXPORTING STAT = '5'  "send new data
  WINID = 'BMAT03'

.....

ENDIF.

CALL FUNCTION 'GRAPH_RECEIVE'
  IMPORTING MCODE = MCODE
    RWNID = RWNID.

ENDWHILE.
Comparison with Waiting in Releases Prior to 2.1

GRAPH_RECEIVE offers a simpler waiting method than was possible with prior releases:

- Waiting with GRAPH_RECEIVE (rather than one of the graphics-program-related function modules) is cleaner. GRAPH_RECEIVE reports all messages received in the same way, regardless of which graphics program sent it. As a result, the ABAP program can handle them all consistently.

- For calling any graphics function module, the STAT parameter now only has two meaningful values:
  - ‘2’: open new window and send data
  - ‘5’: send data to an open window

- The RWNID and M_TYP parameters are no longer necessary when calling any graphics function modules except GRAPH_RECEIVE. (Also, for GRAPH_RECEIVE, M_TYP has been re-named M_CODE.)

💡

Older ABAP programs that use previous waiting mechanisms will still run correctly. However, for new program development, use GRAPH_RECEIVE. Do not mix the old and the new waiting mechanisms in a single program.
Sample Programs

The following sample programs are available in this guide. For further sample programs calling SAP Graphics applications, use transaction code *gral* in the R/3 System.

- **Calling SAP Business Graphics, Without Dialog [Page 115]**
- **Calling SAP Pushbutton, Statistical and Business Graphics, With Dialog [Page 186]**
- **Calling SAP Pushbutton and Business Graphics, With Dialog, Using GRAPH_RECEIVE and GET_xxxx_PARAM [Page 117]**
- **Calling SAP Hierarchy Graphics and Others, With Dialog [Page 138]**
- **Calling SAP Statistical Graphics, Without Dialog [Page 165]**
- **Calling SAP Gantt Chart Graphics, With Dialog [Page 169]**
- **Calling SAP HPGL Display [Page 189]**
- **Calling SAP Portfolio Graphics, With Dialog [Page 190]**
- **Calling SAP Network Graphics, Without Dialog [Page 200]**
- **Calling SAP Bar Chart Graphics, Without Dialog [Page 206]**
- **Calling SAP Structural Graphics [Page 217]**
Calling SAP Business Graphics, Without Dialog

The following ABAP program illustrates some of the function module calls discussed in the preceding topics.

PROGRAM G_BUSG_1.

*------------------------------------------------------------*
  * This program illustrates a call to SAP Business Graphics.  *
  * Each row in the DATA table contains one text label and a  *
  * single numeric value.                                      *
*------------------------------------------------------------*

DATA: BEGIN OF DATA  OCCURS 1,
  TEXT(25),
  VALUE TYPE P,
END OF DATA.

DATA: TCOL1(5) VALUE '#1991'.
DATA: TITLE(25) VALUE 'Sales'.
SET PF-STATUS 'GRAF'.

DATA-TEXT = 'Product_1'.
DATA-VALUE = 153470.
APPEND DATA.
DATA-TEXT = 'Product_2'.
DATA-VALUE = 253150.
APPEND DATA.
DATA-TEXT = 'Product_3'.
DATA-VALUE = 53470.
APPEND DATA.
DATA-TEXT = 'Product_4'.
DATA-VALUE = 182000.
APPEND DATA.
DATA-TEXT = 'Product_5'.

April 2001  115
Calling SAP Business Graphics, Without Dialog

DATA-VALUE = 92410.
APPEND DATA.
WRITE: / 'Products',22 TCOL1.
SKIP 2.
LOOP AT DATA.
  WRITE: / DATA-TEXT, 12 DATA-VALUE.
ENDLOOP.

*------------------------------------------------------------*
* PF11:  Put out a 2D graphic                                *
*------------------------------------------------------------*
AT PF11.
CALL FUNCTION 'GRAPH_2D' EXPORTING TITL = TITLE
  TABLES    DATA = DATA.
Calling SAP Pushbutton, Statistical and Business Graphics, With Dialog

The following ABAP program illustrates some of the function module calls discussed in the preceding topics.

PROGRAM G_BMAT_2.

*--------------------------------------------------------*
*                                                        *
* This report calls SAP Pushbutton Graphics.              *
*                                                        *
* When the user clicks on a push button, a menu first     *
* appears in which additional information can be          *
* requested. These requests cause SAP Statistics or        *
* SAP Business Graphics windows to open.                  *
*                                                        *
* If the user selects a whole row or column (by clicking   *
* on a row or column header), a 2D graph or single        *
* curve is displayed. Clicking on the button in the       *
* upper left corner causes a 3D graph or group            *
* of curves to be displayed.                             *
*--------------------------------------------------------*

DATA: BEGIN OF MATRIX OCCURS 1,                    "// Data for BMAT
          COL1(20),
          COL2 TYPE P,
          COL3 TYPE P,
          COL4 TYPE P,
          COL5 TYPE P,
    END OF MATRIX.

DATA: BEGIN OF ATTR OCCURS 1,                       "// ATTR table for BMAT
          COL1(20),
          COL2 TYPE P,
          COL3 TYPE P,
Calling SAP Pushbutton, Statistical and Business Graphics, With Dialog

```
COL4 TYPE P,
COL5 TYPE P,
END OF ATTR.

DATA: BEGIN OF SIZE OCCURS 1, "// SIZE table for BMAT
       C(10) TYPE C,
END OF SIZE.

DATA: BEGIN OF MENUE OCCURS 1, "// MENU table for BMAT
       TEXT(40),
       TEXT2(20),
END OF MENUE.

DATA: BEGIN OF SDATA OCCURS 1, "// Data for STAT
       X TYPE P,
       Y TYPE P,
END OF SDATA.

DATA: BEGIN OF OPTS OCCURS 1, "// OPTS table for STAT
       "// and BUSG
       C(80) TYPE C,
END OF OPTS.

DATA: BEGIN OF TCOL OCCURS 1, "// TCOL table for BUSG
       C(20) TYPE C,
END OF TCOL.

DATA: RWNID(10), M_TYP(1).
DATA: BUSG_STAT, STAT_STAT, BMAT_STAT.
DATA STAT_ART(3).
DATA IND LIKE SY-INDEX.
DATA: NROW(2), NCOL(2).
DATA: MENROW(2).
DATA: MENCOL(2).

MENUE-TEXT = 'Business Graphics'. "// Fill BMAT popup
MENUE-TEXT2 = 'Business Graphics'.
APPEND MENUE.
MENUE-TEXT = 'Statistics Graphics'.
```
MENUE-TEXT2 = '(logarithmic)'.
APPEND MENUE.

MATRIX-COL1 = 'Row 1'.  "// Fill BMAT Data
MATRIX-COL2 = 11.
MATRIX-COL3 = 12.
MATRIX-COL4 = 13.
MATRIX-COL5 = 14.
APPEND MATRIX.
MOVE-CORRESPONDING MATRIX TO ATTR.
ATTR-COL1 = 0.
APPEND ATTR.

MATRIX-COL1 = 'Row 2'.
MATRIX-COL2 = 21.
MATRIX-COL3 = 22.
MATRIX-COL4 = 23.
MATRIX-COL5 = 24.
APPEND MATRIX.
MOVE-CORRESPONDING MATRIX TO ATTR.
ATTR-COL1 = 0.
APPEND ATTR.

MATRIX-COL1 = 'Sub-total'.
MATRIX-COL2 = 32.
MATRIX-COL3 = 34.
MATRIX-COL4 = 36.
MATRIX-COL5 = 38.
APPEND MATRIX.
MOVE-CORRESPONDING MATRIX TO ATTR.
ATTR-COL1 = 0.
APPEND ATTR.

MATRIX-COL1 = 'Row 4'.
MATRIX-COL2 = 41.
MATRIX-COL3 = 42.
MATRIX-COL4 = 43.
MATRIX-COL5 = 44.
Calling SAP Pushbutton, Statistical and Business Graphics, With Dialog

APPEND MATRIX.
MOVE-CORRESPONDING MATRIX TO ATTR.
ATTR-COL1 = 0.
APPEND ATTR.
MATRX-COL1 = 'T O T A L'.
MATRX-COL2 = 73.
MATRX-COL3 = 76.
MATRX-COL4 = 79.
MATRX-COL5 = 82.
APPEND MATRX.
MOVE-CORRESPONDING MATRIX TO ATTR.
ATTR-COL1 = 0.
APPEND ATTR.

WRITE: //, 'Label', 22 'Column 1', 37 'Column 2',
         52 'Column 3', 67 'Column 4', //.
LOOP AT MATRX.
  WRITE: // (15) MATRX-COL1, (14) MATRX-COL2, (14) MATRX-COL3,
         (14) MATRX-COL4, (14) MATRX-COL5.
ENDLOOP.

WRITE: //, //, 'Function keys:'.
WRITE: //, 'F10 starts SAP Pushbutton Graphics'.
WRITE: //, 'F3 ends SAP Pushbutton Graphics'.

AT PF10.

TCOL = 'Column 1'. APPEND TCOL.       "// fill TCOL for BUSG
TCOL = 'Column 2'. APPEND TCOL.
TCOL = 'Column 3'. APPEND TCOL.
TCOL = 'Column 4'. APPEND TCOL.
SIZE = 'C  1 30'. APPEND SIZE.
SIZE = 'R  1 2 '. APPEND SIZE.

*------------------------------------------------------------*
Calling SAP Pushbutton, Statistical and Business Graphics, With Dialog

* Dialog Loop                                      *
*------------------------------------------------------------*

BUSG_STAT = '2'.  "// start and do not wait
STAT_STAT = '2'.  "// start and do not wait
CLEAR BMAT_STAT.  "// start and wait

DO.

CALL FUNCTION 'GRAPH_BUTTON_MENUE'
  "// Pushbutton pad with menu

EXPORTING
  TTEXT = SY-TITLE      "// Title
  CTXT1 = 'Label'       "// 5 columns
  CTXT2 = 'Column 1'
  CTXT3 = 'Column 2'
  CTXT4 = 'Column 3'
  CTXT5 = 'Column 4'
  VTXT1 = 'Display Criterion 1'
  VALD1 = 1
  VAL11 = 10
  VAL12 = 30
  VAL13 = 40
  VAL14 = 50
  COL11 = 'CYAN'
  COL12 = 'GRAY'
  COL13 = 'YELLOW'
  COL14 = 'GRAY'
  COL15 = 'RED'
  MTITL = 'Menu'        "// Menu title
  STAT  = BMAT_STAT     "// Dialog parameters
  SUPER = 'X'           "// Closing 'BMAT' will
                          // close all other windows
  WINID = 'BMAT'        "// Window Id

IMPORTING
  ROWNR = NROW          "// Selected row
  COLNR = NCOL          "// Selected column
Calling SAP Pushbutton, Statistical and Business Graphics, With Dialog

MROWN = MENROW        "// Selected menu-row
MCOLN = MENCOL        "// Selected menu-column
M_TYP = M_TYP         "// Message-type
RWNID = RWNID         "// Id of sending window

TABLES
DATA  = MATRIX        "// Defines the matrix
MENU  = MENUE         "// Defines the menu
ATTR  = ATTR          "// Defines colors etc.
SIZE  = SIZE.         "// Defines sizes

BMAT_STAT = '4'.      "// Next time, just wait

*------------------------------------------------------------*
* Check the message and react properly                       *
*------------------------------------------------------------*
*                                                            *
* M_TYP : 'D' : Last window has been closed (none left open) *
*         'I' : Message from BMAT                            *
*         '0' : Message from some other (non-BMAT) window     *
*                                                            *
*-----------------------------------------------------------*

CASE M_TYP.
  WHEN 'D'.                     "// BMAT was closed
    EXIT.
  WHEN '0'.                     "// Message from non-BMAT
    IF RWNID(4) EQ 'BUSG'.      "// If it's a BUSG window,
      BUSG_STAT = '2'.          "// it must be a 'Q'
    ENDDIF.
    IF RWNID(4) EQ 'STAT'.      "// If it's a STAT window,
      STAT_STAT = '2'.          "// Restart STAT next time
    ENDDIF.
WHEN 'Q'.                      "// Should not happen when
     "// using SUPER.
WHEN 'I'.                      "// Message from BMAT
     "// Reaction depends on
     "// selected item:
CASE MENROW.
     WHEN 1.                   "// Start or reload BUSG
         REFRESH OPTS.
         IF NROW(1) EQ '1' AND NCOL(1) EQ '0'.
             "// Upleft item --> 3D BUSG
             CALL FUNCTION 'GRAPH_3D'
             EXPORTING
                TITL = 'Overview'
                DIM1 = 'Columns'
                DIM2 = 'Rows'
                STAT = BUSG_STAT
             TABLES
                DATA = MATRIX.
             BUSG_STAT = '5'.          "// Reload next time
         ENDIF.
         IF NROW(1) EQ '1' AND NCOL(1) NE '0'.
             "// Col-header --> 2D BUSG
             READ TABLE TCOL INDEX NCOL.
             CALL FUNCTION 'GRAPH_MATRIX_2D'
             EXPORTING
                TITL = TCOL
                NCOL = NCOL
                STAT = BUSG_STAT
             TABLES
                DATA = MATRIX
                OPTS = OPTS
                TCOL = TCOL.
             BUSG_STAT = '5'.          "// Reload next time
         ENDIF.
         IF NROW(1) NE '1' AND NCOL(1) EQ '0'.
             "// Col-header --> 2D BUSG
             READ TABLE TCOL INDEX NCOL.
             CALL FUNCTION 'GRAPH_MATRIX_2D'
             EXPORTING
                TITL = TCOL
                NCOL = NCOL
                STAT = BUSG_STAT
             TABLES
                DATA = MATRIX
                OPTS = OPTS
                TCOL = TCOL.
             BUSG_STAT = '5'.          "// Reload next time
         ENDIF.
Calling SAP Pushbutton, Statistical and Business Graphics, With Dialog

"// Row-header -> 2D
"// BUSG

NROW = NROW - 1.
READ TABLE MATRIX INDEX NROW.
CALL FUNCTION 'GRAPH_MATRIX_2D'
EXPORTING
  TITL = MATRIX-COL1
  NROW = NROW
  STAT = BUSG_STAT
TABLES
  DATA = MATRIX
  OPTS = OPTS
  TCOL = TCOL.
BUSG_STAT = '5'.  "// Reload next time
ENDIF.
WHEN 2.  "// Start or reload STAT
  IF MENCOL(1) EQ '1'.
    STAT_ART = 'LOG'.
  ELSE.
    STAT_ART = 'LIN'.
  ENDIF.

REFRESH SDATA.
REFRESH OPTS.
IF NROW(1) EQ '1' AND NCOL(1) EQ '0'.
  "// Upleft item -> 4
  "// curves
DO 4 TIMES.
  IND = SY-INDEX.
DO 5 TIMES.
  READ TABLE MATRIX INDEX SY-INDEX.
  SDATA-X = SY-INDEX.
  CASE IND.
    WHEN 1.
      SDATA-Y = MATRIX-COL2.
Calling SAP Pushbutton, Statistical and Business Graphics, With Dialog

WHEN 2.
    SDATA-Y = MATRIX-COL3.
WHEN 3.
    SDATA-Y = MATRIX-COL4.
WHEN 4.
    SDATA-Y = MATRIX-COL5.
ENDCASE.
APPEND SDATA.
ENDDO.

OPTS = '$5'.
APPEND OPTS.
OPTS = 'COLOR=7'.
APPEND OPTS.
OPTS = 'THICK=1'.
APPEND OPTS.
ENDDO.

CALL FUNCTION 'STAT_GRAPH'
    EXPORTING
        TTEXT = 'Overview'
        STAT = STAT_STAT
        Y_ART = STAT_ART
        NOPOS = 'X'
    TABLES
        OPTS = OPTS
        DATA = SDATA.
    STAT_STAT = '5'.  // Reload next time
ENDIF.

IF NROW(1) EQ '1' AND NCOL(1) NE '0'.
    "// Col-header -> 1
    "// curve
DO 5 TIMES.
    READ TABLE MATRIX INDEX SY-INDEX.
SDATA-X = SY-INDEX.
CASE NCOL.
WHEN 1.
   SDATA-Y = MATRIX-COL2.
WHEN 2.
   SDATA-Y = MATRIX-COL3.
WHEN 3.
   SDATA-Y = MATRIX-COL4.
WHEN 4.
   SDATA-Y = MATRIX-COL5.
ENDCASE.
APPEND SDATA.
ENDDO.
OPTS = '$5'.
APPEND OPTS.
OPTS = 'COLOR=7'.
APPEND OPTS.
OPTS = 'THICK=1'.
APPEND OPTS.

READ TABLE TCOL INDEX NCOL.
CALL FUNCTION 'STAT_GRAPH'
   EXPORTING
      TTEXT = TCOL
      STAT  = STAT_STAT
      Y_ART = STAT_ART
      NOPOS = 'X'
   TABLES
      OPTS  = OPTS
      DATA  = SDATA.
   STAT_STAT = '5'.          // Reload next time
ENDIF.

IF NROW(1) NE '1' AND NCOL(1) EQ '0'.
   // Row-header -> 1
NROW = NROW - 1.
READ TABLE MATRIX INDEX NROW.
SDATA-X = 1.
SDATA-Y = MATRIX-COL2.
APPEND SDATA.
SDATA-X = 2.
SDATA-Y = MATRIX-COL3.
APPEND SDATA.
SDATA-X = 3.
SDATA-Y = MATRIX-COL4.
APPEND SDATA.
SDATA-X = 4.
SDATA-Y = MATRIX-COL5.
APPEND SDATA.

OPTS = '$4'.
APPEND OPTS.
OPTS = 'COLOR=7'.
APPEND OPTS.
OPTS = 'THICK=1'.
APPEND OPTS.

CALL FUNCTION 'STAT_GRAPH'
  EXPORTING
    TTEXT = MATRIX-COL1
    STAT = STAT_STAT
    Y_ART = STAT_ART
    NOPOS = 'X'
  TABLES
    OPTS = OPTS
    DATA = SDATA.
    STAT_STAT = '5'. // Reload next time
ENDIF.
ENDCASE.
Calling SAP Pushbutton, Statistical and Business Graphics, With Dialog

   ENDCASE .

   ENDDO .
Calling SAP Pushbutton and Business Graphics, With Dialog, Using GRAPH_RECEIVE and GET_xxxx_PARAM

The following ABAP program illustrates some of the calls to graphics function modules and the utility function modules GRAPH_RECEIVE and GET_xxxx_PARAM discussed in the preceding topics.

*-------------------------------------------------------------*
* This program calls SAP Pushbutton Graphics.                *
* Selecting a pushbutton calls a BUSG window. Selecting a    *
* pushbutton again does not call a new BUSG window but       *
* overwrites the data in the current BUSG window.             *
* If BMAT is closed, the BUSG window also disappears. This is *
* due to the SUPER parameter.                                *
*-------------------------------------------------------------*

PROGRAM GRBMAT01.

* GL 5.7.1993 : NEW DIALOG LOOP
DATA: MCODE, B_TYP.
DATA: OPEN_LOAD VALUE '2'.
DATA: RELOAD VALUE '5'.

DATA: BEGIN OF MATRIX OCCURS 1,
      COL1(20),
      COL2 TYPE P,
      COL3 TYPE P,
      COL4 TYPE P,
      COL5 TYPE P,
END OF MATRIX.

DATA: BEGIN OF ATTR OCCURS 1,
      COL1(20),
      COL2 TYPE P,
      COL3 TYPE P,
      COL4 TYPE P,
      COL5 TYPE P,
END OF ATTR.
Calling SAP Pushbutton and Business Graphics, With Dialog, Using GRAPH_RECEIVE and GET_xxxx_PARAM

DATA: BEGIN OF SIZE OCCURS 1,
    C(10) TYPE C,
  END OF SIZE.
DATA: BEGIN OF MTEXT OCCURS 1,
    TEXT(40),
  END OF MTEXT.
DATA: BEGIN OF OPTS OCCURS 1,
    C(80) TYPE C,
  END OF OPTS.
DATA: BEGIN OF TCOL OCCURS 1,
    C(20) TYPE C,
  END OF TCOL.
DATA TEXT(64).
DATA WINID(10).
DATA INDEX LIKE SY-INDEX.
DATA BUSG_STAT.
DATA BMAT_STAT.
DATA TWIN_STAT.
DATA: NROW(9),
    NCOL(1).
FIELD-SYMBOLS: <F>.
SET PF-STATUS 'GRAF'.

MATRIX-COL1 = 'Line 1'.
MATRIX-COL2 = 11.
MATRIX-COL3 = 12.
MATRIX-COL4 = 13.
MATRIX-COL5 = 14.
APPEND MATRIX.
MOVE-CORRESPONDING MATRIX TO ATTR.
ATTR-COL1 = 0.
APPEND ATTR.
MATRIX-COL1 = 'Line 2'.
MATRIX-COL2 = 21.
MATRIX-COL3 = 22.
Calling SAP Pushbutton and Business Graphics, With Dialog, Using GRAPH_RECEIVE and GET_xxxx_

MATRIX-COL4 = 23.
MATRIX-COL5 = 24.
APPEND MATRIX.
MOVE-CORRESPONDING MATRIX TO ATTR.
ATTR-COL1 = 0.
APPEND ATTR.
MATRIX-COL1 = 'Subtotal'.
MATRIX-COL2 = 32.
MATRIX-COL3 = 34.
MATRIX-COL4 = 36.
MATRIX-COL5 = 38.
APPEND MATRIX.
MOVE-CORRESPONDING MATRIX TO ATTR.
ATTR-COL1 = 0.
APPEND ATTR.
MATRIX-COL1 = 'Line 4'.
MATRIX-COL2 = 41.
MATRIX-COL3 = 42.
MATRIX-COL4 = 43.
MATRIX-COL5 = 44.
APPEND MATRIX.
MOVE-CORRESPONDING MATRIX TO ATTR.
ATTR-COL1 = 0.
APPEND ATTR.
MATRIX-COL1 = 'Sum total'.
MATRIX-COL2 = 73.
MATRIX-COL3 = 76.
MATRIX-COL4 = 79.
MATRIX-COL5 = 82.
APPEND MATRIX.
MOVE-CORRESPONDING MATRIX TO ATTR.
ATTR-COL1 = 0.
APPEND ATTR.

WRITE: // 'Designation',22 'Column 1',37 'Column 2',
Calling SAP Pushbutton and Business Graphics, With Dialog, Using GRAPH_RECEIVE and GET_xxxx_PARAM

LOOP AT MATRIX.
    WRITE: / (15) MATRIX-COL1,
          (14) MATRIX-COL2, (14) MATRIX-COL3, (14) MATRIX-COL4, (14) MATRIX-COL5.
ENDLOOP.
TCOL = 'Column 1'. APPEND TCOL.
TCOL = 'Column 2'. APPEND TCOL.
TCOL = 'Column 3'. APPEND TCOL.
TCOL = 'Column 4'. APPEND TCOL.
SIZE = 'C 1 30'. APPEND SIZE.
SIZE = 'R 1 2'. APPEND SIZE.

AT PF11.
    BUSG_STAT = OPEN_LOAD.
    TWIN_STAT = OPEN_LOAD.
    BMAT_STAT = OPEN_LOAD.
    CALL FUNCTION 'GRAPH_BUTTON_MATRIX' "// pushbutton pad
    EXPORTING
        STAT = BMAT_STAT "// open window and transfer data
        WINID = 'BMAT01'
        TTEXT = SY-TITLE
        CTXT1 = 'Designation'
        CTXT2 = 'Column 1'
        CTXT3 = 'Column 2'
        CTXT4 = 'Column 3'
        CTXT5 = 'Column 4'
        VTXT1 = 'ID number 1'
        VALD1 = 1
        VAL11 = 10
        VAL12 = 30
        VAL13 = 40
        VAL14 = 50
        COL11 = 'CYAN'
        COL12 = 'GRAY'
        COL13 = 'YELLOW'
CALL FUNCTION 'GRAPH_RECEIVE'
IMPORTING
RWNID = WINID
MCODE = MCODE.

* Possible evaluation strategies
* ===============================
* 1) IF MCODE = 'D'. EXIT. ENDIF.
* CASE RWNID.
* WHEN 'win1'.
* IF MCODE = 'Q'.
* "// close code
* ELSE. ('I')
* "// action code
* ENDF.
* WHEN 'win2'.
* ...
* ENDCASE.
* ...
* ENDDO.
*
* 1a) IF MCODE = 'D'. EXIT. ENDIF.
* MOVE formname(MCODE, RWNID) TO FORMNAME
* "// Table ( I/Q, WINID, formname)
* PERFORM FORMNAME.
*
Calling SAP Pushbutton and Business Graphics, With Dialog, Using GRAPH_RECEIVE and GET_xxxx_PARAM

* 2) CASE MCODE.
  * WHEN 'D'.
  * EXIT.
  * WHEN 'Q'.
  * CASE RWNID.
  * WHEN 'win1'.
  * "// close code 'win1'.
  * WHEN 'win2'.
  * "// close code 'win2'.
  * ...
  * ENDCASE.
  * WHEN 'I'.
  * CASE RWNID.
  * WHEN 'win1'.
  * "// action code 'win1'.
  * WHEN 'win2'.
  * "// action code 'win2'.
  * ...
  * ENDCASE.
  *
  * ENDCASE.
 *
  * ENDDO.

IF MCODE = 'D'. EXIT. ENDIF.

CASE WINID(4).
  WHEN 'TEMA'. "// subchild was closed
    "// mcode MUST be Q
    TWIN_STAT = OPEN_LOAD. "// restart next time
  WHEN 'BUSG'. "// subchild was closed
    "// mcode MUST be Q
    BUSG_STAT = OPEN_LOAD. "// restart next time
  WHEN 'BMAT'. "// Message from BMAT
    IF MCODE = 'Q'.

134  April 2001
CALL FUNCTION 'GET_BMAT_PARAM'
IMPORTING
  B_TYP = B_TYP
  ROWNR = NROW
  COLNR = NCOL.

IF NROW(1) EQ '1' AND NCOL(1) EQ '0'.  " 3-D graph
  CALL FUNCTION 'GRAPH_3D'
  EXPORTING
    TITL = TEXT
    DIM1 = 'Columns'
    DIM2 = 'Lines'
    STAT = BUSG_STAT
    TABLES
      DATA = MATRIX.
      BUSG_STAT = RELOAD.  " reload data
ENDIF.

IF NROW(1) EQ '1' AND NCOL(1) NE '0'.  " 2-D graph col.
  READ TABLE TCOL INDEX NCOL.
  CALL FUNCTION 'GRAPH_MATRIX_2D'
  EXPORTING
    TITL = TCOL
    NCOL = NCOL
    STAT = BUSG_STAT
    TABLES
      DATA = MATRIX
      OPTS = OPTS
      TCOL = TCOL.
      BUSG_STAT = RELOAD.  " reload data
ENDIF.

IF NROW(1) NE '1' AND NCOL(1) EQ '0'.  " 2-D graph line
Calling SAP Pushbutton and Business Graphics, With Dialog, Using GRAPH_RECEIVE
and GET_xxxx_PARAM

NROW = NROW - 1.
READ TABLE MATRIX INDEX NROW.
CALL FUNCTION 'GRAPH_MATRIX_2D'
  EXPORTING
    TITL = MATRIX-COL1
    NROW = NROW
    STAT = BUSG_STAT
  TABLES
    DATA = MATRIX
    OPTS = OPTS
    TCOL = TCOL.
    BUSG_STAT = RELOAD. " reload data
ENDIF.
IF NROW(1) GT '1' AND NCOL(1) GT '0'. " display cells text
  NROW = NROW - 1.
  NCOL = NCOL + 1.
  PERFORM GET_TEXT TABLES MTEXT USING NROW NCOL.
  MOVE 'Info für Element ' TO TEXT.
  WRITE NROW TO TEXT+17(2).
  WRITE NCOL TO TEXT+19(2).
  CALL FUNCTION 'TEXT_MATRIX'
    EXPORTING
      TTEXT = TEXT
      RSIZE = '5'
      CSIZE = '20'
      STAT = TWIN_STAT
    TABLES
      DATA = MTEXT.
      TWIN_STAT = RELOAD. " reload data
ENDIF.
ENDIF. " WINID 'BMAT', MCODE 'I'
WHEN OTHERS. " unknown Window
ENDCASE. " WINID
ENDDO.
FORM GET_TEXT TABLES MTEXT USING NROW NCOL.

READ TABLE MATRIX INDEX NROW.

DO.

ASSIGN COMPONENT SY-INDEX OF STRUCTURE MATRIX TO <F>.

IF SY-SUBRC NE 0. EXIT. ENDIF.

IF SY-INDEX EQ NCOL. EXIT. ENDIF.

ENDDO.

CLEAR MTEXT.

* APPEND MTEXT.

MOVE ' Previous year : ' TO MTEXT.

<F> = <F> - 1.

WRITE <F> TO MTEXT+12(5).

APPEND MTEXT.

MOVE ' Current year : ' TO MTEXT.

<F> = <F> + 1.

WRITE <F> TO MTEXT+12(5).

APPEND MTEXT.

MOVE ' Following year : ' TO MTEXT.

<F> = <F> + 1.

WRITE <F> TO MTEXT+12(5).

APPEND MTEXT.

ENDFORM.
Calling SAP Hierarchy Graphics and Others, With Dialog

The following ABAP program illustrates some of the function module calls discussed in the preceding topics.

PROGRAM G_HIER_2.

*-----------------------------------------------------------*
*                                                           *
*-----------------------------------------------------------*
* This Report starts the SAP Hierarchy system.              *
*                                                           *
* It demonstrates several dialogs, depending on which PF key* *
* is pressed. Each dialog starts one or more SAP Graphics    *
* programs. Possible dialogs are:                           *
*   PF10: Start SAP Hierarchy: Node selection opens text     *
*           window                                          *
*   PF11: Start SAP Hierarchy: Node selection opens 2nd HIER* *
*           window that displays a sub-tree of the hierarchy.* *
*           Node selection in the 2nd HIER window has no      *
*           effect.                                         *
*   PF12: Start SAP Hierarchy: Node selection opens a window*
*           for SAP Business Graphics.                      *
*   PF13: Start SAP Hierarchy: Node selection opens a window*
*           for SAP Statistics. Selection of a curve point in *
*           the statistics window opens a text window.       *
*   PF14: Start SAP Hierarchy: Node selection opens four     *
*           different windows:                             *
*           1. SAP Hierarchy, showing a sub-tree of the      *
*              hierarchy                                    *
*           2. SAP Statistics                              *
*           3. SAP Business Graphics                       *
*           4. SAP Text Matrix                             *
*           Selection of a curve point in the statistics    *
*           window opens a text window.                     *
* The description of each dialog is found with the code for *
DATA: BEGIN OF HIER OCCURS 1,
   ELKEY(11),
   UEKEY(11),
   ETEXT(20),
   VAL1 TYPE P,
   VAL2 TYPE P,
   END OF HIER.

DATA: BEGIN OF PHIER OCCURS 1,
   ELKEY(11),
   UEKEY(11),
   ETEXT(20),
   END OF PHIER.

DATA: BEGIN OF BUSGDATA OCCURS 1,
   ELKEY(11),
   VAL1 TYPE P,
   VAL2 TYPE P,
   VAL3 TYPE P,
   VAL4 TYPE P,
   END OF BUSGDATA.

DATA: BEGIN OF MTEXT OCCURS 1,
   TEXT(100),
   END OF MTEXT.

DATA: RBUFF(64), M_TYP(1), RWNID(5), STAT_TYP.
DATA: INDEX LIKE SY-INDEX.
DATA: HIER1_STAT, HIER2_STAT.
DATA: TEMA_STAT, TEMA1_STAT.
DATA: STAT_STAT, BUSG_STAT..
DATA: NODE(64), KURVE(5), PUNKT(5).

DATA: BEGIN OF TCOL OCCURS 1,
   TEXT(20),
Calling SAP Hierarchy Graphics and Others, With Dialog

END OF TCOL,
BEGIN OF GOPTS OCCURS 1,
C(20),
END OF GOPTS.

SET PF-STATUS 'GRAF'.

HIER-UEKEY = SPACE.  "// Fill the data
HIER-ELKEY = '1000000001'.
HIER-ETEXT = 'El.1 - S1'.
HIER-VAL1  = 10.
HIER-VAL2  = 15.
APPEND HIER.
BUSGDATA-ELKEY = HIER-ELKEY.
BUSGDATA-VAL1 = 1234.
BUSGDATA-VAL2 = 2345.
BUSGDATA-VAL3 = 3456.
BUSGDATA-VAL4 = 4567.
APPEND BUSGDATA.

HIER-UEKEY = '1000000001'.
HIER-ELKEY = '1100000002'.
HIER-ETEXT = 'El.1 - S2'.
HIER-VAL1  = 20.
HIER-VAL2  = 25.
APPEND HIER.
BUSGDATA-ELKEY = HIER-ELKEY.
BUSGDATA-VAL1 = 3534.
BUSGDATA-VAL2 = 7845.
BUSGDATA-VAL3 = 1256.
BUSGDATA-VAL4 = 3467.
APPEND BUSGDATA.

HIER-UEKEY = '1000000001'.
HIER-ELKEY = '1200000002'.

140 April 2001
HIER-ETEXT = 'El.2 - S2'.
HIER-VAL1  = 30.
HIER-VAL2  = 35.
APPEND HIER.
BUSGDATA-ELKEY = HIER-ELKEY.
BUSGDATA-VAL1 = 1734.
BUSGDATA-VAL2 = 2345.
BUSGDATA-VAL3 = 4256.
BUSGDATA-VAL4 = 3767.
APPEND BUSGDATA.

HIER-UEKEY = '10000000001'.
HIER-ELKEY = '13000000002'.
HIER-ETEXT = 'El.3 - S2'.
HIER-VAL1  = 40.
HIER-VAL2  = 45.
APPEND HIER.
BUSGDATA-ELKEY = HIER-ELKEY.
BUSGDATA-VAL1 = 8734.
BUSGDATA-VAL2 = 1345.
BUSGDATA-VAL3 = 5256.
BUSGDATA-VAL4 = 3767.
APPEND BUSGDATA.

HIER-UEKEY = '10000000001'.
HIER-ELKEY = '14000000002'.
HIER-ETEXT = 'El.4 - S2'.
HIER-VAL1  = 50.
HIER-VAL2  = 55.
APPEND HIER.
BUSGDATA-ELKEY = HIER-ELKEY.
BUSGDATA-VAL1 = 5890.
BUSGDATA-VAL2 = 3245.
BUSGDATA-VAL3 = 1786.
BUSGDATA-VAL4 = 3647.
Calling SAP Hierarchy Graphics and Others, With Dialog

APPEND BUSGDATA.

HIER-UEKEY = '11000000002'.
HIER-ELKEY = '11100000003'.
HIER-ETEXT = 'El.11 - S3'.
HIER-VAL1 = 60.
HIER-VAL2 = 65.
APPEND HIER.

BUSGDATA-ELKEY = HIER-ELKEY.
BUSGDATA-VAL1 = 2370.
BUSGDATA-VAL2 = 2845.
BUSGDATA-VAL3 = 1286.
BUSGDATA-VAL4 = 7311.
APPEND BUSGDATA.

HIER-UEKEY = '11000000002'.
HIER-ELKEY = '11200000003'.
HIER-ETEXT = 'El.12 - S3'.
HIER-VAL1 = 70.
HIER-VAL2 = 75.
APPEND HIER.

BUSGDATA-ELKEY = HIER-ELKEY.
BUSGDATA-VAL1 = 1210.
BUSGDATA-VAL2 = 2165.
BUSGDATA-VAL3 = 3290.
BUSGDATA-VAL4 = 1311.
APPEND BUSGDATA.

HIER-UEKEY = '11000000002'.
HIER-ELKEY = '11300000003'.
HIER-ETEXT = 'El.13 - S3'.
HIER-VAL1 = 80.
HIER-VAL2 = 85.
APPEND HIER.

BUSGDATA-ELKEY = HIER-ELKEY.
BUSGDATA-VAL1 = 4578.
BUSGDATA-VAL2 = 2789.
BUSGDATA-VAL3 = 5412.
BUSGDATA-VAL4 = 4678.
APPEND BUSGDATA.

HIER-UEKEY = '12000000002'.
HIER-ELKEY = '12100000003'.
HIER-ETEXT = 'El.21 - S3'.
HIER-VAL1 = 90.
HIER-VAL2 = 95.
APPEND HIER.

BUSGDATA-ELKEY = HIER-ELKEY.
BUSGDATA-VAL1 = 4782.
BUSGDATA-VAL2 = 3571.
BUSGDATA-VAL3 = 1672.
BUSGDATA-VAL4 = 2690.
APPEND BUSGDATA.

HIER-UEKEY = '12000000002'.
HIER-ELKEY = '12200000003'.
HIER-ETEXT = 'El.22 - S3'.
HIER-VAL1 = 18.
HIER-VAL2 = 24.
APPEND HIER.

BUSGDATA-ELKEY = HIER-ELKEY.
BUSGDATA-VAL1 = 3710.
BUSGDATA-VAL2 = 3125.
BUSGDATA-VAL3 = 5178.
BUSGDATA-VAL4 = 3730.
APPEND BUSGDATA.

HIER-UEKEY = '12200000003'.
HIER-ELKEY = '12210000004'.
HIER-ETEXT = 'El.221-S4'.

April 2001  143
Calling SAP Hierarchy Graphics and Others, With Dialog

HIER-VAL1  =  37.
HIER-VAL2  =  45.
APPEND HIER.

BUSGDATA-ELKEY = HIER-ELKEY.
BUSGDATA-VAL1 = 2710.
BUSGDATA-VAL2 = 4125.
BUSGDATA-VAL3 = 1278.
BUSGDATA-VAL4 = 1730.
APPEND BUSGDATA.

HIER-UEKEY = '12200000003'.
HIER-ELKEY = '12220000004'.
HIER-ETEXT = 'El.222-S4'.
HIER-VAL1  =  67.
HIER-VAL2  =  72.
APPEND HIER.

BUSGDATA-ELKEY = HIER-ELKEY.
BUSGDATA-VAL1 = 2710.
BUSGDATA-VAL2 = 1125.
BUSGDATA-VAL3 = 4178.
BUSGDATA-VAL4 = 3570.
APPEND BUSGDATA.

SORT HIER.
LOOP AT HIER.
  WRITE: / HIER-ELKEY,20 HIER-UEKEY,50 HIER-ETEXT.
ENDLOOP.

*-------------------------------------------------------------*
* PF10                                                      *
*                                                          *
* Start up an SAP Hierarchy window                          *
*                                                          *
*-------------------------------------------------------------*
* When the user selects (clicks on) a node, a text window appears.
*-------------------------------------------------------------*
* The dialog ends when the SAP Hierarchy is closed by the user.
* Closing the text window will lead to a restart next time.  *
*-------------------------------------------------------------*

AT PF10.

CLEAR HIER1_STAT.

TEMA_STAT = '2'. "// Start and do not wait

DO.

CALL FUNCTION 'GRAPH_HIERARCHY'

EXPORTING

NLVAL = '2/1'
TTEXT = SY-TITLE
NDIMS = 4
SMODE = 2
VTXT1 = 'Display Criterion 1'

"// Criterion 1 has
VAL11 = 20 "// 2 threshold values
"// and
VAL12 = 50 "// 3 colors.
COL11 = 'DARKBLUE'
COL12 = 'DARKYELLOW'
COL13 = 'RED'
VTXT2 = 'Display Criterion 2'

"// Criterion 2 has
VAL21 = 30 "// 2 threshold values
"// and
VAL22 = 60 "// 3 colors.
COL21 = 'BLUE'
COL22 = 'YELLOW'
Calling SAP Hierarchy Graphics and Others, With Dialog

```plaintext
COL23 = 'BLACK'
NTEXT = 'WHITE'
STAT = HIER1_STAT
SUPER = 'X'

IMPORTING
NODES = HIER-ELKEY
M_TYP = M_TYP

TABLES
DATA = HIER.

HIER1_STAT = '4'.  // Next time, just wait

CASE M_TYP.
  WHEN 'D'.
    EXIT.  // Exit condition
  WHEN ' '.
  WHEN '0'.
    // From non-Hier window
    // It must be from text
    // window, and must
    // have been 'Q'.
    TEMA_STAT = '2'.  // Restart TEMA next
    // time
  WHEN 'Q'.
  WHEN 'I'.

READ TABLE HIER WITH KEY HIER-ELKEY.
READ TABLE BUSGDATA WITH KEY HIER-ELKEY.
PERFORM DISPLAY_TEXT TABLES MTEXT USING HIER BUSGDATA.
CALL FUNCTION 'TEXT_MATRIX'
  EXPORTING
    TTEXT = 'Info Window'
    RSIZE = '6'
    CSIZE = '40'
    STAT = TEMA_STAT

TABLES
  DATA = MTEXT.

TEMA_STAT = '5'.  // Reload data next
```
Calling SAP Hierarchy Graphics and Others, With Dialog

"/ time

ENDCASE.
ENDDO.

*-------------------------------------------------------------*
* PF11
* Start an SAP Hierarchy window.
*
* When the user clicks on a node, a second SAP Hierarchy window is opened in which a sub-tree of the hierarchy is displayed.
* Clicking on nodes in the second window has no effect.
*-------------------------------------------------------------*

* The dialog ends when the user closes the first SAP Hierarchy window. If the user closes the second HIER window first, SAP Hierarchy is re-started anew (in a second window) if the user clicks on further hierarchy nodes.

AT PF11.
CLEAR HIER1_STAT. "// Start and wait
HIER2_STAT = '2'. "// Start and do not wait
DO.
CALL FUNCTION 'GRAPH_HIERARCHY'
EXPORTING
WINID = 'HIER1'
TTEXT = SY-TITLE
VTXT1 = 'Display Criterion 1'
    "// Criterion 1 has
VAL11 = 20 "// 2 threshold values
VAL12 = 50 "// 3 colors
COL11 = 'DARKBLUE'
COL12 = 'DARKYELLOW'
COL13 = 'RED'
Calling SAP Hierarchy Graphics and Others, With Dialog

VTXT2 = 'Display Criterion 2'
    "// Criterion 2 has no
STAT = HIER1_STAT    "// threshold values
SMODE = 2
NTEXT = 'WHITE'
SUPER = 'X'

IMPORTING
    NODES = HIER-ELKEY
    M_TYP = M_TYP
    RWNID = RWNID

TABLES
    DATA = HIER.
HIER1_STAT = '4'.

CASE M_TYP.
    WHEN 'D'.
        EXIT.
    WHEN '0'.
    WHEN 'Q'.    "// HIER2 has been
        "// closed.
        "// (Closing HIER1 would
        "// produce a 'D'
        "// message
        "// due to using
        "// SUPER.)
        HIER2_STAT = '2'.    "// Restart HIER2 next
            "// time
    WHEN 'I'.    "// Msg. from an SAP
        "// Hierarchy window
        "// (HIER1
        "// or HIER2)
    IF RWNID EQ 'HIER1'.
        PERFORM DISPLAY NODES USING HIER-ELKEY.
        CALL FUNCTION 'GRAPH_HIERARCHY'
            EXPORTING
                TTEXT = 'Partial Hierarchy Display'
Calling SAP Hierarchy Graphics and Others, With Dialog

SMODE = '2'
STAT = HIER2_STAT
WINID = 'HIER2'

TABLES
DATA = PHIER.

HIER2_STAT = '5'.   "// Reload data next
"// time
ELSE.
"// Ignore any clicking
"// in HIER2 window
ENDIF.
ENDCASE.
ENDDO.

*-------------------------------------------------------------*
* PF12                                                        *
*                                                             *
* Start an SAP Hierarchy window                               *
*                                                             *
*-------------------------------------------------------------*
* When the user clicks on a node, an SAP Business Graphics    *
* window appears to display its data.                         *
*-------------------------------------------------------------*
* The dialog ends when the user closes the hierarchy window.  *
* If the user closes the SAP Business Graphic windows first, *
* it cannot be opened again. That is, further selection of    *
* nodes                                                     *
* (after closing a previous BUSG window) will not result in *
* new SAP Business Graphics windows being opened.            *
*-------------------------------------------------------------*

AT PF12.
TCOL = 'Column 1'. APPEND TCOL.
TCOL = 'Column 2'. APPEND TCOL.
Calling SAP Hierarchy Graphics and Others, With Dialog

```
TCOL = 'Column 3'. APPEND TCOL.
TCOL = 'Column 4'. APPEND TCOL.
CLEAR HIER1_STAT.
BUSG_STAT = '2'.
"// Start and do not wait
"// wait
DO.
   CALL FUNCTION 'GRAPH_HIERARCHY'
      EXPORTING
         TTEXT = SY-TITLE
         NDIMS = 4
         SMODE = 2
         VTXT1 = 'Display Criterion 1'
         VAL11 = 20
         VAL12 = 50
         COL11 = 'DARKBLUE'
         COL12 = 'DARKYELLOW'
         COL13 = 'RED'
         VTXT2 = 'Display Criterion 2'
         NTEXT = 'WHITE'
         STAT  = HIER1_STAT
         SUPER = 'X'
      IMPORTING
         NODES = HIER-ELKEY
         M_TYP = M_TYP
      TABLES
         DATA  = HIER.
      HIER1_STAT = '4'.
"// Next time wait
CASE M_TYP.
   WHEN 'D'.
      EXIT.
   WHEN 'I'.
      READ TABLE HIER WITH KEY HIER-ELKEY.
      READ TABLE BUSGDATA WITH KEY HIER-ELKEY.
      CALL FUNCTION 'GRAPH_MATRIX_2D'
         EXPORTING
```
Calling SAP Hierarchy Graphics and Others, With Dialog

TITL = HIER-ETEXT
NROW = SY-TABIX
STAT = BUSG_STAT

TABLES
DATA = BUSGDATA
OPTS = GOPTS
TCOL = TCOL.

BUSG_STAT = '5'.

WHEN '0'.                        }// Must be from BUSG
"// Must be 'Q'
"// Do NOT restart BUSG
"// here.

ENDCASE.
ENDDO.

*-------------------------------------------------------------*
* PF13                                                        *
*                                                             *
* Start an SAP Hierarchy window                               *
*                                                             *
*-------------------------------------------------------------*
* When the user clicks on a node, an SAP Statistics window    *
* appears to display its data. If the user then clicks on a    *
* point in a statistics curve, a text window appears.          *
*-------------------------------------------------------------*
* The dialog ends when the user closes the hierarchy window.   *
* If the user closes either the statistics or text window, the*
* corresponding program (Statistics or Text Matrix) is re-     *
* started anew the next time the user clicks on nodes or       *
* curve points.                                               *
*-------------------------------------------------------------*

AT PF13.
DATA: BEGIN OF STATDATA OCCURS 1,
    X TYPE P,
    Y TYPE P,
END OF STATDATA.
CLEAR HIER1_STAT.
STAT_STAT = '2'.                      // Start and do not
    //  wait
TEMA_STAT = '2'.                      // Start and do not
    //  wait
DO.
    CALL FUNCTION 'GRAPH_HIERARCHY'
    EXPORTING
        TTEXT = SY-TITLE
        NDIMS = 4
        SMODE = 2
        VTXT1 = 'Display Criterion 1'
        VAL11 = 20
        VAL12 = 50
        COL11 = 'DARKBLUE'
        COL12 = 'DARKYELLOW'
        COL13 = 'RED'
        VTXT2 = 'Display Criterion 2'
        NTEXT = 'WHITE'
        STAT  = HIER1_STAT
        SUPER = 'X'
    IMPORTING
        NODES = NODE
        RBUFF = RBUFF
        RWNID = RWNID
        M_TYP = M_TYP
    TABLES
        DATA  = HIER.
    CASE M_TYP.
    "// Next time wait
WHEN 'D'.
    EXIT.
WHEN 'Q'.
WHEN ' '.
WHEN 'I'.
    MOVE NODE TO HIER-ELKEY.
    READ TABLE HIER WITH KEY HIER-ELKEY.
    READ TABLE BUSGDATA WITH KEY HIER-ELKEY.
    REFRESH GOPTS.
    GOPTS = '$ 4'. APPEND GOPTS.
    GOPTS = 'C_ART=1'. APPEND GOPTS.
    GOPTS = 'CTYPE=3'. APPEND GOPTS.
    GOPTS = 'COLOR=3'. APPEND GOPTS.
    GOPTS = 'LTEXT=Monthly Salary'. APPEND GOPTS.
    REFRESH STATDATA.
    APPEND STATDATA.
    STATDATA-X = 2. STATDATA-Y = BUSGDATA-VAL2.
    APPEND STATDATA.
    APPEND STATDATA.
    APPEND STATDATA.
    MOVE 'Additional Info: ' TO NODE.
    WRITE HIER-ETEXT TO NODE+17(20).
    CALL FUNCTION 'STAT_GRAPH'
      EXPORTING
        INFORM = 'X'
        NOPOS = 'X'
        STAT = STAT_STAT
        TTEXT = NODE
        TAXIS = 'Personnel'
        SCALE = 'MAN'
        XRNGU = 5
        XTCKS = '6'
Calling SAP Hierarchy Graphics and Others, With Dialog

    YRNGU  = 10000
    YTCKS  = '11'

    TABLES
        DATA   = STATDATA
        OPTS   = GOPTS.

    STAT_STAT = '5'.

    WHEN '0'.                      '// Some other window
        IF RWNID(4) = 'STAT'.      '// from Statistics
            '// window
            CALL FUNCTION 'STAT_GRAPH'
        EXPORTING
            STAT  = 'R'
            INBUF = RBUFF
        IMPORTING
            KURVE = KURVE
            INDEX = PUNKT
            M_TYP = STAT_TYP
        TABLES
            DATA   = STATDATA       '// Not used here
            OPTS   = GOPTS.         '// Not used here

        IF STAT_TYP = 'I'.
            PERFORM DISPLAY_STAT TABLES MTEXT
                USING KURVE PUNKT HIER-ETEXT
                BUSGDATA.
            CALL FUNCTION 'TEXT_MATRIX'
        EXPORTING
            TTEXT = 'Additional information window'
            RSIZE = '4'
            CSIZE = '30'
            STAT  = TEMA_STAT
        TABLES
            DATA   = MTEXT.
            TEMA_STAT = '5'.           '// Reload data next
                                      '// time

        ELSE.
STAT_STAT = '2'.          // Restart STAT next
         // time
ENDIF.
ENDIF.

IF RWNID(4) = 'TEMA'.       // Msg comes text
         // window:
         // it must have been
         // 'Q'.
TEMA_STAT = '2'.           // Restart TEMA next
         // time
ENDIF.
ENDCASE.
ENDDO.

*-------------------------------------------------------------*
* PF14                                                      *
*                                                           *
* Start up an SAP Hierarchy window.                         *
*                                                           *
*-------------------------------------------------------------*

* User selection of a hierarchy node results in the opening of*
* - an SAP Hierarchy window displaying a partial hierarchy    *
* - an SAP Statistics window                                   *
* - an SAP Business Graphics window                           *
* - an SAP Text Matrix (text) window                          *
* When the user selects a point in a Statistics window, an    *
* additional text window is displayed.                        *
*-------------------------------------------------------------*

* The dialog ends when the first hierarchy window is closed.  *
* Closing any other window (beside the first one) is          *
* remembered:                                                *
* if the corresponding programs are requested a second time,  *
* they are started anew.                                     *
Calling SAP Hierarchy Graphics and Others, With Dialog

AT PF14.

TCOL = 'Column 1'. APPEND TCOL.
TCOL = 'Column 2'. APPEND TCOL.
TCOL = 'Column 3'. APPEND TCOL.
TCOL = 'Column 4'. APPEND TCOL.

CLEAR HIER1_STAT. // Start, send data and
    // wait
BUSG_STAT = '2'. // Start and do not
    // wait
STAT_STAT = '2'. // --
HIER2_STAT = '2'. // --
TEMA_STAT = '2'. // --
TEMA1_STAT = '2'. // --

DO.

CALL FUNCTION 'GRAPH_HIERARCHY'
EXPORTING
    TTEXT = SY-TITLE
    NDIMS = 4
    SMODE = 2
    VTXT1 = 'Display Criterion 1'
    VAL11 = 20
    VAL12 = 50
    COL11 = 'DARKBLUE'
    COL12 = 'DARKYELLOW'
    COL13 = 'RED'
    VTXT2 = 'Display Criterion 2'
    NTEXT = 'WHITE'
    STAT  = HIER1_STAT
    SUPER = 'X'
    WINID = 'HIER1'

IMPORTING
    M_TYP = M_TYP
    RBUFF = RBUFF
    NODES = HIER-ELKEY
    RWNID = RWNID

156  April 2001
TABLES
DATA = HIER.
HIER1_STAT = '4'.

CASE M_TYP.
WHEN 'D'.
  EXIT.
WHEN 'Q'.
  // Must be from HIER2
  // (since quitting
  //   HIER1
  //   would cause a 'D'
  //   because of using
  //   SUPER)
  HIER2_STAT = '2'.
  // Restart next time
WHEN ' '.
WHEN 'I'.
  IF RWNID(5) = 'HIER1'.
    // Message from HIER1
    MOVE HIER-ELKEY TO NODE.
    READ TABLE HIER WITH KEY HIER-ELKEY.
    READ TABLE BUSGDATA WITH KEY HIER-ELKEY.
    REFRESH GOPTS.
    CALL FUNCTION 'GRAPH_MATRIX_2D'
      EXPORTING
      TITL = HIER-ETEXT
      NROW = SY-TABIX
      STAT = BUSG_STAT
      TABLES
      DATA = BUSGDATA
      OPTS = GOPTS
      TCOL = TCOL.
      BUSG_STAT = '5'.
      PERFORM DISPLAY_TEXT TABLES MTEXT USING HIER
                       BUSGDATA.
    CALL FUNCTION 'TEXT_MATRIX'
      EXPORTING
Calling SAP Hierarchy Graphics and Others, With Dialog

```
TTEXT = 'Information window'
RSIZE = '6'
CSIZE = '30'
STAT = TEMA_STAT

TABLES
  DATA = MTEXT.
TEMA_STAT = '5'.  // Reload data next
    // time
PERFORM DISPLAY_NODES USING HIER-ELKEY.
CALL FUNCTION 'GRAPH_HIERARCHY'
EXPORTING
  TTEXT = 'Partial Hierarchy Display'
  SMODE = '2'
  STAT = HIER2_STAT
  WINID = 'HIER2'

TABLES
  DATA = PHIER.
HIER2_STAT = '5'.  // Reload data next
    // time
MOVE NODE TO HIER-ELKEY.
READ TABLE HIER WITH KEY HIER-ELKEY.
REFRESH GOPTS.
GOPTS = '$ 4'. APPEND GOPTS.
GOPTS = 'C_ART=1'. APPEND GOPTS.
GOPTS = 'CTYPE=3'. APPEND GOPTS.
GOPTS = 'COLOR=3'. APPEND GOPTS.
GOPTS = 'LTEXT=Monthly Salary'. APPEND GOPTS.
REFRESH STATDATA.
APPEND STATDATA.
STATDATA-X = 2. STATDATA-Y = BUSGDATA-VAL2.
APPEND STATDATA.
APPEND STATDATA.
```
APPEND STATDATA.
MOVE 'Additional Info: ' TO NODE.
WRITE HIER-ETEXT TO NODE+17(20).
CALL FUNCTION 'STAT_GRAPH'
  EXPORTING
    INFORM = 'X'
    NOPOS = 'X'
    STAT = STAT_STAT
    TTEXT = NODE
    TAXIS = 'Personnel'
    SCALE = 'MAN'
    XRNGU = 5
    XTCKS = '6'
    YRNGU = 10000
    YTCKS = '11'
  TABLES
    DATA = STATDATA
    OPTS = GOPTS.
    STAT_STAT = '5'.
ELSE.
  '// Msg must be from
  '// HIER2:
  '// ignore it.
ENDIF.
WHEN '0'.
  '// From any non-HIER
  '// window
IF RWNID(4) = 'STAT'.
  '// Msg is from STAT
  '// window
CALL FUNCTION 'STAT_GRAPH'
  '// Reuse RBUFF to fill
  '// STAT return
  '// parameters
    STAT = 'R'
    INBUF = RBUFF
IMPORTING
    KURVE = KURVE
    INDEX = PUNKT
Calling SAP Hierarchy Graphics and Others, With Dialog

M_TYP = STAT_TYP

TABLES
DATA = STATDATA  "// Not used here
OPTS = GOPTS.    "// Not used here

IF STAT_TYP = 'I'.
  CSIZE = '30'
  "// -message
  "// time
  PERFORM DISPLAY_STAT TABLES MTEXT
    USING KURVE PUNKT HIER-ETEXT
    BUSGDATA.

CALL FUNCTION 'TEXT_MATRIX'

  EXPORTING
  TTEXT = 'Statistics information window'
  RSIZE = '4'
  CSIZE = '30'
  STAT = TEMA1_STAT
  WINID = 'TEMA1'

  TABLES
  DATA = MTEXT.

  TEMA1_STAT = '5'.  "// Reload next time
  ELSE.
    "// Must have been a 'Q'
    STAT_STAT = '2'.  "// Restart next time
  ENDIF.
ENDIF.

IF RWNID(4) = 'BUSG'.
  "// Must be a 'Q'
  "// -message
  BUSG_STAT = '2'.  "// Restart BUSG next
  "// time
ENDIF.

IF RWNID(4) = 'TEMA'.
  "// Must be TEMA or
  "// TEMA1
  IF RWNID EQ 'TEMA1'.
    "// Restart TEMA1 next
    "// time
    TEMA1_STAT = '2'.
  ELSE.
TEMA_STAT = '2'.    
"// Restart TEMA next
"// time

ENDIF.
ENDIF.
ENDCASE.
ENDDO.

*-------------------------------------------------------------*
*       FORM DISPLAY_NODES                                    *
*-------------------------------------------------------------*
FORM DISPLAY_NODES USING ELKEY.
FIELD-SYMBOLS: <1>,<2>.
DATA KEY(11).
DATA S.
MOVE ELKEY TO KEY.       
"//Save Key !
REFRESH PHIER.
LOOP AT HIER.
  IF HIER-ELKEY GE KEY.
    ASSIGN KEY+10(1) TO <1>.
    ASSIGN HIER-ELKEY+10(1) TO <2>.
    IF <2> LE <1> AND S NE SPACE. EXIT. ENDIF.
    S = 'X'.
    MOVE HIER TO PHIER.
    MOVE HIER-ELKEY TO PHIER-ETEXT.
    APPEND PHIER.
  ENDIF.
ENDLOOP.
ENDFORM.

*-------------------------------------------------------------*
*       FORM DISPLAY_TEXT                                     *
*-------------------------------------------------------------*
FORM DISPLAY_TEXT TABLES MTEXT USING ETEXT EDATA.
DATA: BEGIN OF TEXT,
      ELKEY(11),
Calling SAP Hierarchy Graphics and Others, With Dialog

UESKEY(11),
ETEXT(20),
VAL1 TYPE P,
VAL2 TYPE P,
END OF TEXT.

DATA: BEGIN OF DATA,
ELKEY(11),
VAL1 TYPE P,
VAL2 TYPE P,
VAL3 TYPE P,
VAL4 TYPE P,
END OF DATA.

DATA FELD(40).
MOVE ETTEXT TO TEXT.
MOVE EDATA TO DATA.
REFRESH MTEXT.
MOVE: 'Element:      ' TO FELD.
MOVE: TEXT-ELKEY  TO FELD+14.
MOVE FELD TO MTEXT.
APPEND MTEXT.
MOVE: 'Parent Elem.: ' to FELD.
MOVE: TEXT-UEKEY  TO FELD+14.
MOVE FELD TO MTEXT.
APPEND MTEXT.
MOVE: 'Label:        ' TO FELD.
MOVE: TEXT-ETEXT  TO FELD+14.
MOVE FELD TO MTEXT.
APPEND MTEXT.
MOVE: 'Value:        ' TO FELD.
MOVE: TEXT-VAL1   TO FELD+14(10).
MOVE FELD TO MTEXT.
APPEND MTEXT.
MOVE: 'Column 1:     ' TO FELD.
MOVE: DATA-VAL1   TO FELD+14(10).
MOVE FELD TO MTEXT.
APPEND MTEXT.
MOVE: 'Column 2: ' TO FELD.
MOVE: DATA-VAL2 TO FELD+14(10).
MOVE FELD TO MTEXT.
APPEND MTEXT.
MOVE: 'Column 3: ' TO FELD.
MOVE: DATA-VAL3 TO FELD+14(10).
MOVE FELD TO MTEXT.
APPEND MTEXT.
MOVE: 'Column 4: ' TO FELD.
MOVE: DATA-VAL4 TO FELD+14(10).
MOVE FELD TO MTEXT.
APPEND MTEXT.
ENDFORM.

*-------------------------------------------------------------*
*       FORM DISPLAY_STAT                                     *
*-------------------------------------------------------------*
FORM DISPLAY_STAT TABLES MTEXT USING KURVE PUNKT ETEXT EDATA.
DATA: BEGIN OF DATA,
      ELKEY(11),
      VAL1 TYPE P,
      VAL2 TYPE P,
      VAL3 TYPE P,
      VAL4 TYPE P,
END OF DATA.
DATA FELD(40).
DATA PU_1.
MOVE EDATA TO DATA.
REFRESH MTEXT.
MOVE: 'Title.......: ' TO FELD.
MOVE: ETEXT TO FELD+13.
MOVE FELD TO MTEXT.
APPEND MTEXT.
MOVE: 'Curve Nr...: ' TO FELD.
Calling SAP Hierarchy Graphics and Others, With Dialog

    WRITE KURVE TO FELD+13.
    MOVE FELD TO MTEXT.
    APPEND MTEXT.
    MOVE: 'Point Nr....: ' TO FELD.
    WRITE PUNKT TO FELD+13.
    MOVE FELD TO MTEXT.
    APPEND MTEXT.
    MOVE: 'Value.......: ' TO FELD.
    PU_1 = PUNKT.
    CASE PU_1.
        WHEN '1'.
            WRITE DATA-VAL1 TO FELD+13(8).
        WHEN '2'.
            WRITE DATA-VAL2 TO FELD+13(8).
        WHEN '3'.
            WRITE DATA-VAL3 TO FELD+13(8).
        WHEN '4'.
            WRITE DATA-VAL4 TO FELD+13(8).
    ENDCASE.
    MOVE FELD TO MTEXT.
    APPEND MTEXT.
ENDFORM.
Calling SAP Statistical Graphics, Without Dialog

The following ABAP program illustrates some of the function module calls discussed in the preceding topics.

PROGRAM G_STAT_1.

*------------------------------------------------------------*
*                                                            *
*------------------------------------------------------------*
* This report calls SAP Statistics without interactive dialog*
*------------------------------------------------------------*

* Representation of Sine and Cosine Functions               *
*                                                            *
* Input: the number of subdivisions of the circle (360 degrees). *
* The report ends when F3 is pressed in the Statistics window. *
*------------------------------------------------------------*

DATA: D  TYPE F,
      W  TYPE F,
      F  TYPE F,
      PI  TYPE F.

DATA: BEGIN OF OPTS OCCURS 1,  "// Options table
      C(80),
      END OF OPTS.

DATA: BEGIN OF DATA OCCURS 1,  "// Data table
      W  TYPE P,
      F  TYPE F,
      END OF DATA.

PARAMETER: NUMBER TYPE P DEFAULT '30'.

PI = '3.14159' / 180.
D = 360 / ( NUMBER - 1).
Calling SAP Statistical Graphics, Without Dialog

*------------*
* Function 1 *
*------------*

DO NUMBER TIMES.                       // Fill data table
   DATA-W = W.
   DATA-F = COS( W * PI).
   APPEND DATA.
   ADD D TO W.
ENDDO.
OPTS = '$'.                            // Fill options table
WRITE NUMBER TO OPTS+1.           // Provide number of points
APPEND OPTS.
OPTS = 'COLOR=7'.                 // Color of curve
APPEND OPTS.
OPTS = 'THICK=1'.                 // Width of curve
APPEND OPTS.
OPTS = 'LTEXT=COSINE Curve'.      // Legend text
APPEND OPTS.
OPTS = 'DTEXT=Description text for representing a COS curve'.
APPEND OPTS.

*------------*
* Function 2 *
*------------*

W = 0.
DO NUMBER TIMES.
   DATA-W = W.
   DATA-F = SIN( W * PI).
   APPEND DATA.
   ADD D TO W.
ENDDO.
OPTS = '$'.
WRITE NUMBER TO OPTS+1.
APPEND OPTS.
OPTS = 'COLOR=4'.
APPEND OPTS.
OPTS = 'THICK=1'.
APPEND OPTS.
OPTS = 'LTEXT=SINE Curve'.
APPEND OPTS.
OPTS = 'DTEXT=Representation of a SIN curve'.
APPEND OPTS.

*-------------*
* Function 3 *
*-------------*

W = 0.
DO NUMBER TIMES.
   DATA-W = W.
   DATA-F = COS( W * PI) * SIN( W * PI).
   APPEND DATA.
   ADD D TO W.
ENDDO.
OPTS = '$'.
WRITE NUMBER TO OPTS+1.
APPEND OPTS.
OPTS = 'C_ART=1'.
APPEND OPTS.
OPTS = 'CTYPE=4'.
APPEND OPTS.
OPTS = 'COLOR=3'.
APPEND OPTS.
OPTS = 'LTEXT=COSINE-SINE Curve'.
APPEND OPTS.
OPTS = 'DTEXT=Description text for representing a COSINE-SINE curve'.
APPEND OPTS.
Calling SAP Statistical Graphics, Without Dialog

*-------------------------------------------------------------*
* Graphical Output                                          *
*-------------------------------------------------------------*
DATA X1 TYPE F VALUE '0'.
DATA X2 TYPE F VALUE '360'.
DATA Y1 TYPE F VALUE '-1.0'.
DATA Y2 TYPE F VALUE '1.0'.

CALL FUNCTION 'STAT_GRAPH'

EXPORTING

    TTEXT = 'Wave Functions'
    UTEXT = 'Sine and Cosine Functions'
    TAXIS = 'Angle in Degrees'
    X_ART = 'LIN'
    Y_ART = 'LIN'
    SCALE = 'MAN'
    XRNGL = X1
    XRNGU = X2
    YRNGL = Y1
    YRNGU = Y2
    XTCKS = '13'
    YTCKS = '11'

TABLES

    OPTS  = OPTS
    DATA  = DATA.
Calling SAP Gannt Chart Graphics, With Dialog

The following ABAP program illustrates some of the function module calls discussed in the preceding topics.

*-------------------------------------------------------------*
* This program calls SAP Gantt Chart.

* GRGANT_2 calls GRAPH_GANTT, sending chart data (line
* items, events, and milestones). The online user can then
* update the chart by moving, modifying, deleting or
* inserting new elements. The ABAP program responds by
* updating its own data.
*-------------------------------------------------------------*

REPORT NO STANDARD PAGE HEADING.

*-------------------------------------------------------------*
* Definition of ITEM table (Line items in the graph)
*-------------------------------------------------------------*

DATA: BEGIN OF ITEM OCCURS 1.
  INCLUDE STRUCTURE GGAIT.
DATA:   NEWNUM LIKE SY-TABIX,   // for moving ITEMS
        MODIF(1),               // 'X' (change), 'D' (delete)
END OF ITEM.

*------------------------------------------------------------*
* Definition of ELEM table (Event elements in the graph)
*------------------------------------------------------------*

DATA: BEGIN OF ELEM OCCURS 1.
  INCLUDE STRUCTURE GGAEL.
DATA:   TBEG TYPE T,           // not send
        MODIF(1),              // 'X' = modif, 'D' = deleted
END OF ELEM.

DATA: BEGIN OF LGEL OCCURS 1.
  INCLUDE STRUCTURE GGAEL.
DATA: END OF LGEL.
DATA: BEGIN OF DFEL OCCURS 1.
  INCLUDE STRUCTURE GGAEL.
Calling SAP Gantt Chart Graphics, With Dialog

DATA: END OF DFEL.

*-------------------------------------------------------------*
* Definition of MIST table (Milestones in the graph)         *
*-------------------------------------------------------------*

DATA: BEGIN OF MIST OCCURS 1.
    INCLUDE STRUCTURE GGAMI.
    DATA: TBEG TYPE T,      "// not send
        MODIF(1),      "// 'X' = modif, 'D' = deleted
    END OF MIST.

DATA: BEGIN OF LGMS OCCURS 1.
    INCLUDE STRUCTURE GGAMI.

DATA: END OF LGMS.

DATA: BEGIN OF DFMS OCCURS 1.
    INCLUDE STRUCTURE GGAMI.

DATA: END OF DFMS.

*-------------------------------------------------------------*
* Definition of return-msg table                            *
* (NOTE: DEF. MUST LOOK EXACTLY LIKE THIS!!)                *
*-------------------------------------------------------------*

DATA: BEGIN OF BACK OCCURS 1.
    INCLUDE STRUCTURE GGABA.

DATA: END OF BACK.

*-------------------------------------------------------------*
* More declarations                                          *
*-------------------------------------------------------------*

DATA: DATE_NULL TYPE D VALUE '18991230'.
DATA  C(10).
DATA: M_TYP(1), G_TYP(1), CUA_ID(10).  "// Dialog return-msg
DATA ITEMNO TYPE P.

SET PF-STATUS 'GRAF'.

ITEM-ITEXT = 'ARD'. ITEM-UCOLOR = 'GREEN'. APPEND ITEM.
CLEAR ITEM-UCOLOR.
ITEMNO = 1.
PERFORM FILL_ELEM USING 'Today Show'
   '19931126' '200000' 15 'M' 'BLUE' '15' 'C'.
PERFORM FILL_ELEM USING 'Play Romeo & Julia'
   '19931126' '201800' 90 'M' 'GREEN' '15' 'L'.
PERFORM FILL_ELEM USING 'Current Issues'
   '19931126' '223000' 30 'M' 'BLUE' '15' 'C'.
ITEM-ITEXT = 'ZDF'. APPEND ITEM.
ITEMNO = 2.
PERFORM FILL_ELEM USING 'Football DFB League'
   '19931127' '203000' 120 'M' 'GREEN' '15' 'L'.
ITEM-ITEXT = 'SAT 1'. APPEND ITEM.
ITEMNO = 3.
PERFORM FILL_ELEM USING 'Today Show'
   '19931128' '200000' 15 'M' 'BLUE' '15' 'C'.
PERFORM FILL_ELEM USING 'Focus'
   '19931128' '201600' 45 'M' 'GREEN' '15' 'L'.
ITEM-ITEXT = 'Cable channel'. APPEND ITEM.
ITEMNO = 4.
PERFORM FILL_ELEM USING 'Sport Highlights'
   '19931130' '210000' 90 'M' 'BLUE' '15' 'C'.
ITEM-ITEXT = 'PRO 7'. APPEND ITEM.
ITEMNO = 5.
PERFORM FILL_ELEM USING 'Rambo I'
   '19931201' '150000' 170 'M' 'GREEN' '15' 'L'.

April 2001  171
Calling SAP Gannt Chart Graphics, With Dialog

MIST-ITEMNO = 1.
MIST-BEG = ELEM-BEG.
MIST-TBEG = ELEM-TBEG.
MIST-TXT = 'TV Coord. Committee'.
MIST-FORM = 'SQUARE'.
MIST-COLOR = 'BLACK'.
APPEND MIST.

PERFORM AUSGABE.

AT PF8.
SY-LSIND = 0.
GANTT_STAT = SPACE.
DO.
  REFRESH BACK.
  DFEL-FILLD = '1'.
  DFEL-BRDON = '1'.

  DATA: BEG LIKE GGAEL-BEG.
  DATA: END LIKE GGAEL-BEG.
  DATA: LIN LIKE GGAEL-BEG.
  DATA: VIW LIKE GGAEL-BEG.
  DATA: TIM LIKE GGAEL-BEG.

  READ TABLE ELEM INDEX 1.

  BEG = ELEM-BEG.
  END = ELEM-BEG + 9000000.
  VIW = ELEM-BEG + 3000000.
  LIN = ELEM-BEG + 7200.

  CALL FUNCTION 'GRAPH_GANTT'
       EXPORTING
Calling SAP Gannt Chart Graphics, With Dialog

*             VGRID   = 'X'
*             HGRID   = 'X'
WHEADER = 'Studio Schedule'
TTITLE = 'SAP AG : Studio 3'
TTEXT = 'Programs'
*             LEGEND  = 'Legend'
*             TLENGTH = '5'
*             TMBEGIN = BEG
*             TMEND   = END
*             TLINE   = LIN
*             TMVIEW  = VIW
NOTXT = SPACE
TUNIT = 'H'       // Time unit: Hour
DFEL = DFEL
DFMS = DFMS
MODIFY = 'ON'
STAT = GANTT_STAT

IMPORTING
M_TYP = M_TYP
G_TYP = G_TYP       // 'C', 'S', '0', '1', '2'
CUA_ID = CUA_ID

TABLES
ITEM = ITEM
MIST = MIST
ELEM = ELEM
LGMS = MIST
LGEL = ELEM
MSGT = BACK.

GANTT_STAT = '4'.       // just wait next time

*------------------------------------*
* Evaluate msg from graphics program *
*------------------------------------*
Calling SAP Gannt Chart Graphics, With Dialog

```
IF M_TYP = 'D'. EXIT. ENDF.       "// Graph closed
CASE G_TYP.
  WHEN 'C'.                        "// Pickup, ignore
  WHEN 'U'.                        "// Timer, ignore
  WHEN 'S'.                        "// Save, exit
    EXIT.                          "// Dialogloop
  WHEN '0'.                        "// CUA-Action bar
    "// CUA_ID contains
    "// ACT_ID
    "// will not happen
  WHEN '1'.                        "// CUA-Help request
    "// not yet
    "// will not happen
  WHEN '2'.                        "// CUA-Helpline
    "// CUA_ID contains
    "// ACTSTRING
    "// will not happen
ENDCASE.
ENDDO.                               "// Dialog Loop

* Close will do nothing, if already closed.

CALL FUNCTION 'GRAPH_DIALOG'
  EXPORTING
    CLOSE = 'X'.
```

* When processing multiple saves, events, milestones and line items are not really deleted from the output table. Instead, they are marked as deleted (ELEM-MODIF = 'D').
* However, before the next data-transfer to the graphics program, these elements must be removed from their
* respective tables. This ensures that objects known internally
* to the graphics program can be matched up correctly with the
* corresponding table rows. (PERFORM PACK)
* ----------------------------------------------------------- *

* Merge BACK to Original Data
* Mark changes

LOOP AT BACK.
  IF BACK-OBJ = 'E'.  "// ELEMENTS
    CASE BACK-ACT.
      WHEN 'M'.  "// Move
        TABIX = BACK-ELM.
        READ TABLE ELEM INDEX TABIX.
        ELEM-MODIF = 'X'.
        ELEM-TBEG = BACK-BEG - ( ( BACK-DAT - DATE_NULL) * 86400).
        ELEM-BEG = BACK-BEG.
        ELEM-TXT = BACK-TXT.
        ELEM-DUR = BACK-DUR.
        ELEM-ITEMNO = BACK-ITM.
        MODIFY ELEM INDEX TABIX.
        DELETE BACK.
      WHEN 'A'.  "// Edit (Length etc)
        TABIX = BACK-ELM.
        READ TABLE ELEM INDEX TABIX.
        ELEM-MODIF = 'X'.
        ELEM-TBEG = BACK-BEG - ( ( BACK-DAT - DATE_NULL) * 86400).
        ELEM-BEG = BACK-BEG.
        ELEM-TXT = BACK-TXT.
        ELEM-DUR = BACK-DUR.
        ELEM-ITEMNO = BACK-ITM.
        MODIFY ELEM INDEX TABIX.
        DELETE BACK.
Calling SAP Gannt Chart Graphics, With Dialog

WHEN 'I'.  "// Insert New
  ELEM-MODIF = 'X'.
  ELEM-TBEG = BACK-BEG - ( ( BACK-DAT - DATE_NULL) *
                         86400).
  ELEM-BEG = BACK-BEG.
  ELEM-TXT = BACK-TXT.
  ELEM-DUR = BACK-DUR.
  ELEM-ITEMNO = BACK-ITM.
  APPEND ELEM.
  DELETE BACK.
WHEN 'D'.  "// Delete Element
  TABIX = BACK-ELM.
  READ TABLE ELEM INDEX TABIX.
  ELEM-MODIF = 'D'.
  MODIFY ELEM INDEX TABIX.
  DELETE BACK.
ENDCASE.
ENDIF.  "// ELEMENTS

IF BACK-OBJ = 'M'.  "// MILESTONES
  CASE BACK-ACT.
    WHEN 'M'.
      TABIX = BACK-ELM.
      READ TABLE MIST INDEX TABIX.
      MIST-MODIF = 'X'.
      MIST-TBEG = BACK-BEG - ( ( BACK-DAT - DATE_NULL) *
                               86400).
      MIST-BEG = BACK-BEG.
      MIST-TXT = BACK-TXT.
      MIST-ITEMNO = BACK-ITM.
      MODIFY MIST INDEX TABIX.
      DELETE BACK.
    WHEN 'A'.
      TABIX = BACK-ELM.
      READ TABLE MIST INDEX TABIX.
MIST-MODIF = 'X'.
MIST-TBEG = BACK-BEG - ( ( BACK-DAT - DATE_NULL) * 86400).
MIST-BEGIN = BACK-BEGIN.
MIST-TXT = BACK-TXT.
MIST-ITEMNO = BACK-ITM.
MODIFY MIST INDEX TABIX.
DELETE BACK.
WHEN 'I'.
MIST-MODIF = 'X'.
MIST-TBEG = BACK-BEG - ( ( BACK-DAT - DATE_NULL) * 86400).
MIST-BEGIN = BACK-BEGIN.
MIST-TXT = BACK-TXT.
MIST-ITEMNO = BACK-ITM.
APPEND MIST.
DELETE BACK.
WHEN 'D'.
TABIX = BACK-ELM.
READ TABLE MIST INDEX TABIX.
MIST-MODIF = 'D'.
MODIFY MIST INDEX TABIX.
DELETE BACK.
ENDCASE.
ENDIF.       // MILESTONES

IF BACK-OBJ = 'I'.       // ITEMS
CASE BACK-ACT.
WHEN 'A'.       // Change without moving -> text only
TABIX = BACK-ITM.
READ TABLE ITEM INDEX TABIX.
ITEM-MODIF = 'X'.
ITEM-ITEXT = BACK-TXT.
MODIFY ITEM INDEX TABIX.
DELETE BACK.
Calling SAP Gannt Chart Graphics, With Dialog

WHEN 'I'. "// Insert -> new number with append
  ITEM-MODIF = 'N'.
  ITEM-ITEXT = BACK-TXT.
  APPEND ITEM.
  DELETE BACK.
WHEN 'M'. "// Move -> exchange the order
  "// ITM : old number, ELM : new number
  TABIX = BACK-ITM.
  READ TABLE ITEM INDEX TABIX.
  ITEM-NEWNUM = BACK-ELM.
  IF BACK-ITM NE BACK-ELM.
    IF ITEM-MODIF NE 'N'. "// save 'NEW'-Info
      ITEM-MODIF = 'M'. "// Marker for PACK
    "// nothing to do
    ENDIF.
    MODIFY ITEM INDEX TABIX.
  ELSE.
    "// unchanged item
    "// inserting items
    "// reports
    "//   ALL items
    "// this is not an error
    "// just ignore it
  ENDIF.
  DELETE BACK.
WHEN 'D'. "// Delete -> ELEM and MIST intentionally
  "// deleted
  TABIX = BACK-ITM.
  READ TABLE ITEM INDEX TABIX.
  ITEM-MODIF = 'D'.
  MODIFY ITEM INDEX TABIX.
  DELETE BACK.
ENDCASE.
ENDIF.                             "// ITEMS
* DELETE BACK. "// global for all cases
ENDLOOP.

* Reorg ELEM after ITEM-Move
LOOP AT ELEM.
  TABIX = ELEM-ITEMNO.
  READ TABLE ITEM INDEX TABIX.
  IF ITEM-MODIF = 'M' OR ITEM-MODIF EQ 'N'.
    ELEM-ITEMNO = ITEM-NEWNUM.
    MODIFY ELEM.
  ENDIF.
ENDLOOP.

* Reorg MIST after ITEM-Move
LOOP AT MIST.
  TABIX = MIST-ITEMNO.
  READ TABLE ITEM INDEX TABIX.
  IF ITEM-MODIF = 'M' OR ITEM-MODIF EQ 'N'.
    MIST-ITEMNO = ITEM-NEWNUM.
    MODIFY MIST.
  ENDIF.
ENDLOOP.

LOOP AT ITEM. "// Causes unambiguous Sort for Merge/Reorder
  IF ITEM-MODIF = 'M' OR ITEM-MODIF EQ 'N'.
    "// already has new
    "// number
  ELSE.
    ITEM-NEWNUM = SY-TABIX. "// current number
    MODIFY ITEM.
  ENDIF.
ENDLOOP.

SORT ITEM BY NEWNUM.

* End of Merge
* Changes are marked
Calling SAP Gannt Chart Graphics, With Dialog

    PERFORM AUSGABE.
    SET PF-STATUS 'GRAF'.

* Clear Markers
* Delete physically

    PERFORM PACK.

*-------------------------------------------------------------*  
*    FORM AUSGABE         (No USING-Parameter)               
*-------------------------------------------------------------*  
FORM AUSGABE.

DETAIL.
TAB_INDEX = 0.
LOOP AT ITEM.
   TAB_INDEX = TAB_INDEX + 1.
   CASE ITEM-MODIF.
      WHEN SPACE.
         WRITE: / ITEM-ITEXT.
      WHEN 'X'.
         WRITE: / ITEM-ITEXT INPUT ON.
      WHEN 'M'.
         WRITE: / ITEM-ITEXT, '<reorder>' INPUT ON.
      WHEN 'N'.
         WRITE: / ITEM-ITEXT, '<new>' INPUT ON.
      WHEN 'D'.
         WRITE: '<deleted>' INPUT ON.
   ENDCASE.
LOOP AT ELEM.
   * ONLY IF SORTED
   * IF ELEM-ITEMNO GT TAB_INDEX.
   * EXIT.
   * ENDF.
   IF ELEM-ITEMNO EQ TAB_INDEX.
D = ((ELEM-BEG - ELEM-TBEG) / 86400) + DATE_NULL.

T = ELEM-DUR.

DAYS = (ELEM-DUR - T) / 86400.

CASE ELEM-MODIF.

  WHEN 'X'.
    WRITE: /2 ELEM-TXT(29) INPUT ON.
    MOVE D TO C.
    WRITE C USING EDIT MASK '___.__.__' INPUT ON.
    MOVE ELEM-TBEGIN TO C.
    WRITE: C USING EDIT MASK '__:__:__' INPUT ON.
    IF DAYS = 0.
    WRITE: ' 0' INPUT ON, ':' INPUT ON.
    ELSE.
    WRITE: DAYS INPUT ON, ':' INPUT ON.
   ENDIF.
    MOVE T TO C.
    WRITE: C USING EDIT MASK '__:__:__' INPUT ON.

  WHEN 'D'.
    WRITE: /2 '<deleted>' INPUT ON.

  WHEN SPACE.
    WRITE: /2 ELEM-TXT(29).
    MOVE D TO C.
    WRITE C USING EDIT MASK '___.__.__'.
    MOVE ELEM-TBEGIN TO C.
    WRITE: C USING EDIT MASK '__:__:__'.
    IF DAYS = 0.
    WRITE: ' 0', '.
    ELSE.
    WRITE: DAYS, ':'.
   ENDIF.
    MOVE T TO C.
    WRITE: C USING EDIT MASK '__:__:__'.

  WHEN OTHERS.
ENDCASE.
ENDIF.
Calling SAP Gannt Chart Graphics, With Dialog

        ENDLOOP.

        LOOP AT MIST.
          IF MIST-ITEMNO EQ TAB_INDEX.
            D = ( ( MIST-BEG - MIST-TBEG) / 86400) + DATE_NULL.
            CASE MIST-MODIF.
              WHEN 'X'.
                WRITE: /2 MIST-TXT(29) INPUT ON.
                MOVE D TO C.
                WRITE: C USING EDIT MASK '___:___:' INPUT ON.
                MOVE MIST-TBEG TO C.
                WRITE: C USING EDIT MASK '___:___:' INPUT ON.
              WHEN 'D'.
                WRITE: / 'MS : deleted' INPUT ON.
              WHEN SPACE.
                WRITE: /2 MIST-TXT(29).
                MOVE D TO C.
                WRITE: C USING EDIT MASK '___:___:___'.
                MOVE MIST-TBEG TO C.
                WRITE: C USING EDIT MASK '___:___:___'.
            ENDCASE.
          ENDIF.
        ENDLOOP.
      ENDLOOP.
    ENDFORM.

*-------------------------------------------------------------------*
*    FORM PACK         (No USING-Parameter)                          *
*-------------------------------------------------------------------*
FORM PACK.

        LOOP AT ELEM.
          IF ELEM-MODIF = 'D'.
            DELETE ELEM.
          ELSE.
            ELEM-MODIF = SPACE.
            MODIFY ELEM.
ENDIF.
ENDLOOP.
LOOP AT MIST.
  IF MIST-MODIF = 'D'.
    DELETE MIST.
  ELSE.
    MIST-MODIF = SPACE.
    MODIFY MIST.
  ENDIF.
ENDLOOP.
LOOP AT ITEM.
  IF ITEM-MODIF = 'D'.
    DELETE ITEM.  "// related elements already gone
  ENDIF.
  IF ITEM-MODIF = 'M'.  "// Shift order
    "// Adapt all elements
    "// Adapt all milestones
      ITEM-NEWNUM = SPACE.
      ITEM-MODIF = SPACE.
      MODIFY ITEM.
  ENDIF.
ENDLOOP.

ENDFORM.

TOP-OF-PAGE.
  PERFORM TOP_TEXT.

TOP-OF-PAGE DURING LINE-SELECTION.
  PERFORM TOP_TEXT.

*-------------------------------------------------------------*
Calling SAP Gannt Chart Graphics, With Dialog

*    FORM FILL_ELEM
*-------------------------------------------------------------*
FORM FILL_ELEM USING TEXT DBEG TBEG DUR UNIT COL SIZE POS.
ELEM-ITEMNO = ITEMNO.
ELEM-TXT = TEXT.
D = DBEG.
T = TBEG.
ELEM-TBEG = T.
ELEM-BEG = (D - DATE_NULL) * 86400 + T.
CASE UNIT.
    WHEN 'S'.
        ELEM-DUR = DUR.
    WHEN 'M'.
        ELEM-DUR = DUR * 60.
    WHEN 'H'.
        ELEM-DUR = DUR * 60 * 60.
    WHEN 'D'.
        ELEM-DUR = DUR * 60 * 60 * 24.
    WHEN 'W'.
        ELEM-DUR = DUR * 60 * 60 * 24 * 7.
    WHEN 'N'.
        ELEM-DUR = DUR * 60 * 60 * 24 * 30.
    WHEN 'Q'.
        ELEM-DUR = DUR * 60 * 60 * 24 * 90.
    WHEN 'Y'.
        ELEM-DUR = DUR * 60 * 60 * 24 * 365.
ENDCASE.
ELEM-BAKGR = COL.
* ELEM-PLACE = POS.
ELEM-HEIGH = SIZE.
CASE POS.
    WHEN 'U'.
        ELEM-PLACE = 'ABOVE'.
    WHEN 'C'.
        ELEM-PLACE = 'CENTER'.

184  April 2001
WHEN 'L'.
    ELEM-PLACE = 'BELOW'.
ENDCASE.
APPEND ELEM.
ENDFORM.

*-------------------------------------------------------------*
*    FORM TOP_TEXT         (No USING-Parameter)                *
*-------------------------------------------------------------*

FORM TOP_TEXT.
WRITE: /32 'Dispo-System for Studios'.
ULINE.
WRITE: / ' Programs                      Date       Time      '.
WRITE: 'Length'.
ULINE.
ENDFORM.
Calling SAP Business Graphics, With Dialog

The following ABAP program illustrates some of the function module calls discussed in the preceding topics.

*-------------------------------------------------------------*
* This program calls SAP Business Graphics.                   *
* GRBUSGMO calls GRAPH_2D, sending graph data (2D chart).     *
* The online user can then update the chart                  *
* (using mouse or BUSG modify window).                        *
* The ABAP program responds by updating its own data.        *
*-------------------------------------------------------------*

REPORT GRBUSGMO.
*
* Data for 2D-Modification
*
DATA: BEGIN OF DATA OCCURS 1,
  TEXT(25),
  VALUE TYPE P,
END OF DATA.
DATA: TCOL1(5) VALUE '#1993'.
DATA: TITLE(25) VALUE 'Merchandise 1993'.
DATA: TIT2(10) VALUE 'Product'.
DATA: MOD_ROW(5), MOD_COL(5), MOD_VAL(20).
DATA: M_TYP, B_TYP, BUSG_STAT.
FIELD-SYMBOLS: <F>.
DATA-TEXT = 'Refrigerators'.
DATA-VALUE = 12345.
APPEND DATA.
DATA-TEXT = 'Freezers'.
DATA-VALUE = 22222.
APPEND DATA.

DATA-TEXT = 'Ovens'.
DATA-VALUE = 11111.
APPEND DATA.

DATA-TEXT = 'Dishwashers'.
DATA-VALUE = 54321.
APPEND DATA.

PERFORM OUTPUT_2D.

AT PF08.

BUSG_STAT = SPACE.                   // start and wait
* 
* Event-Loop
* 
DO.

CALL FUNCTION 'GRAPH_2D'

EXPORTING
STAT       = BUSG_STAT
INFORM     = '7'         // allow modify
MAIL_ALLOW = 'X'         // allow mail

IMPORTING
M_TYP      = M_TYP
B_TYP      = B_TYP
MOD_ROW    = MOD_ROW
MOD_COL    = MOD_COL
MOD_VAL    = MOD_VAL

TABLES
DATA       = DATA.

BUSG_STAT = '4'.                   // Just wait next time

CASE M_TYP.
  WHEN 'D'.                        // Programm ended
    EXIT.
  EXIT.
Calling SAP Business Graphics, With Dialog

    WHEN 'I'.                        // Valid message
        IF B_TYP = 'M'.                // Modify-event
            READ TABLE DATA INDEX MOD_ROW.
            ASSIGN COMPONENT MOD_COL OF STRUCTURE DATA TO <F>.
            <F> = MOD_VAL.
            MODIFY DATA INDEX MOD_ROW.
            ENDF.
            IF B_TYP = 'S'.    // Select-event (ignored)
                ENDF.                   // For element (MOD_ROW-MOD_COL)
        ENDCASE.
    ENDDO.
    PERFORM OUTPUT_2D.

*-------------------------------------------------------------*
*       FORM OUTPUT_2D  (Show the actual data)                *
*-------------------------------------------------------------*

FORM OUTPUT_2D.
    WRITE: / 'Products',22 TCOL1.
    SKIP 2.
    LOOP AT DATA.
        WRITE: / DATA-TEXT, 22 DATA-VALUE.
    ENDDO.
ENDFORM.
Calling SAP HPGL Display

The following ABAP program calls SAP HPGL Display.

REPORT GRHPGL_1.
DATA: BEGIN OF HPGL OCCURS 1,
    ENTRY(128),
    END OF HPGL.

HPGL = 'SP6;PUPA0,0;'. APPEND HPGL.
HPGL = 'PDPA100,100,100,200,200,200,200,100,100,100,100,100;'. APPEND HPGL.
HPGL = 'SP1;PUPA100,150;'. APPEND HPGL.
HPGL = 'PDPA100,150,150,200,200,150,150,100,100,150,100,150,100,150;'. APPEND HPGL.

CALL FUNCTION 'GRAPH_HPGL' TABLES DATA = HPGL.
Calling SAP Portfolio Graphics, With Dialog

The following ABAP program illustrates some of the function module calls discussed in the preceding topics.

*-------------------------------------------------------------*
* This program calls SAP Portfolio Graphics.                  *
*
* GRPORT_1 calls GRAPH_PORT, sending portfolio graph objects, *
* with trend lines to be drawn between them.                  *
*-------------------------------------------------------------*

REPORT GRPORT_1.

DATA: BEGIN OF OBJT OCCURS 1.          }// Object Attributes
    INCLUDE STRUCTURE GPOOBJT.
DATA: END OF OBJT.

DATA: BEGIN OF AREA_TAB OCCURS 1.      }// Areas (unused)
    INCLUDE STRUCTURE GPOAREA.
DATA: END OF AREA_TAB.

DATA: BEGIN OF COL_TEXT OCCURS 1,      }// Value columns
    C(40),
    END OF COL_TEXT.

DATA: BEGIN OF VALUES OCCURS 1,        }// Object fields
    P1 TYPE P,
    P2 TYPE P,
    P3 TYPE P,
    P4 TYPE P,
    P5 TYPE P,
    P6 TYPE P,
    P7 TYPE P,
    P8 TYPE P,
    P9 TYPE P,
    END OF VALUES.
DATA: BEGIN OF AXIS OCCURS 4.
   INCLUDE STRUCTURE GPOAXIS.
DATA: END OF AXIS.

DATA SAVE LIKE SY-TABIX.

SET PF-STATUS 'GRAF'.

* Constraints: Number of VALUES rows == Number of OBJT rows
*               : Num of VALUES columns == Number of COL_TEXT rows

* Column title for $OBJD::VALTIT
COL_TEXT = '% Market share'. APPEND COL_TEXT.
COL_TEXT = '% Growth'. APPEND COL_TEXT.
COL_TEXT = 'Sales'. APPEND COL_TEXT.
COL_TEXT = 'Market share trend'. APPEND COL_TEXT.
COL_TEXT = 'Growth trend'. APPEND COL_TEXT.
COL_TEXT = 'Fixed costs'. APPEND COL_TEXT.
COL_TEXT = 'Overhead costs'. APPEND COL_TEXT.
COL_TEXT = 'Admin. costs'. APPEND COL_TEXT.
COL_TEXT = 'Estimated profits'. APPEND COL_TEXT.

* Attributes for $AXIS
AXIS-POS = 'RIGHT'.
AXIS-BACKCLR = 'DARKGRAY'.
AXIS-TEXTCLR = 'WHITE'.
AXIS-SCALCLR = 'WHITE'.
AXIS-SHADOW = 'N'.
AXIS-TICKS = '50'.
AXIS-START = '0'.
AXIS-END = '100'.
APPEND AXIS.

AXIS-POS = 'TOP'.

April 2001
Calling SAP Portfolio Graphics, With Dialog

AXIS-BACKCLR = 'DARKGRAY'.
AXIS-TEXTCLR = 'WHITE'.
AXIS-SCALCLR = 'WHITE'.
AXIS-SHADOW = SPACE.
AXIS-TICKS = '50'.
AXIS-START = '0'.
AXIS-END = '100'.
APPEND AXIS.

AXIS-POS = 'LEFT'.
AXIS-BACKCLR = 'DARKGRAY'.
AXIS-TEXTCLR = 'WHITE'.
AXIS-SCALCLR = 'WHITE'.
AXIS-SHADOW = 'N'.
AXIS-TICKS = '10'.
AXIS-START = '0'.
AXIS-END = '100'.
APPEND AXIS.

AXIS-POS = 'BOTTOM'.
AXIS-BACKCLR = 'DARKGRAY'.
AXIS-TEXTCLR = 'WHITE'.
AXIS-SCALCLR = 'WHITE'.
AXIS-SHADOW = 'N'.
AXIS-TICKS = '50'.
AXIS-START = '0'.
AXIS-END = '100'.
APPEND AXIS.

* Objects
*BJT-LINECLR = 'WHITE'.

* 1. Trend line

OBJT-FORM = 'RECTANGLE'.

* Objects
*BJT-LINECLR = 'WHITE'.
OBJT-FILLCLR = 'YELLOW'.
OBJT-TEXTCLR = 'BLUE'.
OBJT-TITLE   = 'Clarion 1990'.
OBJT-INFO    = 'Sales in 1990'.
VALUES-P1 = 10.
VALUES-P2 = 20.
VALUES-P3 = 10000.
VALUES-P4 = 22.
VALUES-P5 = 22.
VALUES-P6 = 400.
VALUES-P7 = 300.
VALUES-P8 = 200.
VALUES-P9 = 100.
APPEND OBJT.
APPEND VALUES.

OBJT-FORM    = 'RECTANGLE'.
OBJT-FILLCLR = 'YELLOW'.
OBJT-TEXTCLR = 'BLUE'.
OBJT-TITLE   = 'Clarion 1991'.
OBJT-INFO    = 'Sales in 1991'.
VALUES-P1 = 22.
VALUES-P2 = 22.
VALUES-P3 = 15000.
VALUES-P4 = 45.
VALUES-P5 = 35.
VALUES-P6 = 100.
VALUES-P7 = 200.
VALUES-P8 = 300.
VALUES-P9 = 400.
APPEND OBJT.
APPEND VALUES.
Calling SAP Portfolio Graphics, With Dialog

OBJT-FORM    = 'RECTANGLE'.
OBJT-FILLCLR = 'YELLOW'.
OBJT-TEXTCLR = 'BLUE'.
OBJT-TITLE   = 'Clarion 1992'.
OBJT-INFO    = 'Sales in 1992'.
VALUES-P1 = 45.
VALUES-P2 = 35.
VALUES-P3 = 22000.
VALUES-P4 = 62.
VALUES-P5 = 33.
VALUES-P6 = 20.
VALUES-P7 = 40.
VALUES-P8 = 80.
VALUES-P9 = 160.
APPEND OBJT.
APPEND VALUES.

OBJT-FORM    = 'RECTANGLE'.
OBJT-FILLCLR = 'YELLOW'.
OBJT-TEXTCLR = 'BLUE'.
OBJT-TITLE   = 'Clarion 1993'.
OBJT-INFO    = 'Sales in 1993'.
VALUES-P1 = 62.
VALUES-P2 = 33.
VALUES-P3 = 12000.
VALUES-P4 = 62.
VALUES-P5 = 33.
VALUES-P6 = 600.
VALUES-P7 = 700.
VALUES-P8 = 600.
VALUES-P9 = 700.
APPEND OBJT.
APPEND VALUES.

* 2. Trend line
OBJT-FORM = 'CIRCLE'.
OBJT-FILLCLR = 'GREEN'.
OBJT-TEXTCLR = 'BLUE'.
OBJT-TITLE = 'Sony 1988'.
OBJT-INFO = 'Sony UX400/CD, Sales in 1988'.
VALUES-P1 = 2.
VALUES-P2 = 50.
VALUES-P3 = 2500.
VALUES-P4 = 26.
VALUES-P5 = 57.
VALUES-P6 = 100.
VALUES-P7 = 700.
VALUES-P8 = 20.
VALUES-P9 = 50.
APPEND OBJT.
APPEND VALUES.

OBJT-FORM = 'CIRCLE'.
OBJT-FILLCLR = 'GREEN'.
OBJT-TEXTCLR = 'BLUE'.
OBJT-TITLE = 'Sony 1989'.
OBJT-INFO = 'Sony UX400/CD, Sales in 1989'.
VALUES-P1 = 26.
VALUES-P2 = 57.
VALUES-P3 = 3500.
VALUES-P4 = 58.
VALUES-P5 = 70.
VALUES-P6 = 100.
VALUES-P7 = 600.
VALUES-P8 = 20.
VALUES-P9 = 70.
APPEND OBJT.
APPEND VALUES.
Calling SAP Portfolio Graphics, With Dialog

OBJT-FORM = 'CIRCLE'.
OBJT-FILLCLR = 'GREEN'.
OBJT-TEXTCLR = 'BLUE'.
OBJT-TITLE = 'Sony 1990'.
OBJT-INFO = 'Sony UX400/CD, Sales in 1990'.
VALUES-P1 = 58.
VALUES-P2 = 70.
VALUES-P3 = 11000.
VALUES-P4 = 79.
VALUES-P5 = 68.
VALUES-P6 = 100.
VALUES-P7 = 500.
VALUES-P8 = 25.
VALUES-P9 = 90.
APPEND OBJT.
APPEND VALUES.

OBJT-FORM = 'CIRCLE'.
OBJT-FILLCLR = 'GREEN'.
OBJT-TEXTCLR = 'BLUE'.
OBJT-TITLE = 'Sony 1991'.
OBJT-INFO = 'Sony UX400/CD, Sales 1991'.
VALUES-P1 = 79.
VALUES-P2 = 68.
VALUES-P3 = 13000.
VALUES-P4 = 87.
VALUES-P5 = 54.
VALUES-P6 = 100.
VALUES-P7 = 430.
VALUES-P8 = 45.
VALUES-P9 = 120.
APPEND OBJT.
APPEND VALUES.

OBJT-FORM = 'CIRCLE'.

196  April 2001
OBJT-FILLCLR = 'GREEN'.
OBJT-TEXTCLR = 'BLUE'.
OBJT-TITLE = 'Sony 1992'.
OBJT-INFO = 'Sony UX400/CD, Sales in 1992'.
VALUES-P1 = 87.
VALUES-P2 = 54.
VALUES-P3 = 13500.
VALUES-P4 = 80.
VALUES-P5 = 34.
VALUES-P6 = 90.
VALUES-P7 = 390.
VALUES-P8 = 44.
VALUES-P9 = 150.
APPEND OBJT.
APPEND VALUES.

OBJT-FORM = 'CIRCLE'.
OBJT-FILLCLR = 'GREEN'.
OBJT-TEXTCLR = 'BLUE'.
OBJT-TITLE = 'Sony 1993'.
OBJT-INFO = 'Sony UX400/CD, Sales in 1993'.
VALUES-P1 = 80.
VALUES-P2 = 34.
VALUES-P3 = 23000.
VALUES-P4 = 80. /* Line to itself
VALUES-P5 = 34.
VALUES-P6 = 90.
VALUES-P7 = 370.
VALUES-P8 = 47.
VALUES-P9 = 165.
APPEND OBJT.
APPEND VALUES.

AREA_TAB-FORM = 'RECTANGLE'.
AREA_TAB-FILLCLR = 'RED'.

April 2001
Calling SAP Portfolio Graphics, With Dialog

AREA_TAB_XFRO = '10'.
AREA_TAB_YFRO = '70'.
AREA_TAB_XTO = '40'.
AREA_TAB_YTO = '90'.
AREA_TABTITLE = 'Winner takes all'.
APPEND AREA_TAB.

LOOP AT OBJT.
  WRITE: / OBJT-TITLE(20).
  SAVE = SY-TABIX.
  READ TABLE VALUES INDEX SAVE.
  WRITE 25 VALUES-P3.
ENDLOOP.

AT PF08.

CALL FUNCTION 'GRAPH_PORT'
  EXPORTING
    TITLE_TITLE = 'Trend-Portfolio for Sony and Clarion
Radio'
    TITLE_INFO = 'The two objects over several years'
    TITLE_TEXTCLR = 'YELLOW'
    TITLE_BACKCLR = 'DARKGRAY'
    TITLE_SIZE = '2'
    FRM_BACKCLR = 'GRAY'
    FRM_CORDCLR = 'WHITE'
    FRM_SCALCLR = 'WHITE'
    FRM_SCALMOD = 'LONGDASH'
    FRM_GRIDX = '10'
    FRM_GRIDY = '10'
    FRM_SHADOW = 'X'
    OBJ_XVAL = '0'
    OBJ_YVAL = '1'
    OBJ_SVAL = '2'
    OBJ_XARR = '3'
OBJ_YARR = '4'

TABLES
AXIS = AXIS
COL_TEXT = COL_TEXT
OBJT = OBJT
VALUES = VALUES
AREA_TAB = AREA_TAB.
**Calling SAP Network Graphics, Without Dialog**

The following ABAP program illustrates some of the function module calls discussed in the preceding topics.

*-------------------------------------------------------------*
* This program calls SAP Network Graphics.                 *
*                                                           *
* NETZ_BO1 calls CNET_GRAPHIC_NETWORK, creating nodes and links
* between the nodes.                                        *
*-------------------------------------------------------------*

REPORT NETZ_BO1.

INCLUDE LCNETDAT.       "General data declarations
INCLUDE LCNETCON.       "Constants

* Creating nodes
PERFORM MAKE_NODES TABLES NET_NODES_TAB NVALS_TAB.

* Creating links
PERFORM MAKE_LINES TABLES LINES_TAB LVALS_TAB.

DO.
* Calling SAP Network Graphics
  CALL FUNCTION 'CNET_GRAPHIC_NETWORK'
  EXPORTING
    STAT      = STAT
  IMPORTING
    M_TYP     = M_TYP
  TABLES
    CLUSTERS  = CLUSTERS_TAB
    CVALS    = CVALS_TAB
    DELETIONS = DELETE_TAB
    INODES   = INODES_TAB
    LINES    = LINES_TAB
LVALS    = LVALS_TAB
NODES    = NET_NODES_TAB
NVALS    = NVALS_TAB
POSITIONS = POSITIONS_TAB.

CASE M_TYP.
    WHEN NET_CONST-M_TYP_D.
        EXIT.
    WHEN NET_CONST-M_TYP_I.
        STAT = NET_CONST-STAT_4.
    ENDCASE.
ENDDO.

*---------------------------------------------------------------------*
*       FORM MAKE_NODES                                               *
*---------------------------------------------------------------------*
*       Creating nodes                                                *
*---------------------------------------------------------------------*
*  -->  NODES                                                         *
*  -->  NVALS                                                         *
*---------------------------------------------------------------------*
FORM MAKE_NODES TABLES NODES STRUCTURE CNG_NODES
NVALS STRUCTURE NET_NVALS.

    NODES-ID   = '1'.
    APPEND NODES.
    NVALS-ID   = NODES-ID.
    NVALS-FL   = NET_CONST-TEXT_INDEX_0.
    NVALS-VAL  = TEXT-001.                                    "Node 1
    APPEND NVALS.

    NODES-ID   = '2'.
    APPEND NODES.
    NVALS-ID   = NODES-ID.
    NVALS-FL   = NET_CONST-TEXT_INDEX_0.

April 2001  201
Calling SAP Network Graphics, Without Dialog

```plaintext
NVALS-VAL = TEXT-002. "Node 2
APPEND NVALS.

NODES-ID = '3'.
APPEND NODES.
NVALS-ID = NODES-ID.
NVALS-FL = NET_CONST-TEXT_INDEX_0.
NVALS-VAL = TEXT-003. "Node 3
APPEND NVALS.

NODES-ID = '4'.
APPEND NODES.
NVALS-ID = NODES-ID.
NVALS-FL = NET_CONST-TEXT_INDEX_0.
NVALS-VAL = TEXT-004. "Node 4
APPEND NVALS.

NODES-ID = '5'.
APPEND NODES.
NVALS-ID = NODES-ID.
NVALS-FL = NET_CONST-TEXT_INDEX_0.
NVALS-VAL = TEXT-005. "Node 5
APPEND NVALS.

NODES-ID = '6'.
APPEND NODES.
NVALS-ID = NODES-ID.
NVALS-FL = NET_CONST-TEXT_INDEX_0.
NVALS-VAL = TEXT-006. "Node 6
APPEND NVALS.

ENDFORM.
```

*---------------------------------------------------------------------*
Calling SAP Network Graphics, Without Dialog

* FORM MAKE_LINES *
*---------------------------------------------------------------------*
* Creating links between nodes *
*---------------------------------------------------------------------*
* --> LINES *
* --> LVALS *
*---------------------------------------------------------------------*

FORM MAKE_LINES TABLES LINES STRUCTURE CNG_LINES
LVALS STRUCTURE NET_LVALS.

LINES-ID = '1'.
LINES-PRE = '1'.
LINES-SUC = '2'.
LINES-AB = NET_CONST-AOB_EA. "Normal relation
APPEND LINES.
LVALS-ID = LINES-ID.
LVALS-FL = NET_CONST-TEXT_INDEX_0.
LVALS-VAL = LINES-AB.
APPEND LVALS.

LINES-ID = '2'.
LINES-PRE = '2'.
LINES-SUC = '4'.
LINES-AB = NET_CONST-AOB_EA. " Normal relation
APPEND LINES.
LVALS-ID = LINES-ID.
LVALS-FL = NET_CONST-TEXT_INDEX_0.
LVALS-VAL = LINES-AB.
APPEND LVALS.

LINES-ID = '3'.
LINES-PRE = '1'.
LINES-SUC = '3'.
LINES-AB = NET_CONST-AOB_EA. " Normal relation
APPEND LINES.
Calling SAP Network Graphics, Without Dialog

LVALS-ID = LINES-ID.
LVALS-FL = NET_CONST-TEXT_INDEX_0.
LVALS-VAL = LINES-AB.
APPEND LVALS.

LINES-ID = '4'.
LINES-PRE = '3'.
LINES-SUC = '4'.
LINES-AB = NET_CONST-AOB_EA. " Normal relation
APPEND LINES.
LVALS-ID = LINES-ID.
LVALS-FL = NET_CONST-TEXT_INDEX_0.
LVALS-VAL = LINES-AB.
APPEND LVALS.

LINES-ID = '5'.
LINES-PRE = '4'.
LINES-SUC = '5'.
LINES-AB = NET_CONST-AOB_EA. " Normal relation
APPEND LINES.
LVALS-ID = LINES-ID.
LVALS-FL = NET_CONST-TEXT_INDEX_0.
LVALS-VAL = LINES-AB.
APPEND LVALS.

LINES-ID = '6'.
LINES-PRE = '5'.
LINES-SUC = '6'.
LINES-AB = NET_CONST-AOB_EA. " Normal relation
APPEND LINES.
LVALS-ID = LINES-ID.
LVALS-FL = NET_CONST-TEXT_INDEX_0.
LVALS-VAL = LINES-AB.
APPEND LVALS.
ENDFORM.
Calling SAP Bar Chart Graphics, Without Dialog

The following ABAP program illustrates some of the function module calls discussed in the preceding topics.

*-------------------------------------------------------------*
* This program calls SAP Bar Chart Graphics.                  *
*                                                          *
* BARCBO01 calls BARC_SET_TIME_AXIS and BARC_ADD_CHART, creating a time axis, a table and bars.  
* A graphics profile is used.                                *
*-------------------------------------------------------------*

REPORT BARCBO01.

INCLUDE SGRCCNST.                      "General graphics constants
INCLUDE LBARCCON.                      "Bar chart constants
INCLUDE BARCDATA.                      "General data declarations

DATA: CHART   LIKE BCCHART-ID,
       SECTION LIKE BCSECTION-ID,
       VAL     LIKE BCVALS-VAL.

PARAMETER: GROUP    LIKE TBCGT-PRF_GROUP     DEFAULT 'DEMO',
             NAME      LIKE TBCGT-PRF_NAME       DEFAULT '000000000001',
             INDEX     LIKE TBCGT-PRF_INDEX      DEFAULT '1'.

* Defining a graphics profile
GRAPH_PROFILE-PRF_GROUP   = GROUP.
GRAPH_PROFILE-PRF_NAME    = NAME.
GRAPH_PROFILE-PRF_INDEX   = INDEX.

* Defining beginning and end of time axis
CALL FUNCTION 'BARC_SET_TIME_AXIS' 
              EXPORTING
       START = '01.01.1994;'
       END   = '31.12.1994;'.

206  April 2001
* Creating a chart
CALL FUNCTION 'BARC_ADD_CHART'
   IMPORTING
       ID = CHART.

* Defining the column width in the tables section
CALL FUNCTION 'BARC_SET_COLUMN_WIDTH'
   EXPORTING
       CHART_ID = CHART
       WIDTH    = 4000.

* Defining the table heading
BOXES-ID         = 1.
BOXES-TYPE       = BC_CONST-ROW_BOX.
BOXES-CHART_ID   = CHART.
BOXES-FORM_TYPE  = 'B1'.
BOXES-COLOR_TYPE = 'B0'.
APPEND BOXES.

BOX_VALS-ID  = BOXES-ID.
BOX_VALS-FL  = '0'.
BOX_VALS-VAL = TEXT-001.               "Boxes
APPEND BOX_VALS.

POSITIONS-OBJ_ID     = BOXES-ID.
POSITIONS-OBJ_TYPE   = BC_CONST-BOX_OBJECT.
POSITIONS-CHART_ID   = CHART.
POSITIONS-ROW_NUMBER = 0.
APPEND POSITIONS.

WRITE BOXES-ID TO VAL.

CALL FUNCTION 'BARC_SET_CHART_ATTRIB'
   EXPORTING
       FL  = BC_CONST-CHART_TITLE_BOX
Calling SAP Bar Chart Graphics, Without Dialog

    ID = CHART
    VAL = VAL.

* Creating a section

CALL FUNCTION 'BARC_ADD_SECTION'

    EXPORTING
        SIZE = 100
        START = '01.01.1994;'
        UNIT = '5'

    IMPORTING
        ID = SECTION.

* Inserting ribbons in a section

CALL FUNCTION 'BARC_ADD_RIBBON'

    EXPORTING
        CHART_ID = CHART
        COLOR_TYPE = 'R1'
        SECTION_ID = SECTION
        UNIT = '5'.    "Day ribbon"

CALL FUNCTION 'BARC_ADD_RIBBON'

    EXPORTING
        CHART_ID = CHART
        COLOR_TYPE = 'R2'
        SECTION_ID = SECTION
        UNIT = '3'.    "Month ribbon"

* Defining boxes for table section

PERFORM SET_BOXES TABLES BOXES

    BOX_VALS
    POSITIONS

    USING CHART.

* Defining nodes in the chart, data of bars

PERFORM SET_NODES TABLES NODES
DESCRIBING NODE VALS AND POSITIONS USING CHART.

STAT = SGRC_CONST-STAT_1. "Open Window, Transfer Data, Wait for Input"

DO.
* Processing data for bar chart
  CALL FUNCTION 'BARC_GRAPHIC_PBO'
  EXPORTING
    CONFIRM   = SPACE
    PROFILE   = GRAPH_PROFILE
    STAT      = STAT
  TABLES
    BOXES     = BOXES
    BOX_VALS  = BOX_VALS
    DELETIONS = DELETIONS
    LINKS     = LINKS
    LINK_VALS = LINK_VALS
    NODES     = NODES
    NODE_VALS = NODE_VALS
    POSITIONS = POSITIONS.

* Handing over data to graphics program
  CALL FUNCTION 'GRAPH_RECEIVE'
  IMPORTING
    MCODE             = M_TYP
  EXCEPTIONS
    FRONTEND_ERROR    = 01
    INV_COMMUNICATION = 02
    NO_BATCH          = 03.

  IF NOT SY-SUBRC IS INITIAL.
    EXIT.
  ENDIF.
CASE M_TYP.
  WHEN SGRC_CONST-M_TYP_D.
    EXIT.
  WHEN SGRC_CONST-M_TYP_I.
* Evaluating data for printing
    CALL FUNCTION 'BARC_GRAPHIC_PAI'
      IMPORTING
        GRAPH_CMD = GRAPH_CMD
        TABLES
          BOXES = BOXES
          BOX_VALS = BOX_VALS
          DELETIONS = DELETIONS
          NODES = NODES
          NODE_VALS = NODE_VALS
          POSITIONS = POSITIONS
          LINKS = LINKS
          LINK_VALS = LINK_VALS
      EXCEPTIONS
        INV_RWNID = 01.

      ABAP_CMD = GRAPH_CMD.
      STAT = SGRC_CONST-STAT_4.  "Wait for input
ENDCASE.
ENDDO.

*---------------------------------------------------------------*
*      FORM MAKE_BOX                                             *
*---------------------------------------------------------------*
* Creatin a box                                                  *
*---------------------------------------------------------------*
FORM MAKE_BOX TABLES BOXES STRUCTURE BCBOXES
  BOX_VALS STRUCTURE BCVALS
  POSITIONS STRUCTURE BCPOSITION
USING  CHART LIKE BCCHART-ID
ID TYPE I
FORMAT
COLOR
TEXT
ROW TYPE I.

BOXES-ID         = ID.
BOXES-TYPE       = BC_CONST-ROW_BOX.
BOXES-CHART_ID   = CHART.
BOXES-FORM_TYPE  = FORMAT.
BOXES-COLOR_TYPE = COLOR.
APPEND BOXES.

BOX_VALS-ID  = BOXES-ID.
BOX_VALS-FL  = '0'.
BOX_VALS-VAL = TEXT.
APPEND BOX_VALS.

POSITIONS-OBJ_ID     = BOXES-ID.
POSITIONS-OBJ_TYPE   = BC_CONST-BOX_OBJECT.
POSITIONS-CHART_ID   = BOXES-CHART_ID.
POSITIONS-ROW_NUMBER = ROW.
APPEND POSITIONS.

ENDFORM.  "MAKE_BOX

*---------------------------------------------------------------------*
*       FORM SET_BOXES                                                *
*---------------------------------------------------------------------*
*       Creating boxes for the table section                         *
*---------------------------------------------------------------------*
FORM SET_BOXES TABLES BOXES STRUCTURE BCBOXES
     BOX_VALS STRUCTURE BCVALS
Calling SAP Bar Chart Graphics, Without Dialog

```
POSITIONS STRUCTURE BCPOSITION
  USING CHART LIKE BCCHART-ID.

PERFORM MAKE_BOX TABLES BOXES BOX_VALS POSITIONS
  USING CHART 2 'B1' 'B1' TEXT-002 1.

PERFORM MAKE_BOX TABLES BOXES BOX_VALS POSITIONS
  USING CHART 3 'B1' 'B1' TEXT-003 2.

PERFORM MAKE_BOX TABLES BOXES BOX_VALS POSITIONS
  USING CHART 4 'B1' 'B1' TEXT-004 3.

PERFORM MAKE_BOX TABLES BOXES BOX_VALS POSITIONS
  USING CHART 5 'B1' 'B1' TEXT-005 4.

PERFORM MAKE_BOX TABLES BOXES BOX_VALS POSITIONS
  USING CHART 6 'B1' 'B1' TEXT-006 5.

ENDFORM.

*---------------------------------------------------------------------*
*       FORM SET_NODES                                                *
*---------------------------------------------------------------------*
*       Creating nodes, setting layers                               *
*---------------------------------------------------------------------*
FORM SET_NODES TABLES NODES STRUCTURE BCNODES
  NODE_VALS STRUCTURE BCNVALS
  POSITIONS STRUCTURE BCPOSITION
  USING CHART LIKE BCCHART-ID.

  NODES-ID         = 1.
  NODES-CHART_ID   = CHART.
  APPEND NODES.
```
NODE_VALS-ID = NODES-ID.
NODE_VALS-CHART_ID = NODES-CHART_ID.
NODE_VALS-FL = '1'.
NODE_VALS-VAL = '12.01.1994;'.
APPEND NODE_VALS.

NODE_VALS-FL = '2'.
NODE_VALS-VAL = '27.06.1994;'.
APPEND NODE_VALS.

NODE_VALS-FL = BC_CONST-ADD_LAYER.
NODE_VALS-VAL = '00'.
APPEND NODE_VALS.

POSITIONS-OBJ_ID = NODES-ID.
POSITIONS-OBJ_TYPE = BC_CONST-NODE_OBJECT.
POSITIONS-CHART_ID = NODES-CHART_ID.
POSITIONS-ROW_NUMBER = 1.
APPEND POSITIONS.

*----------------------------------------------------------------------*

NODES-ID = 2.
NODES-CHART_ID = CHART.
APPEND NODES.

NODE_VALS-ID = NODES-ID.
NODE_VALS-CHART_ID = NODES-CHART_ID.
NODE_VALS-FL = '1'.
NODE_VALS-VAL = '02.02.1994;'.
APPEND NODE_VALS.

NODE_VALS-FL = '2'.
NODE_VALS-VAL = '25.07.1994;'.
APPEND NODE_VALS.
NODE_VALS-FL  = BC_CONST-ADD_LAYER.
NODE_VALS-VAL = '01'.
APPEND NODE_VALS.

POSITIONS-OBJ_ID     = NODES-ID.
POSITIONS-OBJ_TYPE   = BC_CONST-NODE_OBJECT.
POSITIONS-CHART_ID   = NODES-CHART_ID.
POSITIONS-ROW_NUMBER = 2.
APPEND POSITIONS.

*----------------------------------------------------------------------
*
NODES-ID         = 3.
NODES-CHART_ID   = CHART.
APPEND NODES.

NODE_VALS-ID       = NODES-ID.
NODE_VALS-CHART_ID = NODES-CHART_ID.
NODE_VALS-FL       = '1'.
NODE_VALS-VAL      = '01.04.1994;'.
APPEND NODE_VALS.

NODE_VALS-FL       = '2'.
NODE_VALS-VAL      = '25.09.1994;'.
APPEND NODE_VALS.

NODE_VALS-FL       = BC_CONST-ADD_LAYER.
NODE_VALS-VAL      = '02'.
APPEND NODE_VALS.

POSITIONS-OBJ_ID   = NODES-ID.
POSITIONS-OBJ_TYPE = BC_CONST-NODE_OBJECT.
POSITIONS-CHART_ID = NODES-CHART_ID.
POSITIONS-ROW_NUMBER = 3.
APPEND POSITIONS.

*-------------------------------------------------------------------------------------------------------------------
*
    NODES-ID     = 4.
    NODES-CHART_ID = CHART.
    APPEND NODES.

    NODE_VALS-ID       = NODES-ID.
    NODE_VALS-CHART_ID = NODES-CHART_ID.
    NODE_VALS-FL       = '1'.
    NODE_VALS-VAL      = '11.02.1994;'.
    APPEND NODE_VALS.

    NODE_VALS-FL      = '2'.
    NODE_VALS-VAL     = '11.07.1994;'.
    APPEND NODE_VALS.

    NODE_VALS-FL = BC_CONST-ADD_LAYER.
    NODE_VALS-VAL = '00'.
    APPEND NODE_VALS.

    POSITIONS-OBJ_ID     = NODES-ID.
    POSITIONS-OBJ_TYPE   = BC_CONST-NODE_OBJECT.
    POSITIONS-CHART_ID   = NODES-CHART_ID.
    POSITIONS-ROW_NUMBER = 4.
    APPEND POSITIONS.

*-------------------------------------------------------------------------------------------------------------------
*
    NODES-ID     = 5.
    NODES-CHART_ID = CHART.
    APPEND NODES.

    NODE_VALS-ID       = NODES-ID.
    NODE_VALS-CHART_ID = NODES-CHART_ID.
Calling SAP Bar Chart Graphics, Without Dialog

    NODE_VALS-FL   = '1'.
    NODE_VALS-VAL  = '01.05.1994;'.
    APPEND NODE_VALS.

    NODE_VALS-FL   = '2'.
    NODE_VALS-VAL  = '11.12.1994;'.
    APPEND NODE_VALS.

    NODE_VALS-FL  = BC_CONST-ADD_LAYER.
    NODE_VALS-VAL = '01'.
    APPEND NODE_VALS.

    POSITIONS-OBJ_ID  = NODES-ID.
    POSITIONS-OBJ_TYPE = BC_CONST-NODE_OBJECT.
    POSITIONS-CHART_ID = NODES-CHART_ID.
    POSITIONS-ROW_NUMBER = 5.
    APPEND POSITIONS.

    ENDFORM.                               "SET_NODES
Calling SAP Structural Graphics

The following ABAP program illustrates some of the function module calls discussed in the preceding topics.

REPORT BCGRSU01.

* parameters

* object parameters

SELECTION-SCREEN BEGIN OF LINE.

  PARAMETERS OBJ_LOAD DEFAULT 'X' AS CHECKBOX.

  SELECTION-SCREEN COMMENT 3(20) TEXT-LOB.

  SELECTION-SCREEN POSITION 24.

  PARAMETERS OBJ_NO LIKE SY-INDEX DEFAULT 10.

SELECTION-SCREEN END OF LINE.

* graphics data options

SELECTION-SCREEN BEGIN OF LINE.

  PARAMETERS MEN_LOAD DEFAULT 'X' AS CHECKBOX.

  SELECTION-SCREEN COMMENT 3(20) TEXT-LME.

  SELECTION-SCREEN POSITION 24.

  PARAMETERS MEN_OPEN AS CHECKBOX.

  SELECTION-SCREEN COMMENT 26(20) TEXT-MOP.

SELECTION-SCREEN END OF LINE.

SELECTION-SCREEN BEGIN OF LINE.

  PARAMETERS MEN_EXTD AS CHECKBOX.

  SELECTION-SCREEN COMMENT 26(20) TEXT-MEX.

SELECTION-SCREEN END OF LINE.

SELECTION-SCREEN BEGIN OF LINE.

  PARAMETERS PRF_LOAD DEFAULT 'X' AS CHECKBOX.

  SELECTION-SCREEN COMMENT 3(20) TEXT-LPR.

  SELECTION-SCREEN POSITION 24.

  PARAMETERS FMT_R2 AS CHECKBOX.

  SELECTION-SCREEN COMMENT 26(20) TEXT-FR2.

SELECTION-SCREEN END OF LINE.

SELECTION-SCREEN BEGIN OF LINE.

  PARAMETERS OPT_LOAD DEFAULT '' AS CHECKBOX.

April 2001
Calling SAP Structural Graphics

    SELECTION-SCREEN COMMENT 3(20) TEXT-LOP.
    SELECTION-SCREEN POSITION 24.
    PARAMETERS OPT_FORC AS CHECKBOX.
    SELECTION-SCREEN COMMENT 26(20) TEXT-OFO.
    SELECTION-SCREEN END OF LINE.
    PARAMETERS TXT_LOAD DEFAULT ' ' NO-DISPLAY.
    * so options
    SELECTION-SCREEN SKIP.
    SELECTION-SCREEN BEGIN OF LINE.
        PARAMETERS SOF_FLAG DEFAULT ' ' AS CHECKBOX.
        SELECTION-SCREEN COMMENT 3(20) TEXT-SEN.
        SELECTION-SCREEN POSITION 24.
        PARAMETERS SOF_RECV LIKE SOS04-L_RECNAM DEFAULT SPACE.
    SELECTION-SCREEN END OF LINE.
    SELECTION-SCREEN SKIP.
    * test options
    PARAMETERS TRC_MODE DEFAULT ' ' NO-DISPLAY.
    PARAMETERS BRK_MODE DEFAULT ' ' NO-DISPLAY.

    * table statements
    TABLES:
        GSUOBJC,
        GSUMENU,
        GSUPROF,
        GSUDISP.

    * data declarations
    * define tables for data blocks
    DATA BEGIN OF OBJECTS OCCURS 10.
        INCLUDE STRUCTURE GSUOBJC.
    DATA END OF OBJECTS.
    DATA BEGIN OF MENU OCCURS 10.
        INCLUDE STRUCTURE GSUMENU.
    DATA END OF MENU.
    DATA BEGIN OF PROFILES OCCURS 10.
INCLUDE STRUCTURE GSUPROF.
DATA END OF PROFILES.
DATA BEGIN OF TEXTS OCCURS 10.
   INCLUDE STRUCTURE GSUDISP.
DATA END OF TEXTS.
* define tables for transferring data
DATA BEGIN OF TRF_OBJECTS OCCURS 10.
   INCLUDE STRUCTURE GSUOBJC.
DATA END OF TRF_OBJECTS.
DATA BEGIN OF TRF_TOOLBOX OCCURS 10.
   INCLUDE STRUCTURE GSUMENU.
DATA END OF TRF_TOOLBOX.
DATA BEGIN OF TRF_PROFILES OCCURS 10.
   INCLUDE STRUCTURE GSUPROF.
DATA END OF TRF_PROFILES.
* several data definitions
DATA:
   DLG_STAT,
   FMT_R3 VALUE 'X',
* parameters for return session
   RET_RWNID LIKE GSUX-APPLID,
   RET_MTYP LIKE GSUX-TYPE,
   RET_CTYP LIKE GSUX-TYPE,
   RET_ITEM LIKE GSUMENU-MENUEID.
DATA BEGIN OF RET_OBJECTS OCCURS 1.
   INCLUDE STRUCTURE GSUOBJC.
DATA END OF RET_OBJECTS.
* data for trace session
DATA:
   TAB_ENTRIES LIKE SY-TABIX,
   TRC_INDEX LIKE SY-TABIX,
   TRC_SUBRC LIKE SY-SUBRC,
   OK_CODE(4).
DATA:
   TRC_MOMENT VALUE '0',
Calling SAP Structural Graphics

   MOM_START VALUE '0',
   MOM_INSERT VALUE '1'.

* initialization
* set pf-status
SET PF-STATUS 'STANDARD'.
* set break flag
IF BRK_MODE NE SPACE.
   PERFORM BREAKLEVEL_SET(SAPLGRAP)
       USING '2'.
ENDIF.
* set format
IF FMT_R2 NE SPACE.
   FMT_R3 = SPACE.
ELSE.
   FMT_R3 = 'X'.
ENDIF.
* fill table of objects if required
IF OBJ_LOAD NE SPACE.
   PERFORM BUILD_OBJECTS
       TABLES OBJECTS
       USING OBJ_NO.
ENDIF.
* fill table of menu items if required
IF MEN_LOAD NE SPACE.
   PERFORM BUILD_MENU
       TABLES MENU.
ENDIF.
* fill table of profiles if required
IF PRF_LOAD NE SPACE.
   PERFORM BUILD_PROFILES
       TABLES PROFILES.
ENDIF.
* fill table of texts if required
PERFORM BUILD_TEXTS
TABLES TEXTS.

* BREAK-POINT if required
IF BRK_MODE NE SPACE.
    BREAK-POINT 'start graphics...'.
ENDIF.
* start graphics session
DLG_STAT = '1'.
* copy data tables into transfer tables
LOOP AT OBJECTS.
    MOVE OBJECTS TO TRF_OBJECTS.
    APPEND TRF_OBJECTS.
ENDLOOP.
LOOP AT MENU.
    MOVE MENU TO TRF_TOOLBOX.
    APPEND TRF_TOOLBOX.
ENDLOOP.
LOOP AT PROFILES.
    MOVE PROFILES TO TRF_PROFILES.
    APPEND TRF_PROFILES.
ENDLOOP.
* set toolbox
IF MEN_LOAD NE SPACE.
    CALL FUNCTION 'GRAPH_STRU_SET_TOOLBOX'
    TABLES TOOLBOX = TRF_TOOLBOX.
ENDIF.
* set profiles
IF PRF_LOAD NE SPACE.
    CALL FUNCTION 'GRAPH_STRU_SET_PROFILES'
    TABLES PROFILES = TRF_PROFILES.
ENDIF.
* set display
IF TXT_LOAD NE SPACE.
    CALL FUNCTION 'GRAPH_STRU_SET_DISPLAY'
    TABLES DISPLAY = TEXTS.
Calling SAP Structural Graphics

ENDIF.

* set message
CALL FUNCTION 'GRAPH_STRU_SET_MESSAGE'

* set options
IF OPT_LOAD NE SPACE.
  DATA:
    COLBACK LIKE TGSUX-COLBACK VALUE '09',
    COLSHADOW LIKE TGSUX-COLSHADOW VALUE '07'.
  CALL FUNCTION 'GRAPH_STRU_SET_OPTIONS'
    EXPORTING FORC     = OPT_FORCE
    COLBACK   = COLBACK
    COLSHADOW = COLSHADOW
    PROFILE   = 'TY'.
ENDIF.

* standard graphX loop
DO.
* call graphX
  CALL FUNCTION 'GRAPH_STRUCTURAL'
    EXPORTING ADDSHAPES   = 'X'
    EDITABLE    = 'X'
    EXTACTBAR   = MEN_EXTD
    FOCUS       = 'X'
    NEWFORMAT   = FMT_R3
    PATTERN     = ''
    PWDID       = ''
    REFRESH     = ''
    SO_CONTENTS = TEXT-BSP
    SO_RECEIVER = SOF_RECV
    SO_SEND     = SOF_FLAG
    SO_TITLE    = 'TEST'(TES)
    STATUS      = DLG_STAT
    SUPER       = ''
    TBOXOPEN    = MEN_OPEN
* TBOXPOSX = 'L'
* TBOXPOSY = 'T'
* TBOXSIZEX = '20'
* TBOXSIZEY = '40'
TBOXTITLE = 'Tools'(TOL)
THELPTEXT = 'Documentation'(THX)
APPLTITLE = 'Generated model structure'(BSP)
TOOLSTITLE = 'Functions'(TOT)
TSETSTITLE = 'Subareas'(TST)
VERSION30A = FMT_R3
VIEWTITLE = 'Complete Structure'(GST)
WINID = '
* WINPOSX = 'C'
* WINPOSY = 'C'
* WINSIZEX = '80'
* WINSIZEY = '80'
IMPORTING M_TYP       = RET_MTYP
             RNID       = RET_RWNID
             TABLES OBJECTS     = TRF_OBJECTS
             EXCEPTIONS OTHERS = 3.

* refresh transfer table
REFRESH:
             TRF_OBJECTS,
             TRF_TOOLBOX,
             TRF_PROFILES.

* exit in sap office mode
IF SOF_FLAG EQ 'X'.
  EXIT.
ENDIF.

* return session
IF RET_MTYP = 'D'.
  EXIT.
ELSE.
  DLG_STAT = '4'.
ENDIF.
Calling SAP Structural Graphics

IF RET_MTYP = 'I'.
  IF BRK_MODE NE SPACE.
    BREAK-POINT 'message received'.
  ENDIF.
  CALL FUNCTION 'GET_STRU_PARAM'
    IMPORTING  C_TYP   = RET_CTYP
                ITEM    = RET_ITEM
                TABLES   OBJECTS = RET_OBJECTS
    EXCEPTIONS OTHERS = 1.
  IF SY-SUBRC NE 0.
    DLG_STAT = '3'.
    CALL FUNCTION 'GRAPH_STRU_SET_MESSAGE'
      EXPORTING MESSAGE = 'Error while enqueuing'(ENQ).
      CONTINUE.
  ENDIF.
  IF BRK_MODE NE SPACE.
    BREAK-POINT 'parameters evaluated'.
  ENDIF.
ENDIF.
CASE RET_CTYP.
  WHEN 'C'.
    CASE RET_ITEM.
      WHEN '02'.
        DLG_STAT = '3'.
        TEXTS-GTEXT = 'Again'(NOM).
        APPEND TEXTS.
        CALL FUNCTION 'GRAPH_STRU_SET_DISPLAY'
          TABLES DISPLAY = TEXTS.
      * insert object
        WHEN 'AD'.
          ADD 1 TO OBJ_NO.
          CLEAR TRF_OBJECTS.
      * propose any values
        MOVE OBJ_NO TO TRF_OBJECTS-OBJECT.
        CONDENSE TRF_OBJECTS-OBJECT NO-GAPS.
        TRF_OBJECTS-MASTER = RET_OBJECTS-OBJECT.
TRF_OBJECTS-EMPHASIZED = 'X'.
TRF_OBJECTS-OTYPE = 'AA'.
TRF_OBJECTS-OSTATUS = '9'.
TRF_OBJECTS-AUTFTEXT = 'New object'(NOB).
TRF_OBJECTS-AUTFTEXT+20 = '&%'.
MOVE OBJ_NO TO TRF_OBJECTS-AUTFTEXT+25.
CONDENSE TRF_OBJECTS-AUTFTEXT NO-GAPS.
TRF_OBJECTS-DISPTEXT = 'Typ'(TYP).
TRF_OBJECTS-DISPTEXT+10 = TRF_OBJECTS-OTYPE.
TRF_OBJECTS-DISPTEXT+12 = '&%'.
TRF_OBJECTS-DISPTEXT+15 = 'Status'(STA).
TRF_OBJECTS-DISPTEXT+25 = TRF_OBJECTS-OSTATUS.
TRF_OBJECTS-DISPTEXT+30 = '&%'.
TRF_OBJECTS-DISPTEXT+35 = TEXT-NOB.
TRF_OBJECTS-DISPTEXT+60 = '&%'.
MOVE SY-INDEX TO TRF_OBJECTS-DISPTEXT+65.
CONDENSE TRF_OBJECTS-DISPTEXT NO-GAPS.
TRF_OBJECTS-RELATION = 'ZZZ'.
TRF_OBJECTS-RELATEXT = 'Priority'(PRI).
TRF_OBJECTS-RELATEXT+15 = TRF_OBJECTS-PRIORITY.
TRF_OBJECTS-RELATEXT+20 = '&%'.
TRF_OBJECTS-RELATEXT+25 = TRF_OBJECTS-RELATION.
TRF_OBJECTS-RELATEXT+30 = '&%'.
TRF_OBJECTS-RELATEXT+32 = 'Relation'(REL).
CONDENSE TRF_OBJECTS-RELATEXT NO-GAPS.
TRF_OBJECTS-RSTATUS = TRF_OBJECTS-OSTATUS.
APPEND TRF_OBJECTS.

* trace object before inserting it

IF TRC_MODE NE SPACE.
TRC_MOMENT = MOM_INSERT.
CALL SCREEN 2000
STARTING AT 2 1.
IF TRC_SUBRC NE 0.
CLEAR TRF_OBJECTS.
REFRESH TRF_OBJECTS.
Calling SAP Structural Graphics

ENDIF.
ENDIF.

IF TRF_OBJECTS NE SPACE.
    MOVE TRF_OBJECTS TO OBJECTS.
    APPEND OBJECTS.
ELSE.
    SUBTRACT 1 FROM OBJ_NO.
ENDIF.

* set new dialog status
    DLG_STAT = '3'.
    CALL FUNCTION 'GRAPH_STRU_SET_MESSAGE'
        EXPORTING MESSAGE = 'Object inserted'(ADD).

* otherwise do nothing
WHEN OTHERS.
    CALL SCREEN 6000
        STARTING AT 3 2
        ENDING AT 37 16.
ENDCASE.

* otherwise do nothing
WHEN OTHERS.

* set new dialog status
    DLG_STAT = '3'.
    CALL FUNCTION 'GRAPH_STRU_SET_MESSAGE'
        EXPORTING MESSAGE = 'Task completed'(AAG).
ENDCASE.

ENDIF.
ENDDO.

*----------------------------------------------------------------*
*       FORM BUILD_OBJECTS                                       *
*----------------------------------------------------------------*
*       creates table with objects                               *
*----------------------------------------------------------------*
* TABLES OBJECTS
*----------------------------------------------------------------*
FORM BUILD_OBJECTS

TABLES OBJECTS STRUCTURE GSUOBJC
USING OBJ_NO.

DATA:

OBJ_MASTER LIKE GSUOBJC-MASTER VALUE ' ',
OBJ_ORIGINALTYPESET(26) VALUE 'ABCDEFGHIJKLMNOPQRSTUVWXYZ',
OBJ_TYPESET(26) VALUE 'ABCDEFGHIJKLMNOPQRSTUVWXYZ',
OBJ_ORIGINALSTATUSSET(26) VALUE '12345',
OBJ_STATUSSET(26) VALUE '12345',
OBJ_ORIGINALRELATIONSET(30) VALUE 'A0A1A2A3A4B0B1B2B3B4C0C1C2C3C4',
OBJ_RELATIONSET(30) VALUE 'A0A1A2A3A4B0B1B2B3B4C0C1C2C3C4',
OBJ_ORIGINALPRIORITYSET(10) VALUE '0945621387',
OBJ_PRIORITYSET(10) VALUE '0945621387',
MST_INDEX LIKE SY-INDEX VALUE 1,
TYP_CURRENT LIKE SY-INDEX VALUE 1,
STA_CURRENT LIKE SY-INDEX VALUE 1,
REL_CURRENT LIKE SY-INDEX VALUE 1,
PRI_CURRENT LIKE SY-INDEX VALUE 1.

* BREAK-POINT if required
IF BRK_MODE NE SPACE.
  BREAK-POINT 'build objects...'.
ENDIF.

DO OBJ_NO TIMES.

* object key
  MOVE SY-INDEX TO OBJECTS-OBJECT.
  CONDENSE OBJECTS-OBJECT NO-GAPS.

* master object
  OBJECTS-MASTER = OBJ_MASTER.
  IF MST_INDEX EQ SY-INDEX.

* different ways to define the new master object
  * MOVE SY-INDEX TO OBJ_MASTER.
    ADD 1 TO OBJ_MASTER.
    CONDENSE OBJ_MASTER NO-GAPS.
  *
  * ADD 1 TO OBJ_MASTER.

* set distance to the next master
Calling SAP Structural Graphics

ADD 3 TO MST_INDEX.
ENDIF.

* emphasized flag - used later on
OBJECTS-EMPHASIZED = SPACE.

* object type
OBJECTS-OTYPE = OBJ_TYPESET(1).
SHIFT OBJ_TYPESET.
ADD 1 TO TYP_CURRENT.
IF TYP_CURRENT GT 26.
   TYP_CURRENT = 1.
   OBJ_TYPESET = OBJ_ORIGINALTYPESET.
ENDIF.

* object status
OBJECTS-OSTATUS = OBJ_STATUSSET(1).
SHIFT OBJ_STATUSSET.
ADD 1 TO STA_CURRENT.
IF STA_CURRENT GT 5.
   STA_CURRENT = 1.
   OBJ_STATUSSET = OBJ_ORIGINALSTATUSSET.
ENDIF.

* object text in overview mode
OBJECTS-AUTFTEXT = 'Objekt'(OBJ).
OBJECTS-AUTFTEXT+20 = '&%'.
MOVE SY-INDEX TO OBJECTS-AUTFTEXT+25.
CONDENSE OBJECTS-AUTFTEXT NO-GAPS.

* object text in detail mode
OBJECTS-DISPTEXT = 'Typ'(TYP).
OBJECTS-DISPTEXT+10 = OBJECTS-OTYPE.
OBJECTS-DISPTEXT+12 = '&%'.
OBJECTS-DISPTEXT+15 = 'Status'(STA).
OBJECTS-DISPTEXT+25 = OBJECTS-OSTATUS.
OBJECTS-DISPTEXT+30 = '&%'.
OBJECTS-DISPTEXT+35 = TEXT-OBJ.
OBJECTS-DISPTEXT+60 = '&%'.
MOVE SY-INDEX TO OBJECTS-DISPTEXT+65.
CONDENSE OBJECTS-DISPTXT NO-GAPS.

* validity - not yet used
  OBJECTS-OBEGDATE = SPACE.
  OBJECTS-OENDDATE = SPACE.

* staff mode
  IF SY-INDEX = 3
    OR SY-INDEX = 7
    OR SY-INDEX = 9
    OR SY-INDEX = 13.
    OBJECTS-STAFF = 'X'.
  ELSE.
    OBJECTS-STAFF = SPACE.
  ENDIF.

* dummy flag - not used
  OBJECTS-DUMMY = SPACE.

* relation data only if master object is defined
  IF OBJECTS-MASTER NE SPACE.
  * relation type from current to master object
    OBJECTS-RELATION = OBJ_RELATIONSET(2).
    SHIFT OBJ_RELATIONSET BY 2 PLACES.
    ADD 1 TO REL_CURRENT.
    IF REL_CURRENT GT 15.
      REL_CURRENT = 1.
      OBJ_RELATIONSET = OBJ_ORIGINALRELATIONSET.
    ENDIF.

  * priority of relation
    OBJECTS-PRIORITY = OBJ_PRIORITYSET(1).
    SHIFT OBJ_PRIORITYSET.
    ADD 1 TO PRI_CURRENT.
    IF PRI_CURRENT GT 10.
      PRI_CURRENT = 1.
      OBJ_PRIORITYSET = OBJ_ORIGINALPRIORITYSET.
    ENDIF.

  * relation text
    OBJECTS-RELATEXT = 'Prio'(PRI).
Calling SAP Structural Graphics

OBJECTS-RELATEXT+15 = OBJECTS-PRIORITY.
OBJECTS-RELATEXT+20 = ' &% '.
OBJECTS-RELATEXT+25 = OBJECTS-RELATION.
OBJECTS-RELATEXT+30 = ' &% '.
OBJECTS-RELATEXT+32 = 'Verkn.'(REL).
CONDENSE OBJECTS-RELATEXT NO-GAPS.

* status of relation – in this case same as object status
OBJECTS-RSTATUS = OBJECTS-OSTATUS.

* validity – not yet used
OBJECTS-RBEGDATE = SPACE.
OBJECTS-RENDDATE = SPACE.
ENDIF.

* insert object into table and print it out
APPEND OBJECTS.
CLEAR OBJECTS.
ENDDO.

* show object list if trace mode is on
IF TRC_MODE NE SPACE.
   DESCRIBE TABLE OBJECTS
   LINES TAB_ENTRIES.
   TRC_INDEX = 1.
   CALL SCREEN 2000
      STARTING AT 2 1.
   IF TRC_SUBRC NE 0.
      STOP.
   ENDIF.
ENDIF.
ENDIF.

ENDFORM.

*----------------------------------------------------------------*
*       FORM BUILD_MENU                                          *
*----------------------------------------------------------------*
*       creates table with menu items                            *
*----------------------------------------------------------------*
*       TABLES MENU                                              *
*----------------------------------------------------------------*
FORM BUILD_MENU
  TABLES MENU STRUCTURE GSUMENU.
  * BREAK-POINT if required
  IF BRK_MODE NE SPACE.
    BREAK-POINT 'build menu...'.
  ENDIF.
  * new action bar menu for structure items
  IF FMT_R3 NE SPACE.
    MENU-MENUTITLE = 'Structure processed'(STB).
  ENDIF.
  * item RE for refreshing structure
    MENU-MENUEID = 'RE'.
    MENU-MENUETEXT = 'Updating' (REF).
    MENU-ACTIVATED = '0'.
    APPEND MENU.
    CLEAR MENU.
  * other action bar menu for object items
    IF FMT_R3 NE SPACE.
    MENU-MENUTITLE = 'Object processed'(OBB).
  ENDIF.
  * item AD for inserting objects
    MENU-MENUEID = 'AD'.
    MENU-MENUETEXT = 'Inserting object'(INS).
    MENU-ACTIVATED = '1'.
    MENU-OBJFIELDS = 'X                 '.
    APPEND MENU.
    CLEAR MENU.
  * reserved item 01 for moving objects
    MENU-MENUEID = '01'.
    MENU-MENUETEXT = 'Moving object'(MOV).
    MENU-ACTIVATED = '2'.
    MENU-OBJFIELDS = 'X X                 X  '.
    APPEND MENU.
    CLEAR MENU.
Calling SAP Structural Graphics

* reserved item 02 for displaying object data
  MENU-MENUEID = '02'.
  MENU-MENUTEXT = 'Displaying object data'(DAT).
  MENU-ACTIVATED = '1'.
  MENU-OBJFIELDS = 'X '.
  APPEND MENU.
  CLEAR MENU.

* show toolbox items if trace mode is on
  IF TRC_MODE NE SPACE.
    DESCRIBE TABLE MENU
    LINES TAB_ENTRIES.
    TRC_INDEX = 1.
    CALL SCREEN 3000
      STARTING AT 2 1.
    IF TRC_SUBRC NE 0.
      STOP.
    ENDIF.
  ENDIF.
ENDFORM.

*----------------------------------------------------------------*
*       FORM BUILD_PROFILES                                       *
*----------------------------------------------------------------*
*       creates table with profiles and rules                     *
*----------------------------------------------------------------*
*       TABLES PROFILES                                          *
*----------------------------------------------------------------*

FORM BUILD_PROFILES
  TABLES PROFILES STRUCTURE GSUPROF.
  * BREAK-POINT if required
  IF BRK_MODE NE SPACE.
    BREAK-POINT 'build profiles...'.
  ENDIF.
  * types profile
    PROFILES-PROFILEID = 'TY'.

ENDFORM.
PROFILES-PROFILETXT = 'Type-oriented'(PTY).
IF FMT_R3 NE SPACE.
   PROFILES-RULEID = '01'.
ENDIF.
PROFILES-OBJFIELD = 'OTYPE'.
PROFILES-CONTENTS = 'A'.
PROFILES-LINECOLOUR = '01'.
PROFILES-LINESTYLE = '00'.
PROFILES-TEXTCOLOUR = '00'.
PROFILES-BOXCOLOUR = '12'.
PROFILES-SHAPE = '00'.
APPEND PROFILES.
CLEAR PROFILES.
PROFILES-PROFILEID = 'TY'.
IF FMT_R3 NE SPACE.
   PROFILES-RULEID = '02'.
ENDIF.
PROFILES-OBJFIELD = 'OTYPE'.
PROFILES-CONTENTS = 'B'.
PROFILES-LINECOLOUR = '02'.
PROFILES-LINESTYLE = '01'.
PROFILES-TEXTCOLOUR = '00'.
PROFILES-BOXCOLOUR = '13'.
PROFILES-SHAPE = '01'.
APPEND PROFILES.
CLEAR PROFILES.
PROFILES-PROFILEID = 'TY'.
IF FMT_R3 NE SPACE.
   PROFILES-RULEID = '03'.
ENDIF.
PROFILES-OBJFIELD = 'OTYPE'.
PROFILES-CONTENTS = 'C'.
PROFILES-LINECOLOUR = '03'.
PROFILES-LINESTYLE = '02'.
PROFILES-TEXTCOLOUR = '00'.
Calling SAP Structural Graphics

    PROFILES-BOXCOLOUR = '14'.
    PROFILES-SHAPE = '02'.
    APPEND PROFILES.
    CLEAR PROFILES.
    PROFILES-PROFILEID = 'TY'.
    IF FMT_R3 NE SPACE.
        PROFILES-RULEID = '04'.
    ENDIF.
    PROFILES-OBJFIELD = 'RELATION'.
    PROFILES-CONTENTS = 'A0'.
    PROFILES-LINECOLOUR = '04'.
    PROFILES-LINESTYLE = '03'.
    PROFILES-TEXTCOLOUR = '00'.
    PROFILES-BOXCOLOUR = '09'.
    PROFILES-SHAPE = '03'.
    APPEND PROFILES.
    CLEAR PROFILES.
    PROFILES-PROFILEID = 'TY'.
    IF FMT_R3 NE SPACE.
        PROFILES-RULEID = '05'.
    ENDIF.
    PROFILES-OBJFIELD = 'RELATION'.
    PROFILES-CONTENTS = 'B0'.
    PROFILES-LINECOLOUR = '05'.
    PROFILES-LINESTYLE = '04'.
    PROFILES-TEXTCOLOUR = '00'.
    PROFILES-BOXCOLOUR = '10'.
    PROFILES-SHAPE = '04'.
    APPEND PROFILES.
    CLEAR PROFILES.
    PROFILES-PROFILEID = 'TY'.
    IF FMT_R3 NE SPACE.
        PROFILES-RULEID = '06'.
    ENDIF.
    PROFILES-OBJFIELD = 'RELATION'.
PROFILES-CONTENTS  =  'C0'.
PROFILES-LINECOLOUR =  '06'.
PROFILES-LINESTYLE  =  '05'.
PROFILES-TEXTCOLOUR =  '00'.
PROFILES-BOXCOLOUR  =  '11'.
PROFILES-SHAPE    =  '05'.
APPEND PROFILES.
CLEAR PROFILES.

* status profile
  PROFILES-PROFILEID   =  'ST'.
  PROFILES-PROFILETXT = 'Status-oriented'(PST).
  IF FMT_R3 NE SPACE.
    PROFILES-RULEID =  '01'.
  ENDIF.
  PROFILES-OBJFIELD   =  'OSTATUS'.
  PROFILES-CONTENTS   =  '1'.
  PROFILES-LINECOLOUR =  '01'.
  PROFILES-LINESTYLE  =  '06'.
  PROFILES-TEXTCOLOUR =  '00'.
  PROFILES-BOXCOLOUR  =  '12'.
  PROFILES-SHAPE    =  '06'.
  APPEND PROFILES.
  CLEAR PROFILES.
  PROFILES-PROFILEID   =  'ST'.
  IF FMT_R3 NE SPACE.
    PROFILES-RULEID =  '02'.
  ENDIF.
  PROFILES-OBJFIELD   =  'OSTATUS'.
  PROFILES-CONTENTS   =  '2'.
  PROFILES-LINECOLOUR =  '02'.
  PROFILES-LINESTYLE  =  '07'.
  PROFILES-TEXTCOLOUR =  '00'.
  PROFILES-BOXCOLOUR  =  '13'.
  PROFILES-SHAPE    =  '07'.
  APPEND PROFILES.
Calling SAP Structural Graphics

CLEAR PROFILES.
PROFILES-PROFILEID = 'ST'.
IF FMT_R3 NE SPACE.
    PROFILES-RULEID = '03'.
ENDIF.

PROFILES-OBJFIELD = 'OSTATUS'.
PROFILES-CONTENTS = '3'.
PROFILES-LINECOLOUR = '03'.
PROFILES-LINESTYLE = '08'.
PROFILES-TEXTCOLOUR = '00'.
PROFILES-BOXCOLOUR = '14'.
PROFILES-SHAPE = '08'.
APPEND PROFILES.
CLEAR PROFILES.

PROFILES-PROFILEID = 'ST'.
IF FMT_R3 NE SPACE.
    PROFILES-RULEID = '04'.
ENDIF.

PROFILES-OBJFIELD = 'RSTATUS'.
PROFILES-CONTENTS = '3'.
PROFILES-LINECOLOUR = '04'.
PROFILES-LINESTYLE = '09'.
PROFILES-TEXTCOLOUR = '00'.
PROFILES-BOXCOLOUR = '15'.
PROFILES-SHAPE = '09'.
APPEND PROFILES.
CLEAR PROFILES.

PROFILES-PROFILEID = 'ST'.
IF FMT_R3 NE SPACE.
    PROFILES-RULEID = '05'.
ENDIF.

PROFILES-OBJFIELD = 'RSTATUS'.
PROFILES-CONTENTS = '4'.
PROFILES-LINECOLOUR = '05'.
PROFILES-LINESTYLE = '01'.

236  April 2001
PROFILES-TEXTCOLOUR = '00'.
PROFILES-BOXCOLOUR = '09'.
PROFILES-SHAPE = '10'.
APPEND PROFILES.
CLEAR PROFILES.

PROFILES-PROFILEID = 'ST'.
IF FMT_R3 NE SPACE.
  PROFILES-RULEID = '06'.
ENDIF.

PROFILES-OBJFIELD = 'RSTATUS'.
PROFILES-CONTENTS = '5'.
PROFILES-LINECOLOUR = '06'.
PROFILES-LINESTYLE = '02'.
PROFILES-TEXTCOLOUR = '00'.
PROFILES-BOXCOLOUR = '10'.
PROFILES-SHAPE = '11'.
APPEND PROFILES.
CLEAR PROFILES.

* priority profile
IF FMT_R3 NE SPACE.
  PROFILES-PROFILEID = 'PR'.
  PROFILES-PROFILETXT = 'Priority'(PPR).
  PROFILES-RULEID = '01'.
  PROFILES-OBJFIELD = 'PRIORITY'.
  PROFILES-CONTENTS = '0'.
  PROFILES-LINECOLOUR = '01'.
  PROFILES-LINESTYLE = '03'.
  PROFILES-TEXTCOLOUR = '00'.
  PROFILES-BOXCOLOUR = '11'.
  PROFILES-SHAPE = '12'.
APPEND PROFILES.
CLEAR PROFILES.
PROFILES-PROFILEID = 'PR'.
PROFILES-RULEID = '02'.
PROFILES-OBJFIELD = 'PRIORITY'.

April 2001  237
Calling SAP Structural Graphics

   PROFILES-CONTENTS = '2'.
   PROFILES-LINECOLOUR = '02'.
   PROFILES-LINESTYLE = '04'.
   PROFILES-TEXTCOLOUR = '00'.
   PROFILES-BOXCOLOUR = '12'.
   PROFILES-SHAPE = '00'.
   APPEND PROFILES.
   CLEAR PROFILES.
   PROFILES-PROFILEID = 'PR'.
   PROFILES-RULEID = '03'.
   PROFILES-OBJFIELD = 'PRIORITY'.
   PROFILES-CONTENTS = '4'.
   PROFILES-LINECOLOUR = '03'.
   PROFILES-LINESTYLE = '05'.
   PROFILES-TEXTCOLOUR = '00'.
   PROFILES-BOXCOLOUR = '13'.
   PROFILES-SHAPE = '01'.
   APPEND PROFILES.
   CLEAR PROFILES.
ENDIF.

* show menu items if trace mode is on
   IF TRC_MODE NE SPACE.
      DESCRIBE TABLE PROFILES
         LINES TAB_ENTRIES.
      TRC_INDEX = 1.
      CALL SCREEN 4000
         STARTING AT 2 1.
      IF TRC_SUBRC NE 0.
         STOP.
      ENDIF.
   ENDIF.
ENDIF.
ENDFORM.
ENDFORM.

*-----------------------------------------------------------------------------------------------------*
* FORM BUILD_TEXTS
FORM BUILD_TEXTS
  TABLES TEXTS STRUCTURE GSUDISP.
  DATA TXT_COUNT(3) VALUE '0'.
  * BREAK-POINT if required
  IF BRK_MODE NE SPACE.
      BREAK-POINT 'build texts...'.
  ENDIF.
  * fill text lines in table
    TEXTS-GTEXT = 'Version:'(T00).
    TEXTS-GTEXT+20 = TXT_COUNT.
    CONDENSE TEXTS-GTEXT.
    ADD 1 TO TXT_COUNT.
    TEXTS-GTEXT = 'This is a '(T01).
    APPEND TEXTS.
    TEXTS-GTEXT = 'a sample'(T02).
    APPEND TEXTS.
    TEXTS-GTEXT = 'text,'(T03).
    APPEND TEXTS.
    TEXTS-GTEXT = 'that is to show'(T04).
    APPEND TEXTS.
    TEXTS-GTEXT = 'that this window'(T05).
    APPEND TEXTS.
    TEXTS-GTEXT = 'can be called.'(T06).
    APPEND TEXTS.
  * show texts if trace mode is on
    IF TRC_MODE NE SPACE.
        DESCRIBE TABLE TEXTS
        LINES TAB_ENTRIES.
        TRC_INDEX = 1.
        CALL SCREEN 5000
Calling SAP Structural Graphics

    STARTING AT 3 2
    ENDING AT 70 16.
    IF TRC_SUBRC NE 0.
        STOP.
    ENDF.
    ENDF.
ENDFORM.

*----------------------------------------------------------------*
*       MODULE PBO_2000                                          *
*----------------------------------------------------------------*
*       PBO module dynpro 2000 - object list                     *
*----------------------------------------------------------------*
MODULE PBO_2000 OUTPUT.
    CASE TRC_MOMENT.
    * at the beginning
        WHEN MOM_START.
            SET PF-STATUS 'STANDARD'.
            READ TABLE OBJECTS
                INDEX TRC_INDEX.
            MOVE OBJECTS TO GSUOBJC.
        * when objects should be inserted
            WHEN MOM_INSERT.
                SET PF-STATUS 'INSERT'.
                MOVE TRF_OBJECTS TO GSUOBJC.
        * stop when no moment is set
            WHEN OTHERS.
                SET SCREEN 0.
                LEAVE SCREEN.
        ENDCASE.
ENDMODULE.

*----------------------------------------------------------------*
*       MODULE PBO_3000                                          *
*----------------------------------------------------------------*
* PBO module dynpro 3000 - menu items *
*----------------------------------------------------------------------------*

MODULE PBO_3000 OUTPUT.
  SET PF-STATUS 'STANDARD'.
  READ TABLE MENU
    INDEX TRC_INDEX.
  MOVE MENU TO GSUMENU.
ENDMODULE.

*----------------------------------------------------------------------------*
* MODULE PBO_4000 *
*----------------------------------------------------------------------------*
* PBO module dynpro 4000 - profiles *
*----------------------------------------------------------------------------*

MODULE PBO_4000 OUTPUT.
  SET PF-STATUS 'STANDARD'.
  READ TABLE PROFILES
    INDEX TRC_INDEX.
  MOVE PROFILES TO GSUPROF.
ENDMODULE.

*----------------------------------------------------------------------------*
* MODULE PBO_5000 *
*----------------------------------------------------------------------------*
* PBO module dynpro 5000 - texts *
*----------------------------------------------------------------------------*

MODULE PBO_5000 OUTPUT.
  SET PF-STATUS 'STANDARD'.
ENDMODULE.

*----------------------------------------------------------------------------*
* MODULE PBO_6000 *
*----------------------------------------------------------------------------*
* PBO module dynpro 6000 - texts *
*----------------------------------------------------------------------------*
Calling SAP Structural Graphics

MODULE PBO_6000 OUTPUT.
    SET PF-STATUS 'STANDARD'.
ENDMODULE.

*---------------------------------------------------------------*
*       read texts for dynpro 6000                              *
*       process ok_code for all dynpros                        *
*---------------------------------------------------------------*
MODULE READ_TABLE_5000 OUTPUT.
    DATA PBO_STARTINDEX LIKE SY-INDEX.
    PBO_STARTINDEX = SY-STEPL + TRC_INDEX - 1.
    READ TABLE TEXTS
        INDEX PBO_STARTINDEX.
    IF SY-SUBRC EQ 0.
        MOVE TEXTS TO GSUDISP.
    ENDIF.
ENDMODULE.

*---------------------------------------------------------------*
*       read texts for dynpro 6000                              *
*---------------------------------------------------------------*
MODULE READ_TABLE_6000 OUTPUT.
    DATA PBO_STARTINDEX2 LIKE SY-INDEX.
    PBO_STARTINDEX2 = SY-STEPL + TRC_INDEX - 1.
    READ TABLE RET_OBJECTS
        INDEX PBO_STARTINDEX2.
ENDMODULE.

*---------------------------------------------------------------*
*       process ok_code for all dynpros                         *
*---------------------------------------------------------------*
MODULE OK_CODE INPUT.

   TRC_SUBRC = 4.

   CASE OK_CODE.
*   WHEN 'TRAC'.
*     IF TRC_MODE NE SPACE.
*       TRC_MODE = ' '.
*     ELSE.
*       TRC_MODE = 'X'.
*     ENDIF.
*   ENDIF.

   WHEN 'SAVE'.
    CASE TRC_MOMENT.
     WHEN MOM_INSERT.
      MOVE GSUOBJC TO TRF_OBJECTS.
      MODIFY TRF_OBJECTS INDEX 1.
     WHEN OTHERS.
      ENDCASE.
   ENDW.

   WHEN 'GO  '.
    TRC_SUBRC = 0.
    SET SCREEN 0.
    LEAVE SCREEN.

   WHEN 'BACK'.
    SET SCREEN 0.
    LEAVE SCREEN.

   WHEN 'STOP'.
    SET SCREEN 0.
    LEAVE SCREEN.

   WHEN 'END '.
    SET SCREEN 0.
    LEAVE SCREEN.

   WHEN 'P++ '
     TRC_INDEX = TAB_ENTRIES.
   WHEN 'P+  '.
     IF TRC_INDEX < TAB_ENTRIES.
       ADD 1 TO TRC_INDEX.
Calling SAP Structural Graphics

ENDIF.

WHEN 'P--'.

IF TRC_INDEX > 1.

SUBTRACT 1 FROM TRC_INDEX.

ENDIF.

WHEN 'P--'.

TRC_INDEX = 1.

WHEN OTHERS.

ENDCASE.

ENDMODULE.