Rtt Editor

User's Guide

96 04 15

Introduction 5
Working with rtt editor 6
Start the editor 6
Menu editor 6
Main title 6
Title prefix 6
Setup 6
Create menu entries 7
Creating a submenu 7
Menu item types 7
menu 7
picture 7
permpicture 7
function 7
exit 8
objecthierarchy 8
command 8
commandhold 8
keys 8
vmscommand 8
vmshold 9
vmsconfirm 9
vmsnowait 9
objpicture 9
syspicture 9
Remove menu items 9
Relocation of menu items 10
Modify a menu item 10
Import and export of menu-tree between programs 10
Import the standard menues 10
Edit pictures 10
Inverse mode 10
Line graphics 10
Create an update field 11
Edit an update field 11
Analog attributes 11
Bar 12
Boolean attributes 12
Text 12
Input fields 13
Not changeable fields 14
Inverted fields 14
Pushbuttons 14
Objid 15
Time 15
Open a picture for a pushbutton 15
Modify a field without opening it 16
Remove a field 16
Cut-Copy-Paste 16
Edit fields in a text editor 16
Copy pictures between programs 16
Search for attributes 16
Helptexts 17
Save 17
Create a rtt-program 17
Quit 17
Run the program 17
Menues 18
Menu mode 18
Create menues and menu items 18
Delete menu items 19

Move menu items....19
Picture editor....20
Edit mode....20
Edit an update field....20
Update field data....21
Bar....23
Field displaying current time....24
Fields displaying alarms....24
Pictures with code behind....25
Local variables....25
Function file....25
Function keys....27
Fastkey menu....27
Helptexts....28
System pictures....29
Object pictures....30
Commands....31
Picture editor commands....31
clear picture....31
clear items....31
connect....31
copy....31
create....31
cut....31
delete....31
dualconnect....31
include items....32
include picture....32
modify....32
paste....32
save....32
select....32
set....32
unselect....33
write items....33
write picture....33
Menu editor....33
create....33
include menu....33
modify....33
show....34
undo delete....34
write menu....34
Common commands....34
compile....34
edit....34
exit....35
export gdhreflist....35
export externref....35
link....35
quit....35
save....35
setup....35
show collection....35
Appendix A....37

Introduction

Pwr_rtt_edit is an editor for building pictures for maintenance and simple operator communication in Proview.

Pwr_rtt_edit includes a tool for building menus of the type used in object hierarchies in rtt, and an editor for building images that present values of attributes in the real-time database.

Pwr_rtt_edit is started with the symbol pwr_rtt_edit with an appropriate program name as argument. The RTT program that is created will be named rt_rtt_'programnamn '.

Pwr_rtt_edit requires two logical names are defined: $pwrp_rtt and $pwrp_rttbld. These should point to directories in the project tree. $pwrp_rtt will contain the source code for menus and pictures, and $pwrp_rttbld various build files.

Working with rtt editor

This is an introduction to get started and work with rtt-editor.

Start the editor

Start with the command

```
> wb_rtt
```

The editor than asks for 'Program'. Program is a name that will be included in any file that is generated. A rtt program generated with the program-name 'vhxn2r' will be named $pwrp_exe/rt_rtt_vhxn2r.

The program name can also be added as an argument to the start command.

```
> wb_rtt vhxn2r
```

Menu editor

When the program name is specified, you will enter the menu editor, where menu entries for pictutres, submenus etc are created.

Created menu items are viewed in the editor, as they will be viewed in the rtt program, with the difference that the item type is written in brackets after the menu text. The top menu is called main menu. It contains a head title, a menu entry, and a not yet visible title prefix in the upper left corner.

Ctrl/B opens a prompt where commands are typed. With the ArrowUp key, old commands can be reused and modified.

Main title

The default main title is "RTT IN 'program'". It is modified with the command 'modify /maintitle'
If you for example want it in Swedish, you can write

```
rtt_edit> mod /maintitle = "RTT i VHXN2R"
```

If the title contains spaces or lowercase letters, it should be enclosed by quotation marks.

Title prefix

Title Prefix is a text that is printed in the upper left corner of each page of the RTT (except edited picture pages). The prefix should contain the system and node-names so that one can easily see the current node. If you are editing a program that is going to be run on different nodes, it is possible to use the symbol RTT_NODE and RTT_SYS that contains the current node and system name respectively. The prefix is changed with the command

```
rtt_edit> mod /titleprefix = "RTT#'RTT_NODE#'-#'RTT_SYS#'"
```

The # is used to avoid the symbols to be converted at edit time. The title prefix will contain the text "RTT'RTT_NODE'-'RTT_SYS ' which at runtime will be converted to eg "RTT VHXOP4-VHXN2R".

Setup

An easier way to insert the maintitle and titleprefix is to enter the setup with the command

```
rtt_edit> setup
```

Select the 'Main Title' or 'Title prefix' and enter the value with Ctrl/E of F3. Here you do not use the # sign as in the example above.

Also enter the platforms that RTT program will be built on. In the field the 'Operating System' you add the code for the platform (or the sum of the codes if you want to build for multiple platforms).

Create menu entries

When editing a program but the first time you get a dummy menu item named “'programname' MENU '”, eg VHXN2R MENU. New menu items are created with the 'create' command.

```
rtt_edit> create /menu MAINTENANCE
```

New menu items are created below the selected item. If you want to create a menu item at the top, it has to be created below the top item and then you move down the top item.

Creating a submenu

One can build menu trees by creating sub-menus with the command 'create /child'. Submenus can be created to menu items of type 'menu'. Select an entry of type 'menu' and type the command

```
rtt_edit> create /menu /child "itemname"
```

Pressing return opens it in the menu, which now contains a menu entry. More menu items in the submenu is created the same way as in the main menu. Use Ctrl/R to return to the previous menu.

Menu item types

There are a number of different types of menyentryn.

menu

menu contains a submenu and is created with the command

```
rtt_edit> create / menu "entry name"
```

picture

picture contains a edited picture

```
rtt_edit> create / picture "picture name"
```

permpicture

Permpicture is like a picture an edited picture. The difference is that a permpicture retain its subscriptions if you leave the picture, and have them available next time you enter the picture. A normal picture deletes the subscription and have to tie them up again each time the picture is entered.
A picture that is often used should be made as a permpicture.

```
rtt_edit> create /permpicture 'picturename'
```

function

Function is an edited picture with user written code behind. In create-command a c-function is specified which will be added by the constructor in the file $pwrp_rtt/ra_rtt_'program'.c. The c-function is called

when the picture initiated, terminated, when the data in the image is changed, and for cyclic updating of the picture.

```
rtt_edit> create /picture /function=VHXN2R_ZON1 "Zone 1"
```

The current function for a menu item is viewed with the command

```
rtt_edit> show function
```

exit

Exit is a menu entry which terminates the execution of the RTT.

```
rtt_edit> create / exit "EXIT"
```

objecthierarchy

Objecthiearchy shows the Proview database.

```
rtt_edit> create /objecthierarchy “DATABASE"
```

command

Command executes a rtt command specified at the creation of the menu item. The following menu item will show the alarm list.

```
rtt_edit> create /command="show alarms" "Alarm List"
```

You can also set attributes in the database from a menu entry

```
rtt_edit> crea / command = "set param /name=vhxn2r-zon1-start.ActualValue
/value=1" "Start the furnish"
```

The current command for a menu item is showed with the command

```
rtt_edit> show command
```

commandhold

A variation on the 'command' item to use for function keys if you want to keep the current picture. If one has a 'command' entry defined as a function key, the current picture will be left before the command is executed (except for the command 'set'). With commandhold the current picture i kept and will also not be redrawn when the command is executed.

keys

Keys contains a submenu with menu items which can be activated from the function keys on a rtt keyboard.
The order of the menu items determine what function key they are connected to. The top menu item is connected to F1, the second-highest to F2 and so on. The keys menu item has to be positioned in the main menu.

```
rtt_edit> create /key "Function Keys"
```

vmscommand

vmscommand executes a shell command.

```
rtt_edit> create /vmscommand="logg_list 1" "List Logging"
```

The current shell command for a menu entry is displayed with the command

```
rtt_edit> show vms
```

vmshold

Vmshold is the same as vmscommand except for the fact that the current rtt menu or picture is not redrawn when the shell command has completed execution. A printout that is generated from a script file that is called, will be displayed at the bottom line of the terminal window. In this way it is possible to print error messages or information in the shell scripts.

```
rtt_edit> create /vmshold="logg_list 2" "List more logging"
```

vmsconfirm

Vmsconfirm executes a command, but requires only a user confirmation. The user confirm by pressing the PF1 and stop by pressing PF4.

```
rtt_edit> create /vmsconfirm="logg_list 3" "List more ..."
```

vmsnowait

Vmsnowait is another variant of executing a shell command, where the session does not wait for the command to finish. The command may continue to execute in a detached process, while the RTT program continues to run.

```
rtt_edit> create /nowait=”logg_list 4" "List in a separate window"
```

objpicture

objpicture are object pictures, i.e. pictures that presents an object of a certain class. Object images are available for the classes PID and Av. In addition to the object name, the name of the object picture (RTTSYS_OBJECT_'class') is stated in the create command.

```
rtt_edit> create /objpicture=rttsys_object_pid /name=vhxn2r-zon1-tempreg
"Temperature Controllers Zone 1"
```

You can view the current object picture for a menu item with the command

```
rtt_edit> show objpicture
```

Object name is viewed with

```
rtt_edit> show name
```

syspicture

syspicture are pictures showing information about a proview system, e.g. which nodes you have contact with (SHOW NODES).

```
rtt_edit> create /syspicture=rttsys_show_nodes "SHOW NODES"
```

You can view the current system picture for a menu item with the command

```
rtt_edit> show syspicture
```

Remove menu items

A menu item is be deleted by selecting it and pressing the backspace key. If the menu entry contains a submenu, this will also be deleted. The most recent delete operation can be re-created with "undo delete" command.

```
rtt_edit> undo delete
```

Relocation of menu items

Delete - 'undo delete' functions can be used to move a menu item as "undo delete" recreates the previously removed menu item below the selected item.

Modify a menu item

You can change the text of a menu item with the modify function. Select menu item and enter the new text.

```
rtt_edit> modify "New Name"
```

The type of a menu entry can not be changed. Nor can the commands or functions. In this case, a new menu item has to be created.

Import and export of menu-tree between programs

A menu-tree may be written to a text file and loaded again with the commands

```
rtt_edit> write the menu "filename"
rtt_edit> menu include "filename"
```

Import the standard menues

The standard version of RTT contains a number of menus with system pictures which also should be included in project-specific rtt programs. By including the file file rtt_menu_template.dtt_m these menus are included.

```
rtt_edit> include menu rtt_menu_template
```

Edit pictures

If you open a menu entry of type picture, permpicture or function, the picture editor is opened. In this you can write texts, draw line graphics and create update fields. The work area is 80 characters wide and 22 characters high. You move the cursor with the arrow keys. As in the menu editor, commands are inserted with Ctrl/B.

Inverse mode

With the command "set inverse" you can write characters in inverse mode, i.e. white on black background. Spaces are black boxes that can be used as graphics. Both text and line graphics can be printed in inverse mode.

```
rtt_edit> set inverse
```

With 'set noinverse' you leave the inverse mode.

```
rtt_edit> set noinverse
```

Line graphics

With 'set line' you can draw line graphics. See figure which characters are useful.

```
rtt_edit> set line
```

Return to text using the command

```
rtt_edit> set ascii
```

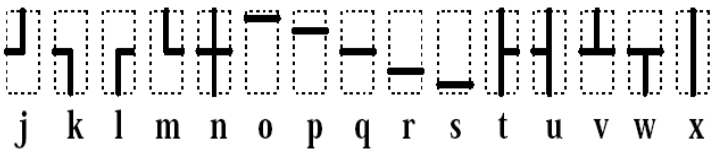

Fig Line graphics

Create an update field

An updated field is connected to an attribute in the Proview database. If you edit a function picture (a menu item of type function) you can also connect the field to a variable in an internal database.

An update field is created by positioning the cursor where the field is to be placed, and enter the command

```
rtt_edit> create %
```

The field is marked with '>' followed by a number.

Edit an update field

You can edit data for an update field by placing the cursor on the first position of the field en press Ctrl/A. Select the field property that is to be modified and enter a new value with the 'modify' command.

Analog attributes

Assume that we want to show the value of an Av object in the field. In the 'Parameter' property, the name of the Av with the attribute (ActualValue), should be stated. Select 'Parameter' with the arrow keys and write the command

```
rtt_edit> mod “vhx-Zon1-Temperature.ActualValue”
```

The name can be copied from the configurator. Also state the size of the field in 'Characters' and the number of decimals in 'Decimals'.

Example

Number	1
Text	%
Type	UPDATE
Parameter	vhx-Zon1-Temperature.ActualValue
Text/Dualpar	
Privileges	NO
Outputflags	
Characters	6
Decimals	2
Maxlimit	0.000000
Minlimit	0.000000
Database	GDH
Declaration	

Bar

An analog attribute can be displayed as a horizontal bar, by setting 'Outputflags' to 'BAR'. Also state the min and max values for the bar in 'MinLimit' and 'MaxLimit', and the length of the bar int 'Characters'.

Example

A 20 character long bar, 0-500 C.

Number	1
Text	%
Type	UPDATE
Parameter	vhx-Zon1-Temperature.ActualValue
Text/Dualpar	
Privileges	NO
Outputflags	BAR
Characters	10
Decimals	0
Maxlimit	500.000000
Minlimit	0.000000
Database	GDH
Declaration	

Boolean attributes

Assume that we want to show the value of a Dv object in the field. The name is stated in 'Parameter' with the attribute (ActualValue)

Example

Number	1
Text	%
Type	UPDATE
Parameter	vhx-Zon1-Enable.ActualValue
Text/Dualpar	
Privileges	NO
Outputflags	
Characters	1
Decimals	1
Maxlimit	0.000000
Minlimit	0.000000
Database	GDH
Declaration	

Text

The property 'Outputflags' states how the field is presented in the picture. If empty, the value is presented as a digit, i.e. 0 or 1 for digital attributes. By changing 'Outputflags' you can get the output in shape of a text (“On/Off”, “True/False”, “Auto/Man”).

You can also state the text by entering “TEXT” in 'Outputflags' and state the text in “Text/Dualpar'. A '/' sign separates what is displayed when the value i 1 and 0 respectively.

For example, by entering “To/From” in 'Text/Dualpar' “To” is written when the Dv value is 1, and “From” when the value is 0.

An exclamation mark indicates that the text should be written in inverted mode, e.g. “!To/From” will imply that “To” is written in inverted mode, and “From” in normal mode.

The prefix _L_ indicates that the text is written as line graphics. In the text, ascii characters corresponding to the line graphic signs are written. For example “_L_qqqq/_L_xxxx” will be written as “----” when the value is 1, and “||||” when the value is 0.

Also state in 'Characters' the size of the field, i.e. the longest of the two texts.

Flashing text is achieved by stating “FLASHTEXT” in 'Outputflags'.

Example

Number	1
Text	%
Type	UPDATE
Parameter	vhx-Zon1-Enable.ActualValue
Text/Dualpar	!Zon 1 enabled/Zon 1 disabled
Privileges	NO
Outputflags	TEXT
Characters	14
Decimals	0
Maxlimit	0.000000
Minlimit	0.000000
Database	GDH
Declaration	

Input fields

In the property 'Text' you can specify a text that will be written in front of the field. This text is used when you want to change the value of the attribute in the field. The text will be inverted when the field is selected, and with the arrow keys it is possible to move the selection between different input fields. The selection order is decided from the content of 'Number'. By pressing ArrowUp or ArrowDown the field with the closest higher or lower value, respectively, will be selected.

When the desired field is selected, it is possible to change the value by pressing F3 and enter a new value. In 'Privileges' is stated who is authorized to change the value, and in 'Maxlimit' and 'Minlimit' the max and min values for the input.

Example

Number	3
Text	Temperatur Setpoint
Type	UPDATE
Parameter	vhx-Zon1-TempSetPoint.ActualValue
Text/Dualpar	
Privileges	OP
Outputflags	
Characters	6
Decimals	2
Maxlimit	500.000000
Minlimit	0.000000
Database	GDH
Declaration	

Not changeable fields

If the field is not an input field, i.e. you should not be able to select the field, 'Text' should be set to “%” and 'Privileges' to “NO”.

Inverted fields

By setting an exclamation mark in “Text/Dualpar' the field is written in inverse mode.

Example

Number	1
Text	%
Type	UPDATE
Parameter	vhx-Zon1-Fan-Speed.ActualValue
Text/Dualpar	!
Privileges	NO
Outputflags	
Characters	5
Decimals	2
Maxlimit	0.000000
Minlimit	0.000000
Database	GDH
Declaration	

Pushbuttons

The value of a digital field can be changed by selecting it, and pressing Ctrl/A or Ctrl/T. The available functions are

- SET, set the value to 1 with Ctrl/A.
- RESET, reset the value with Ctrl/A.
- SET_RESET, set the value with Ctrl/A and reset with Ctrl/T.
- TOGGLE, toggle the value with PF1.

If the value of the attribute should not be viewed, you set the 'Outputflags' to “NO”.

Example
Set the Dv StartEngine with Ctrl/A

Number	1
Text	Start engine
Type	SET
Parameter	Vhx-Zon1-StartEngine.ActualValue
Text/Dualpar	
Privileges	OP
Outputflags	NO
Characters	0
Decimals	0
Maxlimit	0.000000
Minlimit	0.000000
Database	GDH
Declaration	

Objid

Attributes of type objid, e.g. dataobjects in Nmps cells, are displayed with the object name. If 'Decimals' is set to 1, only the last segment of the name is shown.

Example

Number	1
Text	%
Type	UPDATE
Parameter	Vhx-Tracking-W-Cell.Data1_Objid
Text/Dualpar	
Privileges	NO
Outputflags	
Characters	10
Decimals	1
Maxlimit	0.000000
Minlimit	0.000000
Database	GDH
Declaration	

Time

Attributes of type pwr_tTime are converted to a string with format “DD-MM-YYYY HH:MM:SS.hh”, e.g. 30-MAR-1999 12:35:10.40. If you set decimals to 1, only the time, not the date, is viewed.

Open a picture for a pushbutton

You can create a pushbutton that executes a rtt command when Ctrl/A is pressed. The command 'show menu' can be used to open another picture from the pushbutton. Set “COMMAND” in the 'Type' property and the rtt-command in 'Text/Dualpar'.

If the expression inserted into 'Text/Dualpar' contains spaces, it should be enclosed by quotation marks. Though, if you want the quotation marks to stay in the rtt command, write \” instead of “.

Example

Number	1
Text	Show pumps
Type	COMMAND
Parameter	
Text/Dualpar	Show menu “Maintenance-Pumps-Pumps Overview”
Privileges	OP
Outputflags	NO
Characters	0
Decimals	0
Maxlimit	0.000000
Minlimit	0.000000
Database	USER
Declaration	

This pushbutton will open the picture Maintenance-Pumps-Pumps Overview. The command to enter the command into 'Text/Dualpar' is

```
rtt_edit> mod “show menu \”Maintenance-Pumps-Pumps Overview\””
```

Modify a field without opening it

You don't have to open a field to modify it. By placing the cursor on the first position of the field, you can with the 'modify' command change properties of the field. This is useful when you connect database attributes to the field, or set the same data in serveral fields.

```
rtt_edit> mod/par=”vhx-Zon1-Temperature.ActualValue”
rtt_edit> mod/char=7
```

Remove a field

A field is delete by placing the cursor on the field and entering the command 'delete item'.

rtt_edit> delete item

Cut-Copy-Paste

An area in the picture editor can be selected by placing the cursor in one corner, press Ctrl/T, and move to the other corner. The selected area is drawn inverted.

You can copy the selected area to the paste buffer by pressing Ctrl/E (or command 'copy'). The command 'cut' will also clear the selected area. The content of the paste buffer is copied to the current cursor position with Ctrl/F or command 'paste'. It is also possible to copy between different pictures in the same program (to copy between programs, the content has to be exported to file).

```
rtt_edit> cut
rtt_edit> paste
```

Reset of the selected area is done with the 'unselect' command

```
rtt_edit> unselect
```

Edit fields in a text editor

If there are many similar fields in a picture, the picture content can be written to a text file, which can be edited and then imported. The command

rtt_edit> write items item.txt

write the fields to the file item.txt. After editing the file, you first remove the old fields

```
rtt_edit> clear items
```

and reload the modified fields

```
rtt_edit> inlcude items items.txt
```

Copy pictures between programs

To copy a picture from one program to another, enter the picture and write it to file with the command

```
rtt_edit> write picture pict.tmp
```

The file can be read into an empty picture editor with the command

```
rtt_edit> include picture pict.tmp
```

Search for attributes

Sometimes you want to know where a certain database attributes is affected.

The command

```
rtt_edit> write items/all items.tmp
```

will write all fields in all pictures to the file items.tmp. By searching in the file it is possible to find all the fields that handles the attribute.

Helptexts

It is possible to write helptexts for pictures and submenus by editing an c include file.

The file is opened with the vi editor wit the command

```
rtt_edit> edit help
```

A helptext entry is divided in subject, info and text. RTT_HELP_SUBJ is the name of the submenu or picture, RTT_HELP_INFO a line that is written at the bottom of a menu page, and RTT_HELP_TEXT a text that is displayed when Ctrl/H is pressed when the submenu or picture is open.

A helptext entry for the submenu “Maintenance” can look like this.

```
RTT_HELP_SUBJ(“MAINTENANCE”)
RTT_HELP_INFO(“\
 Select a subject and press Return”)
RTT_HELP_TEXT(“\
        Various functions for maintenance of the system.\n\n\
Valves        Display valves in the plant.\n\
Motors        Display motors in the plant.\n\
Temperatures  Display tempertures sensors in the furnish.\n\
“)
```

The info-text is not used for pictures.

The file is a c include-file and the syntax follows the string handling in c. \n is linefeed and \ at the end of a row means that the string continues on the next line. The helpfile is common for all menues and pictures in the program.

Save

The command 'save' saves menues and pictures. If you enter save in the menu editor the menues are saved, if you enter save in the picture editor, the current picture is saved.

```
rtt_edit> save
```

Create a rtt-program

When the editing of menues and pictures is finished, you create the executable with the command

```
rtt_edit> link
```

Quit

The edit session is terminated by the command 'quit'

```
rtt_edit> quit
```

Run the program

The program is started from the shell by typing the name of the executable which is rt_rtt_'programname'. If the programname is vhxn2r the start command will be

```
> rt_rtt_vhxn2r
```

Menues

Menues in rtt is used to navigate to different pictures, to enter the object tree etc. A menu tree is built in the rtt editor with the command 'create'. When the menu tree is created it is used to navigate and open pictures and submenues in the same way as in the final program.

Menu mode

In menu mode you create menues and navigate in the created menues. The following keys are defined in a menu picture.

Key	**Function**
Arrow keys	Navigate and select a menu item.
Return	Open a submenu or picture.
Ctrl/R or F4	Go back to previous menu.
Page Down or Ctrl/F	Go to next page in the menu.
Page Up or Ctrl/D	Go to previous page in the menu.
Ctrl/Z	Back to main menu.
Ctrl/W	Redraw the picture.
Help	Help.
Backspace	Remove a menu item.
Ctrl/B	Open command prompt.

Create menues and menu items

New menu items are created with the 'create' command.
There are a number of different menu items.

menu	Item for a submenu.
picture	Item for a picture.
objecthierarchy	Item for the database.
exit	Item to exit.
command	Item to execute a rtt command.
vms	Item to exectute a shell command.
permpicture	Item for a picture that stores the subscriptions.
keys	Item for a menu where functions to fast keys are created.
syspicture	Item for a system picture.
objpicture	Item for an object picture.

A menu is created by selecting the menu item above the position you want to insert the new item, and typing the create command with specified type.

`rtt_edit> create/menu 'title'`	Create item for a submenu
`rtt_edit> create /picture 'title'`	Create item for the object tree
`rtt_edit> create /exit 'title'`	Create item to exit
`rtt_edit> create /command=”command” 'title'`	Create item to execute a rtt command
`rtt_edit> create /cmdhold=”command” 'title'`	Craete item to execute a rtt command from a function key without leaving the current picture.
`rtt_edit> create /vms=”command” 'title'`	Create item to execute a shell command.

`rtt_edit> create permpicture 'title'`	Create item for a permanent picture.
`rtt_edit> create keys 'title'`	Create item for fast key functions.
`rtt_edit> create /syspict='syspicture' 'title'`	Create item for a system picture.
`rtt_edit> create /objpict='objpicture' /name='objectname' 'title'`	Create item for an object picture.

'title' is the text that is written in the menu item. If you want to have spaces, the title should be enclosed by qoutation marks. To open a menu item there has to be at least one item in the submenu. This is created by selecting the parent item an add /child to the create command. When this is done it is possible to open the menu item and continue to create items in the submenu.

```
rtt_edit> create /menu/child 'title'
```

A menu title is modified by the 'modify' command

```
rtt_edit> modify ”text”
```

The command in a command menu item is viewed with the command 'show command'

Delete menu items

A menu item is removed with the Backspace key. If the item contains a submenu, or a menu tree, this is removed as well.

The last remove operation is recreated with the command 'undo delete'.

Move menu items

You can move a menu item with the 'delete' / 'undo' commands. The last delete operation is stored in a buffer, and with the 'undo delete' command the content of the buffer is inserted below the selected menu item.

Picture editor

An picture item is created with the command

```
rtt_edit> create /picture 'title'
```

By selecting an item and pressing Return you enter the picture editor.

Edit mode

The following keys are defined in edit mode.

Key	Function
Arrow keys	Navigate and select a menu item.
Ctrl/A	Open an update field.
Ctrl/T	Select.
Ctrl/E or F3	Copy selected area to paste buffer.
Ctrl/R or F4	Go back to the menu.
Ctrl/Z	Go back to root menu.
Ctrl/W	Redraw the picture.
Ctrl/N	Show collection picture.
Ctrl/V	Connect an update field to selected object in the collection picture.
Ctrl/F	Paste
Help	Help.
Ctrl/B	Open command prompt.

In edit mode, a work area with the size 22 x 80 characters is created in which characters and update fields can be inserted. The two bottom rows are reserved for commands and messages.

The character set is changed with the 'set' command:

```
rtt_edit> set ascii
rtt_edit> set line
rtt_edit> set inverse
rtt_edit> set noinverse
```

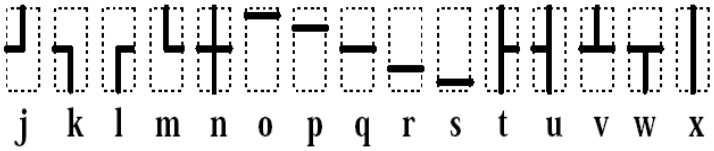

Fig Line character set

Update fields are created with the 'create' command

rtt_edit> create 'text'

'text' is a text that is inverted when the update field is selected in the final picture. If you should not be able to select the field, the text shoud be set to '%'.

Edit an update field

By placing the cursor on the first character of an update field and pressing Ctrl/A, the properties of the field are displayed. Use the arrow keys to select a property and set a value with the 'modify' command

```
rtt_edit> modify 'value'
```

Update field data

Number

Number states the order in which the update fields will be selected in the picture when the fields are selected with the ArrowUp and ArrowDown keys. ArrowLeft and ArrowRight selects the closest field to the left or to the right on the same row independent of 'Number'. The fields should be numbered by columns with the lowest number in the upper left corner, and the highest number in the lower right corner. See figure below.

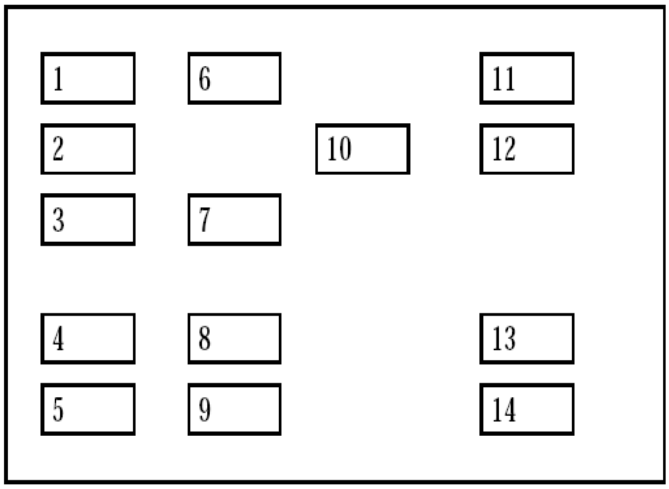

Fig Update field numbers

Text

The text property is a text in front of the field, that is drawn inverse when the field is selected. If the text is set to '%' no text is drawn and it is not possible to select the field.

Type

Type of function to change value of the connected database attributes.

Type	***Description***
UPDATE	The value of the attributes is changed with the change value function, Ctrl/E or F4.
SET	A boolean attribute is set by pressing Ctrl/A.
RESET	A boolean attribute is reset by pressing Ctrl/A.
SET_RESET	A boolean attribute is set by Ctrl/A and reset by Ctrl/T.
TOGGLE	A boolean attribute is toggled by Ctrl/A.
DUAL_SET	One attribute is displayed in the picture, and another, the dualparameter, is affected by the set function, i.e. the dual parameter is set by Ctrl/A.
DUAL_RESET	One attribute is displayed, and another is reset by Ctrl/A.
DUAL_SET_RESET	One attribute is displayed and another is set by Ctrl/A and reset by Ctrl/T.
DUAL_TOGGLE	One attribute is displayed en another is toggled by Ctrl/A.
SET_DUALSET	One attribute is set by Ctrl/A and possibly displayed, and another is set by Ctrl/T.
TOGGLE_DUALTOGGLE	One attribute is toggled by Ctrl/A and possibly displayed, and another is toggled by Ctrl/T.
COMMAND	A rtt command stated in the 'Text/Dualpar' property is executed by pressing Ctrl/A.

Parameter

Name of the attribute which value is to be displayed in the field.

Text/Dualpar

If 'Outputflags' is TEXT, the text is set in 'Text/Dualpar'. The texts that is to be displayed for true or false value of the connected attribute, are separated by a '/'.
If the text should be draw inverted, an exclamation mark is inserted as the first character.
If the text should be drawn as line graphics the prefix _L_ is added to the text. The text is stated with the ascii characters that correponds to the line graphic signs.
In the 'Characters' property the length, or if two text, the length of the longest text, is stated.

Example

”The motor is started/!The motor is stopped”

When the value of the connected attribute is true the text ”The motor is started” is displayed in field, when the value is false the text ”The motor is stopped” is displayed with inverted drawing.

If the field type is COMMAND, the rtt command is stated in 'Text/Dualpar'.

If the dual function is used, the name of the attribute that is to be affected by the dual function is inserted in 'Text/Dualpar'.

Privileges

The privileges required to change the attribute value.

Privilege	***Description***
OP	All users with runtime privilege are authorized.
EL	Users with Maintenance privilege are authorized.
PROC	Users with Process or Instrument privilege are authorized.
SYS	Users with System privilege are authorized.
NO	The value can not be changed.

Outputflags

Type of presentation of a boolean or a float.

Outputflags	***Attribute type***	***Description***
ONOFF	Boolean	Displayed with the texts ON (true) and OFF (false).
AUTOMAN	Boolean	Displayed with the texts AUTO (true) and MAN (false).
OPENCLOSED	Boolean	Displayed with the texts OPEN (true) and CLOSED (false).
TRUEFALSE	Boolean	Displayed with the texts TRUE (true) and FALSE (false).
NO	Boolean	The value is not displayed, only the text in the 'Text' property.
BAR	Float	The value is displayed as a horizontal bar.
TEXT	Boolean	One text string is displayed when the value is true, and another when the value is false. The text strings are stated in the 'Text/Dual' property.
FLASHTEXT	Boolean	Flashing text.

Characters

The size of the update field. Number of characters.

Decimals

Number of decimals in a float value.

If the object is a gdh-object of type Objid you can specify that only the last name segment should be displayed by setting Descimals to 1.

If the object is of type pwr_tTime or TIME you can specify that only the time, not the date should be displayed by settting Decimals to 1.

Maxlimit

The maximum value that is accepted when the value is changed by the user.

MinLimit

The minimum value that is accepted when the value is changed by the user.

Database

Database	***Description***
GDH	The attribute reside in the realtime database.
RTT	The attribute is created locally in rtt and handled by the user in the function picture. Only valid in function pictures.
USER	The pointer to the attribute is set by the user in the function of a function picture. Only valid in function pictures.

Declaration

Declaration of a local variable. Only user when Database is RTT.

Declaration	***Descripition***
BOOLEAN	Boolean (unsigned char).
INT	Integer.
FLOAT	Float.
CHAR	Char.
STRING	String with 80 characters.
STRING4	String with 4 characters.
STRING10	String with 10 characters.
STRING20	String with 20 characters.
STRING40	String with 40 characters.

Bar

A float value can be displayed as a horizonal bar. The field should have the following settings.

Text	%
Type	UPDATE
Privileges	NO
Outputflags	BAR
Characters	The length of the bar in characters.
Maxlimit	The maxium value for the bar.
Minlimit	The minimum value for the bar
Database	GDH

Field displaying current time

With the command

```
rtt_edit> create /time
rtt_edit> create /fulltime
```

a field is created that shows the current time with or without date.

Fields displaying alarms

It is possible to create fields that displays the five lastest alarms.
The fields are created by the 'create' command and should have the settings.

Text	%
Type	UPDATE
Parameter	RTT_ALARMTEXT1 (display the lastest alarm) RTT_ALARMTEXT2 (display the second latest alarm) ... RTT_ALARMTEXT5 (display the fifth latest alarm).
Privileges	NO
Characters	State the text size in the picture.
Database	RTT
Declaration	STRING

Pictures with code behind

Pictures with user written c-code is created by the command

```
rtt_edit> create /picture/function='function name'
```

'function name' is a function that is going to be called when

- the picture is opened.
- the picture is closed.
- every scan.
- when PageUp or PageDown is pressed.
- when the value of any parameter in the picture is changed by the user.

When you save a function picture the first time, a file for the function is created where the code for the function is edited. The filename is ra_rtt_'programname'.c and is opened with the command.

```
rtt_edit> editerings
```

In this file one function for each function picture should be inserted. The function should contain code to handle opening and closing of the picture etc.

When the files is edited, it is compiled by the command

```
rtt_edit> compile
```

Local variables

It is possible to display the value of local variables in a function picture. You then create an update field in the picture, set a suitable variable name in the 'Parameter' property, set ”RTT” as 'Database' and set the c-type of the variable (BOOLEAN, CHAR, INT, FLOAT or STRING) in the 'Declaration' property. The local variables are accessible in the functions file as they are automatically decared in an include file which is included by the function file.

Function file

In the function file a function should be declared with the name that was specified when the function item was created (block letters). The function should be of type int and declared as

```
int 'functionname' (  menu_ctx ctx,
                      int      event,
                      char     *parameter_ptr)
```

ctx is an identity for the picture that normally is not used.

event type of event that causes the call of the function. The different events are

Event	***Description***
RTT_APP_INIT	Call when the picture is opened.
RTT_APPL_EXIT	Call when the picture is closed.
RTT_APPL_UPDATE	Cyclic call to updated values.
RTT_APPL_VALUECHANGED	Call when a value is changed by the operator.
RTT_APPL_NEXTPAGE	Call when the PageDown key is pressed.
RTT_APPL_PREVPAGE	Call when the PageUp key is pressed.

parameter_ptr a pointer to changed attribute value for a RTT_APPL_VALUECHANGED event. See appendix A for function file example.

Function keys

It is possible to start up to fifteen menu alternatives from fastkeys. All different types of menu items can be configured as fastkeys.

Fastkey menu

The fastkey items are created below a special menu items. This menu item has to be positioned on the top level in the menu hierarchy. The menu item is created with the commandfile

rtt_edit> create /keys 'title'

The menu items created below this item is activated when the fastkeys are pressed. The order of the menu items decides which fast key is is connected to. The first menu item is activated by the first fast key etc.

From an ordinary keyboard the fastkey-functions are activate with the key sequences ESC*A, ESC*B, ..., ESC*O.

Helptexts

To every menu and picture a helptext can be specified that is opened when Ctrl/H is pressed. Helptext are edited in the helptext file,

```
dtt_appl_'programname'_m.hlp
```

which is created the first time a program is saved.

There are a number of macro to specify the helptext.

- RTT_HELP_START Start the helptext definition
- RTT_HELP_SUBJ(”title”) Picture or menu that is connected to the helptext. 'title' is the title of the menu or picture.
- RTT_HELP_INFO(”Some info...”) Information string written on row 23 in a menu picture.
- RTT_HELP_TEXT(”Some help text...”) Help text.
- RTT_HELP_END End of definition

The helptext can be written in severel lines, every line has to be ended with the string continuation sign '\' (note that there shouldn't be any character to the right of \). A new line in the final helptext is achieved with '\n'.

Example

```
RTT_HELP_TEXT(”\
Some text with an empy line after\n\n\
more text\n\
even more text”)
```

System pictures

There are a number of system pictures that can be inlcude into a program.

The standard layout is stored in the file rtt_menu_template.dtt_m. Select the parent menu item for the system pictures and type the commandfile

```
rtt_edit> include menu rtt_menu_template
```

It is also possible to create the items by hand. Create a menu item on the top level

```
rtt_edit> create SYSTEM
```

Create the first item below SYSTEM

```
rtt_edit> create/child NETHANDLER
```

Open the SYSTEM menu an create the other items on this level

```
rtt_edit> create /syspict=RTTSYS_ERROR ”SHOW ERROR”
rtt_edit> create /syspict=RTTSYS_PLCPGM PLCPGM
rtt_edit> create /syspict=RTTSYS_GRAFCET GRAFCET
rtt_edit> create /syspict=RTTSYS_DEVICE DEVICE
rtt_edit> create /syspict=RTTSYS_PID PID
rtt_edit> create /syspict=RTTSYS_LOGGING LOGGING
```

Select NETHANDLER and create the first system picture below NETHANDLER

```
rtt_edit> create /child/syspict=RTTSYS_SHOW_NODES ”SHOW NODES”
```

Create the other items below NETHANDLER by opening the NETHANDLER menu

```
rtt_edit> create /syspict=RTTSYS_SHOW_SUBCLI ”SHOW SUBSCRIPTION CLIENT”
rtt_edit> create /syspict=RTTSYS_SHOW_SUBSRV ”SHOW SUBSCRIPTION SERVER”
```

Also create the STORE item on the top level

```
rtt_edit> create STORE/command=”show file”
```

Object pictures

For some classes there are object pictures displaying data for an instance of the class. The object pictures can be opened from menu items, where you specify which object picture is to be used, and which object i shoud be connected to. Create a menu item with the command

rtt_edit> create /objpict='classpicture' /name='objectname' 'title'

Example

```
rtt_edit> create /objpict=RTTSYS_OBJECT_PID /name=vhx-z1-tempregl-w-pid0
”TEMPERATURE CONTROLLER”
```

Object picture exist for the classes PID and Av.

Commands

The command prompt is opened by pressing Ctrl+B.

Picture editor commands

clear picture

Remove everything in the working area.

```
rtt_edit> clear picture
```

clear items

Remove all update fields.

```
rtt_edit> clear items
```

connect

Connect a attribute in the database to an update field.
Places the attribute selected in collection picture in the parameter property of the current update field (also Ctrl+V)

```
rtt_edit> connect
```

copy

Copy the selected area to the paste buffer (also F3).

```
rtt_edit> copy
```

create

Create an update field in a picture.
With /time the current time is displayed (hh:mm:ss), and with /fulltime also the date (dd-mmm-yyyy hh:mm:ss).

```
rtt_edit> create 'text'
rtt_edit> create /time 'text'
rtt_edit> create /fulltime 'text'
```

cut

Remove selected area and insert it into the paste buffer.

```
rtt_edit> cut
```

delete

Remote an update field (delete item) or the seleted area in a picture.

```
rtt_edit> delete
rtt_edit> delete item
```

dualconnect

Connect a database attribute to an update field.

Places the attribute selected in the collection picture in the dualpar propery of the curren update field.

```
rtt_edit> dualconnect
```

include items

Read a textfile with update fields created with the 'write items' command.

```
rtt_edit> include items 'filename'
```

include picture

Read a picture into the current picture.

```
rtt_edit> include picture 'filename¨
```

modify

Modify data in an update field.

When an update items is opened, the selected property is set with modify.

```
rtt_edit> modify 'value'
```

Properties for an update field can also be modified from the picture editor without opening the field.

```
rtt_edit> modify /number=
rtt_edit> modify /text=
rtt_edit> modify /type=
rtt_edit> modify /parameter=
rtt_edit> modify /dualparameter=
rtt_edit> modify /privileges=
rtt_edit> modify /outputflags=
```

paste

Copy the content of the paste buffer into the current pictire (also Ctrl+F).

```
rtt_edit> paste
```

save

Save a picture. The picture are saved for the platforms stated in Operating system' in the setup. If you want to save for a specific platform this can be added as argument.

```
rtt_edit> save
rtt_edt> save axp_vms
```

select

Select an area (also Ctrl+T).
The selected area i drawn inverted.

```
rtt_edit> select
```

set

Set character set (ascii or line) and inverted on not inverted drawing.

```
rtt_edit> set line
rtt_edit> set ascii
```

```
rtt_edit> set inverse
rtt_edit> set noinverse
```

unselect

Reset select.

```
rtt_edit> unselect
```

write items

Write the update fields in the current picture into a textfile.
The textfile can be read by the 'include items' command.

```
rtt_edit> write items 'filename'
```

write picture

Write the picture to file. The file is read with the include command.

```
rtt_edit> write picture 'filename'
```

Menu editor

create

Create a menu item.
The qualifiers /menu, /picture, /exit and /objecthierarchy create items of different types. The enry is positioned after the selected item.
With the qualifier /child a item is created in a submenu, i.e. as a child to an item of type menu.

```
rtt_edit> create /menu 'title'
rtt_edit> create /picture 'title'
rtt_edit> create /picture/function=”functionname” 'title'
rtt_edit> create /exit 'title'
rtt_edit> create /objecthiearchy 'title'
rtt_edit> create /command=”command” 'title'
rtt_edit> create /cmdhold=”command” 'title'
rtt_edit> create /menu /child 'title'
rtt_edit> create /picture/child 'title'
rtt_edit> create /exit/child 'title'
rtt_edit> create /objecthierarchy/child 'title'
rtt_edit> create /command=”command”/child 'title'
```

Example

```
rtt_edit> create/command=”debug children/name=vkv-vkvau1-rack-card3” ”Card 3”
rtt_edit> create/picture ”My picture”
```

include menu

Read a menu tree from a textfile created by the 'write menu' command. The menu is placed after the selected item.

```
rtt_edit> include menu 'filename'
```

modify

Change the title of a menu item or the title of the main menu.

'maintitle' is the title of the root menu.
'titleprefix' it a text viewed in the upper left corner on every menu page.
'titleprefix' can contain the current node or system by using rtt symbols. The rtt standard program uses the string ”RTT-'RTT_NODE'-'RTT_SYS'”. To be able to input this string, the ' signs has to be replaced by #' otherwise the symbols are converted at the input, i.e. the corresponding input command will be

```
rtt_edit> modify /titleprefix=RTT-#'RTT_NODE#'-#'RTT_SYS#'
```

If the 'text' contains quotes, the text has to be enclose by qoutes.

```
rtt_edit> modify 'text'
rtt_edit> modify /maintitle='title'
rtt_edit> modify /titleprefix=
```

show

The show command displays
- the store command for a command item.
- the stored function for a function picture, system picture or object picture.
- the stored object for an object picture.

```
rtt_edit> show command
rtt_edit> show function
rtt_edit> show syspicture
rtt_edit> show vms
rtt_edit> show objpicture
rtt_edit> show name
```

undo delete

Restore the last deleted menu item.
The menu item is restored after the selected menu item. Undo delete can also be used to move menu items.

```
rtt_edit> undo delete
```

write menu

Writes a menu tree to textfile. The file can be read by the 'include menu' command. With the /all qualifier all the whole menu tree, otherwise only the selected menu item and its descendants are written.

```
rtt_edit> write menu /all 'filename'
rtt_edit> write menu 'filename'
```

Common commands

compile

Compile the function file and insert it into an archive.

```
rtt_edit> compile
```

edit

Edit the function file. Opens the function files in the vi editor.

```
rtt_edit> edit
```

exit

Save the menues and the current picture and quit.

export gdhreflist

Lists all database references in all pictures on the specified file.

```
rtt_edit> export gdhreflist 'filename'
```

export externref

Obsolete.

link

Compile and link the rtt program. The link is made for the platforms specified in 'Operating system' in the setup. If no platform is specified, the link is made for the current platform.
You can choose to link for a specific platform by adding the platform as an argument to the link command. The platform can be vax_eln, vax_vms, axp_vms or x86_lynx.

```
rtt_edit> link
rtt_edit> link axp_vms
rtt_edit> link x86_lynx
```

quit

Terminates without save.

save

Save the menu tree and possibly the current picture.
In the menu editor the menu tree is saved, in the picture editor the current picture and the menu tree are saved.

```
rtt_edit> save
```

setup

Display the setup menu.

Property	***Description***
Program name	Program name. If the name is changed, all the pictures has to be saved.
Main title	Title in the main menu.
Title prefix	Text displayed in the upper left corner in all menues.
Operating system	Plattform for which the rtt program is to be built. The code for the platform is stated. If several platforms, the codes are added. The codes are val_eln 1, vax_vms 2, axp_vms 4, ppc_lynx 8, x86_lynx 16.
Default directory	Default directory for the edit session.
Verify	If commandfiles executed in the edit session should be exectuted with verify or not.
Build directory	Directory for build files ($pwrp_rttbld).

show collection

Show the collection picture.

```
rtt_edit> show collection
```

Appendix A

In this example of the c-code function of a function picture. The function START_FUNCTION is connect to the picture item, and has the local variables SPEED (FLOAT), START (BOOLEAN and START_TEXT (STRING) declared in update fields in the picture. The picture is create by the command

```
rtt_edit> create /picture/function=START_FUNKTION "START MOTOR"
```

```
int START_FUNKTION ( menu_ctx ctx,
                     int event,
                     char *parameter_ptr)
{
  /************************************************************
  * Update rtt database
  ************************************************************/
  if ( event == RTT_APPL_UPDATE)
  {
    return RTT__SUCCESS;
  }
  /************************************************************
  * Initialization of the picture
  ************************************************************/
  if ( event == RTT_APPL_INIT)
  {
    SPEED = 44.;
    START = 0;
    strcpy( &TEXT, "Motorn is ready to start");
  }
  /************************************************************
  * Exit of the picture
  ************************************************************/
  else if ( event == RTT_APPL_EXIT)
  {
  }
  /************************************************************
  * The value of a parameter is changed.
  ************************************************************/
  else if ( event == RTT_APPL_VALUECHANGED)
  {
    if ( parameter_ptr == &START)
    {
      if ( START == 1)
        strcpy( &TEXT, "The motor is started");
      else
        strcpy( &TEXT, "The motor is stopped");
    }
  }
  return RTT__SUCCESS;
}
```