

PART 3

Reporting Functions

Introduction to Part 3

Contents of Part 3

Part 3 is divided into the following chapters:

Statistical Variable Functions

Bar Chart Functions

History Chart Functions

Line Chart Functions

Within each chapter, the functions are listed alphabetically by function name.

Chapter 12

Statistical Variable Functions

Table of Statistical Variable Functions

Task	Function
Creating	SV'createint
Initializing and clearing	SV'init
Updating	SV'upd
Inspecting average count of number of values current value first value maximum minimum standard deviation sum sum of the squares sum of the squares of the deviation variance	SV'avrg SV'count SV'value SV'first SV'max SV'min SV'std SV'sum SV'ss SV'ssd SV'vari

Reporting Functions

SV'avrg

Returns the average value of a statistical variable.

Synopsis

```
SV'avrg : (int Statvar) -> real
```

Description

Returns the average value of a statistical variable.

Arguments

Statistical variable created by `SV'createint`.

Return Value

Average value of `Statvar`.

Exceptions

None.

Example

```
val isv_UseRes = SV'createint ();
SV'init isv_UseRes;
SV'upd (isv_UseRes, 4);
SV'upd (isv_UseRes, 2);
SV'upd (isv_UseRes, 8);
SV'avrg isv_UseRes; -> 4.6666666666666667:real
```

See Also

```
SV'createint
SV'init
SV'upd
SV'std
SV'vari
```

Corresponding real function

```
SV'avrg : (real Statvar) -> real
```

SV'count

Returns the count of the number of times `SV'upd` has been called on a statistical variable.

Synopsis

```
SV'count : (int Statvar) -> int
```

Description

Returns the count of the number of times a statistical variable is updated with `SV'upd`.

Arguments

Statistical variable created by `SV'createint`.

Return Value

Number of times `Statvar` is updated with `SV'upd`.

Exceptions

None.

Example

```
val isv_UseRes = SV'createint ();
SV'init isv_UseRes;
SV'upd (isv_UseRes, 4);
SV'upd (isv_UseRes, 2);
SV'upd (isv_UseRes, 8);
SV'count isv_UseRes; -> 3 : int
```

See Also

`SV'createint`
`SV'init`
`SV'upd`
`SV'sum`

Reporting Functions

Corresponding real function

```
SV'count : (real Statvar) -> int
```

SV'createint

Creates an integer statistical variable.

Synopsis

```
SV'createint : unit -> int Statvar
```

Description

Creates an integer statistical variable.

Arguments

None.

Return Value

Integer statistical variable

Exceptions

None.

Example

```
val isv_UseRes = SV'createint ();  
> val sv_test = Statvar{ Avrg=ref 0.0, Count=ref 0,  
  First=ref 0, Maxi=ref 0, Mini=ref 0, SSum=ref 0,  
  Sx=ref 0, Ssd=ref 0.0, Std=ref 0.0, Value=ref 0,  
  Vari=ref 0.0 } : int Statvar
```

See Also

```
SV'init  
SV'upd
```

Corresponding real function

```
SV'createreal : unit -> real Statvar
```

Reporting Functions

SV'first

Returns the first value of a statistical variable.

Synopsis

```
SV'first : (int Statvar) -> int
```

Description

Returns the first value of a statistical variable.

Arguments

Statistical variable created by `SV'createint`.

Return Value

First value of `Statvar`.

Exceptions

None.

Example

```
val isv_UseRes = SV'createint ();
SV'init isv_UseRes;
SV'upd (isv_UseRes, 4);
SV'upd (isv_UseRes, 2);
SV'upd (isv_UseRes, 8);
SV'first isv_UseRes; -> 4 : int
```

See Also

```
SV'createint
SV'init
SV'upd
SV'value
```

Corresponding real function

```
SV'first : (real Statvar) -> real
```

SV'init

Initializes a statistical variable.

Synopsis

```
SV'init : (int Statvar) -> unit
```

Description

Initializes a statistical variable. Discards all previous values.

Arguments

Statistical variable created by `SV'createint`.

Return Value

None.

Exceptions

None.

Example

```
val isv_UseRes = SV'createint ();  
SV'init isv_UseRes; -> () : unit
```

See Also

SV'createint
SV'upd
SV'value

Corresponding real function

```
SV'init : (real Statvar) -> unit
```

Reporting Functions

SV'max

Returns the max of the values of a statistical variable.

Synopsis

```
SV'max : (int Statvar) -> int
```

Description

Returns the max of the values of a statistical variable.

Arguments

Statistical variable created by `SV'createint`.

Return Value

Max of the values of `Statvar`.

Exceptions

None.

Example

```
val isv_UseRes = SV'createint ();
SV'init isv_UseRes;
SV'upd (isv_UseRes, 4);
SV'upd (isv_UseRes, 2);
SV'upd (isv_UseRes, 8);
SV'max isv_UseRes; -> 8 : int
```

See Also

```
SV'createint
SV'init
SV'upd
SV'min
```

Corresponding real function

```
SV'max : (real Statvar) -> real
```

SV'min

Returns the min of the values of a statistical variable.

Synopsis

```
SV'min : (int Statvar) -> int
```

Description

Returns the min of the values of a statistical variable.

Arguments

Statistical variable created by `SV'createint`.

Return Value

Min of the values of `Statvar`.

Exceptions

None.

Example

```
val isv_UseRes = SV'createint ();
SV'init isv_UseRes;
SV'upd (isv_UseRes, 4);
SV'upd (isv_UseRes, 2);
SV'upd (isv_UseRes, 8);
SV'min isv_UseRes; -> 2 : int
```

See Also

```
SV'createint
SV'init
SV'upd
SV'max
```

Corresponding real function

```
SV'min : (real Statvar) -> real
```

Reporting Functions

SV'ss

Returns the sum of all squares of the values of a statistical variable.

Synopsis

```
SV'ss : (int Statvar) -> int
```

Description

Returns the sum of all squares of the values of a statistical variable.

Arguments

Statistical variable created by `SV'createint`.

Return Value

Sum of all squares of the values of `Statvar`.

Exceptions

None.

Example

```
val isv_UseRes = SV'createint ();
SV'init isv_UseRes;
SV'upd (isv_UseRes, 4);
SV'upd (isv_UseRes, 2);
SV'upd (isv_UseRes, 8);
SV'ss isv_UseRes; -> 84 : int
```

See Also

```
SV'createint
SV'init
SV'upd
SV'sum
```

Corresponding real function

```
SV'ss : (real Statvar) -> real
```

SV'ssd

Returns the sum of the squares of deviation of a statistical variable.

Synopsis

```
SV'ssd : (int Statvar) -> real
```

Description

Returns the sum of the squares of deviation of a statistical variable.

Arguments

Statistical variable created by `SV'createint`.

Return Value

Sum of the squares of deviation of `Statvar`.

Exceptions

None.

Example

```
val isv_UseRes = SV'createint ();
SV'init isv_UseRes;
SV'upd (isv_UseRes, 4);
SV'upd (isv_UseRes, 2);
SV'upd (isv_UseRes, 8);
SV'ssd isv_UseRes; -> 18.666666666666666 : real
```

See Also

```
SV'createint
SV'init
SV'upd
SV'ss
SV'std
SV'vari
```

Corresponding real function

```
SV'ssd : (real Statvar) -> real
```

Reporting Functions

SV'std

Returns the standard deviation of a statistical variable.

Synopsis

```
SV'std : (int Statvar) -> real
```

Description

Returns the standard deviation of a statistical variable.

Arguments

Statistical variable created by `SV'createint`.

Return Value

Standard deviation of `Statvar`.

Exceptions

None.

Example

```
val isv_UseRes = SV'createint ();
SV'init isv_UseRes;
SV'upd (isv_UseRes, 4);
SV'upd (isv_UseRes, 2);
SV'upd (isv_UseRes, 8);
SV'std isv_UseRes; -> 3.055050463303893 : real
```

See Also

```
SV'createint
SV'init
SV'upd
SV'ss
SV'vari
```

Corresponding real function

```
SV'std : (real Statvar) -> real
```

SV'sum

Returns the sum of the values of a statistical variable.

Synopsis

```
SV'sum : (int Statvar) -> int
```

Description

Returns the sum of the values of a statistical variable.

Arguments

Statistical variable created by `SV'createint`.

Return Value

Sum of the values of `Statvar`.

Exceptions

None.

Example

```
val isv_UseRes = SV'createint ();
SV'init isv_UseRes;
SV'upd (isv_UseRes, 4);
SV'upd (isv_UseRes, 2);
SV'upd (isv_UseRes, 8);
SV'sum isv_UseRes; -> 14 : int
```

See Also

```
SV'createint
SV'init
SV'upd
SV'count
SV'ss
```

Corresponding real function

```
SV'sum : (real Statvar) -> real
```

Reporting Functions

SV'upd

Updates a statistical variable.

Synopsis

```
SV'upd : (int Statvar * int) -> unit
```

Description

Updates a statistical variable.

Arguments

Integer statistical variable and integer constant

Return Value

None.

Exceptions

None.

Example

```
val isv_UseRes = SV'createint ();  
SV'init isv_UseRes;  
SV'upd (isv_UseRes, 4); -> () : unit  
SV'value isv_UseRes; -> 4
```

See Also

```
SV'createint  
SV'init
```

Corresponding real function

```
SV'upd : (real Statvar * real) -> unit
```

SV'value

Returns the current value of a statistical variable.

Synopsis

```
SV'value : (int Statvar) -> int
```

Description

Returns the current value of a statistical variable.

Arguments

Statistical variable created by `SV'createint`.

Return Value

Current value of `Statvar`.

Exceptions

None.

Example

```
val isv_UseRes = SV'createint ();
SV'init isv_UseRes;
SV'upd (isv_UseRes, 4);
SV'upd (isv_UseRes, 2);
SV'upd (isv_UseRes, 8);
SV'value isv_UseRes; -> 8 : int
```

See Also

```
SV'createint
SV'init
SV'upd
SV'first
```

Corresponding real function

```
SV'value : (real Statvar) -> real
```

Reporting Functions

SV'vari

Returns the variance of a statistical variable.

Synopsis

```
SV'vari : (int Statvar) -> real
```

Description

Returns the variance of a statistical variable.

Arguments

Statistical variable created by `SV'createint`.

Return Value

Variance of `Statvar`.

Exceptions

None.

Example

```
val isv_UseRes = SV'createint ();
SV'init isv_UseRes;
SV'upd (isv_UseRes, 4);
SV'upd (isv_UseRes, 2);
SV'upd (isv_UseRes, 8);
SV'vari isv_UseRes; -> 9.333333333333329:real
```

See Also

```
SV'createint
SV'init
SV'upd
SV'ssd
SV'std
```

Corresponding real function

```
SV'vari : (real Statvar) -> real
```

Chapter 13

Overview of Chart Functions

Table of Chart Functions

	All Bar Charts	History Bar Charts	Snapshot Bar Charts	Line Charts
Creating	BC_create			LC_create
Updating	BC_upd_partnames BC_upd_title	HC_upd_bar HC_upd_barnames HC_upd_chart	SC_upd_bar SC_upd_barnames SC_upd_chart SC_upd_part	LC_upd_axisnames LC_upd_chart LC_upd_line LC_upd_linenames LC_upd_pline LC_upd_title
Code Segment	BC_GetCodeSeg			LC_GetCodeSeg
Clearing	BC_clear_chart BC_init_chart			LC_init_chart
Deleting	BC_delete			LC_delete

Chapter 14

Bar Chart Functions

Snapshot Chart Example

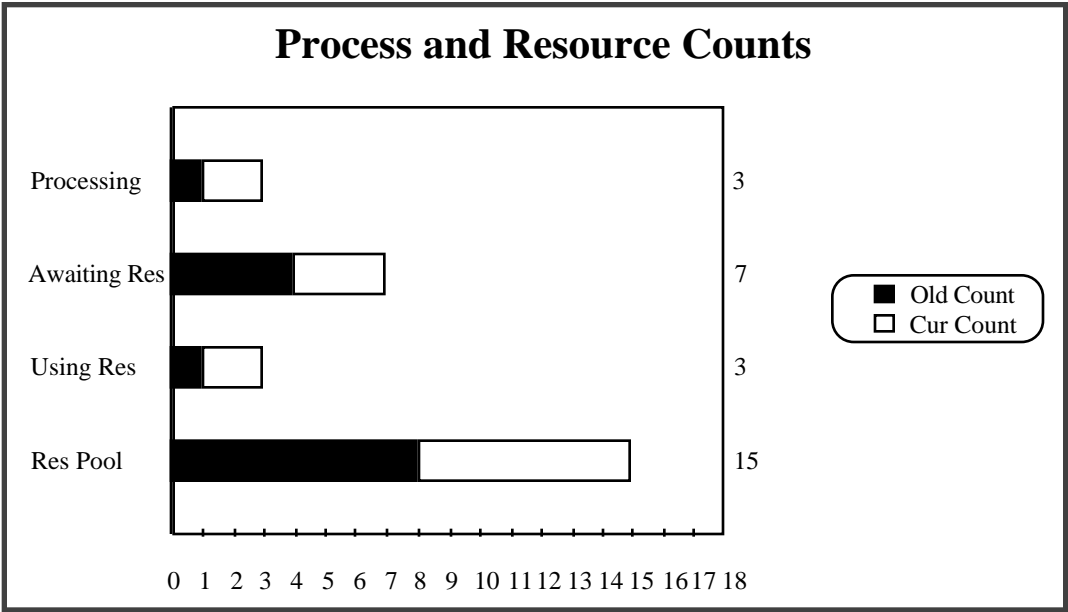


Fig. 14-1: Resource Use Model Snapshot Chart ("Res")

History Chart Example

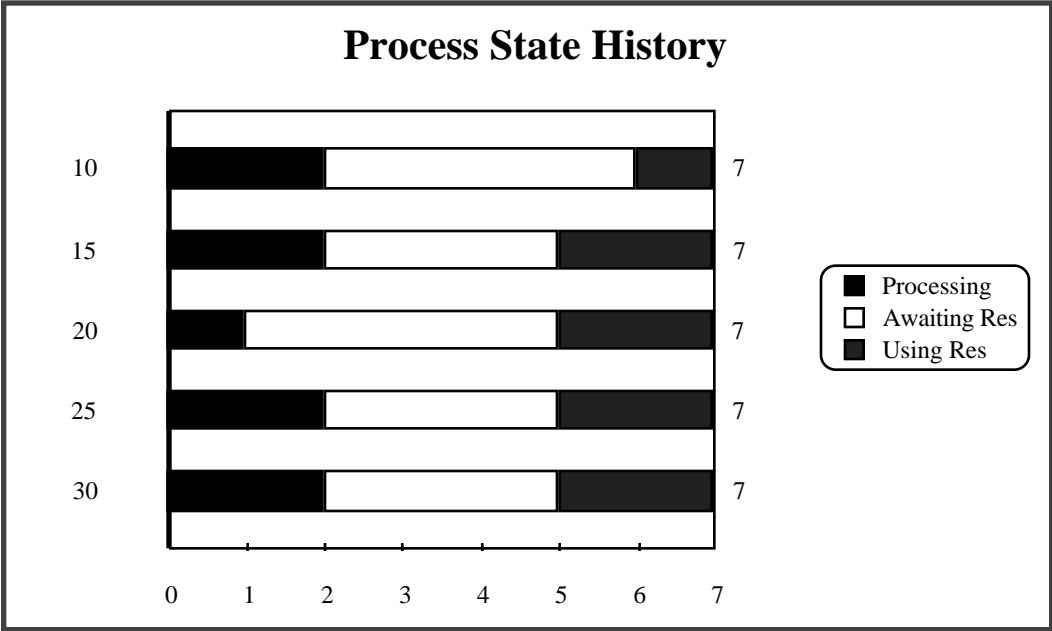


Fig. 14-2: Resource Use Model History Chart (“Hist”)

Table of Bar Chart Functions

Creating	BC_create
Updating	BC_upd_partnames BC_upd_title
History	HC_upd_bar HC_upd_barnames HC_upd_chart
Snapshot	SC_upd_bar SC_upd_barnames SC_upd_chart SC_upd_part
Code Segment	BC_GetCodeSeg
Clearing	BC_clear_chart BC_init_chart
Deleting	BC_delete

BC_create

Creates a bar chart.

Synopsis

```
BC_create : {
  Name:string,      (*Chart identifier, used by all ML
                    functions that reference this chart.
                    The chart name must be unique within
                    the diagram. *)
  Integer:bool,    (*Specifies whether values displayed
                    in a chart are integer (true) or
                    real (false). *)
  NoOfBars:int,    (*Number of bars in the chart. *)
  NoOfParts:int,   (*Number of parts into which each bar
                    is subdivided. *)
  BarHeight:int,   (*Height of the individual bars in
                    pixels. *)
  Hist:bool,       (*Causes the system to generate a
                    history chart. *)
  Retain:int,      (*Number of history chart bars to
                    retain when the bars are moved up
                    during updating. The number must be
                    smaller than or equal to the number
                    specified in NoOfBars and greater
                    than or equal to zero. *)
  GridNumber:int,  (*Causes the distance between two
                    grid lines to be fixed while the
                    numeric range between the grid lines
                    varies. *)
  GridDist:''a,    (*Causes the numeric range between two
                    grid lines to be fixed while the
                    number of grid lines varies. The
                    range number entered must be of the
                    type specified in the Integer op-
                    tion. *)
  TicksOnly:bool, (*Specifies whether the grid lines
                    are visible or not. If checked,
                    only small tics can be seen where
                    there are value regions. *)
  Title:bool,      (*Causes the system to generate a
                    title region for the chart. The
                    default title is the string "Title".
                    *)
  Legend:bool,     (*Causes the system to generate a
                    legend describing the bar part pat-
                    terns. *)
  BarNames:bool,   (*Causes the system to generate bar
                    names. *)
  PosValue:bool,   (*Causes the system to generate
                    positive value regions containing
                    default text. The font information
                    may be changed in the Editor. *)
}
```

Reporting Functions

```
NegValue:bool, (*Causes the system to generate
               negative value regions containing
               default text. The font information
               may be changed in the Editor. *)
UpperValue:bool, (*Causes the system to generate
                 value range regions corresponding to
                 the grid range above the chart. The
                 text in these regions is generated
                 from the Min and Max values and can
                 not be changed by the user. The
                 font can be changed in the Editor.
                 *)
LowerValue:bool, (*Causes the system to generate
                 value range regions corresponding to
                 the grid range below the chart. The
                 text in these regions is generated
                 from the Min and Max values and can
                 not be changed by the user. The
                 font information may be changed in
                 the Editor. *)
SaveCopy:bool, (*Causes the chart to be copied to a
               new page as an Aux node when the
               Initial State (Sim menu) command is
               invoked. *)
KeepCont:bool, (*Specifies whether or not the chart
               will be initialized when the Initial
               State (Sim menu) command is invoked.
               If true, the chart will not be
               initialized when calling Initial
               State. *)
EndOfRun:bool, (*Specifies that the chart is to be
               updated at the end of the simulation
               run. *)
Time:''b,      (*The number of time units between
               chart updates for timed models being
               simulated with time. If both Time
               and Step are included, the chart
               will be updated with both time and
               step intervals. The type is the
               kind of time specified in the
               Simulation Code Options dialog. *)
Step:int,      (*The number of steps between chart
               updates. If both Step and Time are
               included, the chart will be updated
               with both time and step intervals.
               *)
Min:''a,
Max:''a,
               (*Specify the minimum to maximum range
               of values which are initially
               displayed in the plot area. Real
               values must be entered for Min and
               Max if Integer is false. The value
               of Min must not be larger than 0;
               Max must not be smaller than 0.
               *)
Height:int,    (*Height of the chart *)
Width:int,     (*Width of the chart *)
```

Bar Chart Functions

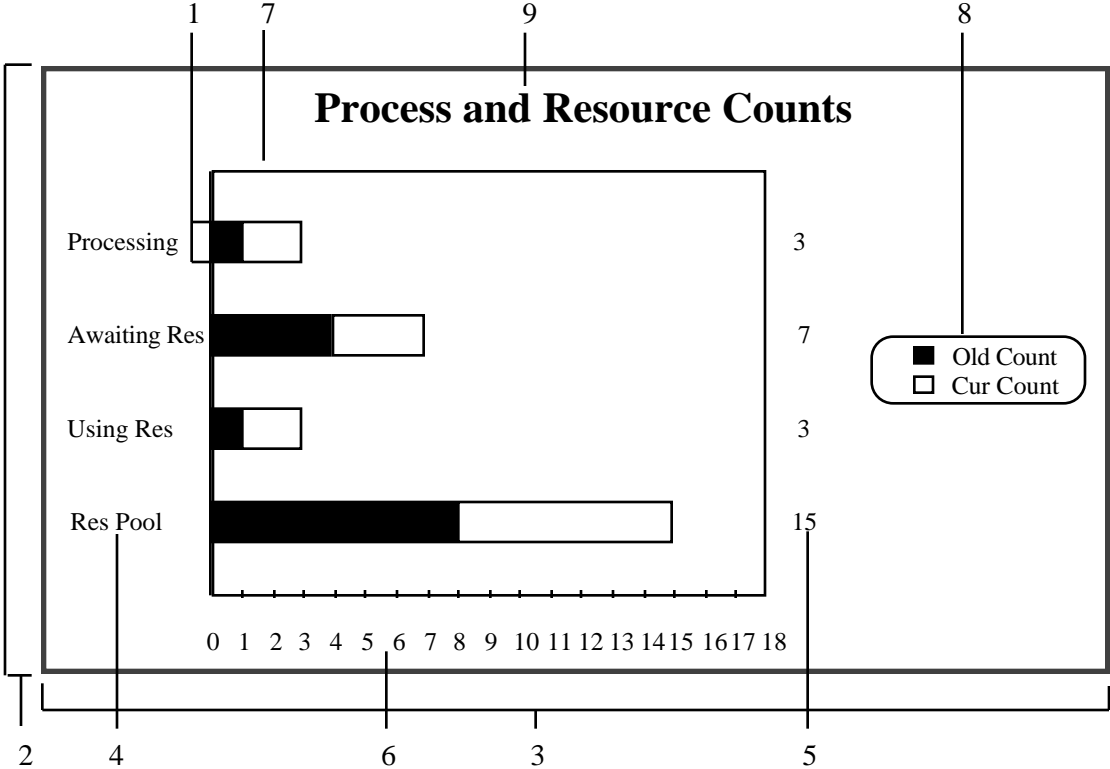
```
x:int, (*Horizontal location on the page of  
the upper right-hand corner*)  
y:int (*Vertical location on the page of the  
upper right-hand corner*)  
} -> string;
```

Description

Creates a bar chart.

Arguments

The Figure Location column in the table below refers to this diagram:



Reporting Functions

Argument	Type	Example	Figure Location	Comments
Name	string	"Res"	N/A	The chart identifier, used by all ML functions that reference this chart.
BarHeight	int	20	1	Height of the individual bars in pixels.
Height	int	200	2	Height of the chart in pixels.
Width	int	500	3	Width of the chart in pixels.
x	int	0	N/A	Horizontal location on the page of the upper right-hand corner.
y	int	700	N/A	Vertical location on the page of the upper right-hand corner.
NoOfParts	int	2	N/A	Number of parts in each bar.
NoOfBars	int	4	N/A	Number of bars in the chart.
BarNames	bool	true	4	Causes the chart to have bar labels.
PosValue	bool	true	5	Causes the chart to have positive value labels.
NegValue	bool	false	5	Causes the chart to have negative value labels.
Min	'a	0	N/A	The minimum value to be displayed in the plot area.
Max	'a	18	N/A	The maximum value to be displayed in the plot area.
GridNumber	int	N/A	N/A	Causes the distance between two grid lines to be fixed while the numeric range between the grid lines varies.
GridDist	'a	1	N/A	Causes the numeric range between two grid lines to be fixed while the number of grid lines varies.
TicksOnly	bool	true	N/A	Specifies whether or not the grid lines are visible.

Bar Chart Functions

Argument	Type	Example	Figure Location	Comments
LowerValue	bool	true	6	Specifies whether or not the chart has grid labels on the bottom of the chart.
UpperValue	bool	false	7	Specifies whether or not the chart has grid labels at the top of the chart. If the chart has a title, UpperValue is usually false to avoid text conflict.
Legend	bool	true	8	Specifies whether or not the chart has a legend identifying bar part patterns.
Title	bool	true	9	Text of the chart title. The default title can be changed by BC_upd_title.
Hist	bool	false	N/A	Causes the system to generate a history chart.
Retain	int	N/A	N/A	The number of history chart bars to retain when the bars are moved up during updating.
SaveCopy	bool	false	N/A	Causes the chart to be copied to a new page as an Aux node when the Initial State (Sim menu) command is invoked.
KeepCont	bool	true	N/A	Causes the chart to be initialized when the Initial State (Sim menu) command is invoked.
EndOfRun	bool	true	N/A	Causes the chart to be updated at the end of the simulation run.
Time	int/ real	5	N/A	The number of time units between chart updates for timed models being simulated with time. If both Time and Step are included, the chart will be updated with both time and step intervals.
Step	int	N/A	N/A	The number of steps between chart updates. If both Step and Time are included, the chart will be updated with both time and step intervals.
Integer	bool	true	N/A	Specifies whether values displayed in a chart are integer (true) or real (false).

Return Value

Name of the chart.

Reporting Functions

Exceptions

None.

Example

To create the integer snapshot bar chart "Res" shown above in Figure 14-1, in integer time mode, type:

```
BC_create
{Name="Res", BarHeight=20, Height=200,
Width=500, x=0, y=700, NoOfParts=2,
NoOfBars=4, BarNames=true, PosValues=true,
NegValues=false, Min=0, Max=18, GridDist=1,
TicksOnly=true, LowerValue=true,
UpperValue=false, Legend=true, Title=true,
Hist=false, SaveCopy=false, KeepCont=true,
EndOfRun=true, Time=5, Integer=true};
BC_upd_title (bc="Res", title="Process and Resource
Counts");
SC_upd_barnames {sc = "Res", tags = ["Processing",
"Awaiting Res", "Using Res", "Res Pool"]};
```

BC_clear_chart

Clears a bar chart.

Synopsis

```
BC_clear_chart : string -> unit
```

Description

Clears a bar chart.

Arguments

Name of a bar chart.

Return Value

None.

Exceptions

None.

Example

To clear the bar chart "Res" shown in Figure 14-1, type:

```
BC_clear_chart "Res";
```

BC_delete

Deletes a bar chart.

Synopsis

```
BC_delete : string -> unit
```

Description

Deletes a bar chart.

Arguments

Name of a bar chart.

Return Value

None.

Exceptions

None.

Example

To delete the bar chart "Res" shown in Figure 14-1, type:

```
BC_delete "Res";
```

BC_GetCodeSeg

Gets the ID of the CPN code segment region of a bar chart.

Synopsis

```
BC_GetCodeSeg : string -> gid
```

Description

Accesses a bar chart code segment region..

Arguments

Name of a bar chart.

Return Value

ID of the code segment.

Exceptions

None.

Example

To access the code segment of the bar chart "Res" shown in Figure 14-1, type:

```
BC_GetCodeSeg "Res" ;
```

BC_init_chart

Initializes a bar chart.

Synopsis

```
BC_init_chart : string -> unit
```

Description

Initializes a bar chart.

Arguments

Name of a bar chart.

Return Value

None.

Exceptions

None.

Example

To initialize the bar chart "Res" shown in Figure 14-1, type:

```
BC_init_chart "Res";
```

BC_upd_partnames

Updates the partnames in a bar chart.

Synopsis

```
BC_upd_partnames : { bc:string, tags:string list }  
-> unit
```

Description

Establishes the partnames in a bar chart. These names are used in the legend, if any.

Arguments

bc	String name of a bar chart
tags	List of string partnames.

Return Value

None.

Exceptions

None.

Example

To establish the partnames shown in the legend in Figure 14-1:

```
BC_upd_partnames {bc = "Res", tags = ["Old Count",  
"Cur Count"]};
```

Reporting Functions

BC_upd_title

Updates the title of a bar chart.

Synopsis

```
BC_upd_title : { bc:string, title:string}
              -> unit
```

Description

Establishes the title of a bar chart.

Arguments

<code>bc</code>	String name of a bar chart
<code>title</code>	String title for a bar chart.

Return Value

None.

Exceptions

None.

Example

To establish the title shown in Figure 14-1:

```
BC_upd_title {bc="Res", title= "Process and Resource
Counts"};
```

HC_upd_bar

Updates the current bar of a history bar chart.

Synopsis

```
HC_upd_bar:{bc:string, part_values:''a list}  
-> unit
```

Description

Updates the current bar of a history bar chart.

Arguments

bc	String name of a bar chart
part_values	List of values for each part of a history chart bar. The values must be of the type specified by the Integer option when the chart was created. The order is from left to right, and there must be as many elements in the list as there are parts in the bar.

Return Value

None.

Exceptions

None.

Example

To update the three-part bar of an integer history chart, named "Hist":

```
HC_upd_bar {hc="Hist", part_values=[5,8,2]};
```

Reporting Functions

HC_upd_barnames

Updates the barnames in a history bar chart.

Synopsis

```
HC_upd_barnames:{hc:string, tags:string list }  
-> unit
```

Description

Updates the barnames in a history bar chart. These names identify the meaning of each bar that appears in the chart.

Arguments

```
hc      String name of a history bar chart.  
tags    List of string barnames.
```

Return Value

None.

Exceptions

None.

Example

To update the barnames of a history chart, named "Hist" which has been created with a Retain count of 4:

```
HC_upd_barnames {hc="Hist", tags=["10", "15", "20",  
"25"]};
```

HC_upd_chart

Updates bars in a history bar chart.

Synopsis

```
HC_upd_chart:{hc:string, values:(('a list * bool)
list, tags:string list }
-> unit
```

Description

Updates the bars in a history bar chart. More than one bar may be created.

Arguments

hc	String name of a history bar chart.
values	List of value-specifier tuples, consisting of a (value-list * boolean) pair for each bar.
tags	List of string barnames.

Return Value

None.

Exceptions

None.

Example

To update a history chart, named "Hist", which has been created with a Retain count of 4 and whose bars consist of two parts:

```
HC_upd_chart {hc="Hist", values = [([3,4] * true),
([2,7] * true), ([5,6] * true), ([1, 4] * true)],
tags=["10", "15", "20", "25"]};
```

Reporting Functions

SC_upd_bar

Updates a bar of a snapshot bar chart.

Synopsis

```
SC_upd_bar:{sc:string, bar:int, part_values:''a list}  
-> unit
```

Description

Updates a bar of a history bar chart.

Arguments

sc	String name of a snapshot bar chart.
bar	Number of the bar to be updated.
part_values	List of values for each part of the snapshot chart bar. The values must be of the type specified by the Integer option when the chart was created. The order is from left to right, and there must be as many elements in the list as there are parts in the bar.

Return Value

None.

Exceptions

None.

Example

To update the three parts of the second bar of an integer snapshot chart, named "Res":

```
SC_upd_bar {sc="Res", part_values=[5,8,2]};
```

SC_upd_barnames

Updates the barnames in a snapshot bar chart.

Synopsis

```
SC_upd_barnames:{sc:string, tags:string list }  
-> unit
```

Description

Updates the barnames in a snapshot bar chart. These names identify the meaning of each bar that appears in the chart.

Arguments

```
sc      String name of a snapshot bar chart.  
tags   List of string barnames.
```

Return Value

None.

Exceptions

None.

Example

To update the barnames of a snapshot chart, named "Res" which has four bars:

```
SC_upd_barnames {sc = "Res", tags = ["Processing",  
"Awaiting Res", "Using Res", "Res Pool"]};
```

Reporting Functions

SC_upd_chart

Updates bars in a snapshot bar chart.

Synopsis

```
SC_upd_chart:{sc:string, values:(int * 'a list *
bool) list, tags:string list }
-> unit
```

Description

Updates the bars in a snapshot bar chart. Each bar is identified by an integer.

Arguments

sc	String name of a history bar chart.
values	List of value-specifier tuples, consisting an (int * value-list * boolean) tuple for each bar.
tags	List of string barnames.

Return Value

None.

Exceptions

None.

Example

To update a snapshot chart, named "Res", which has 4 bars each consisting of two parts:

```
SC_upd_chart {sc="Res", values = [(1 * [3,4] *
true), (2 * [2,7] * true), (3 * [5,6] * true), (4 *
[1, 4] * true)], tags=["10", "15", "20", "25"]};
```

SC_upd_part

Updates a part for the bars of a snapshot bar chart.

Synopsis

```
SC_upd_part:{sc:string, part:int, bar_values:''a list}  
-> unit
```

Description

Updates a part for the bars of a snapshot bar chart.

Arguments

sc	String name of a snapshot bar chart.
part	Number of the part to be updated.
bar_values	List of values for the specified part of each bar of the snapshot chart. The values must be of the type specified by the Integer option when the chart was created. There must be as many elements in the list as there are bars.

Return Value

None.

Exceptions

None.

Example

To update the second part of the four bars of an integer snapshot chart, named "Res":

```
SC_upd_part {sc="Res", part = 3,  
            bar_values=[3,6,2,9]};
```


Chapter 15

Line Chart Functions

Line Chart Example

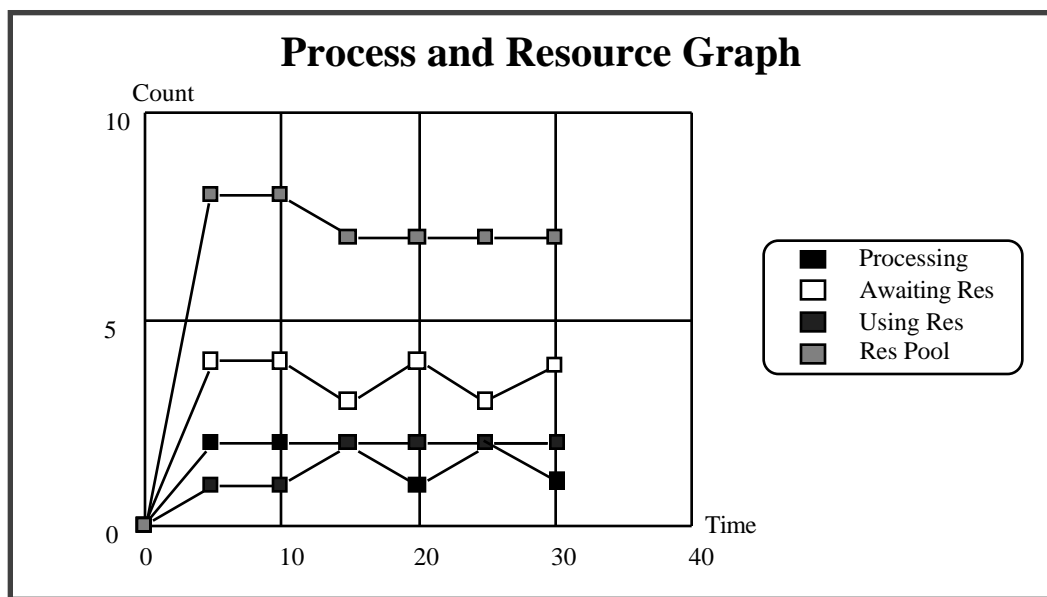


Fig. 15-1: Resource Use Model - Line Chart ("Line")

Table of Line Chart Functions

Creating	LC_create
Initializing	LC_init_chart
Updating:	
Labels	LC_upd_axisnames LC_upd_linenames LC_upd_title
Data	LC_upd_chart LC_upd_pline LC_upd_line

Reporting Functions

Accessing Code Segment	LC_GetCodeSet
Deleting	LC_delete

LC_create

Creates a line chart.

Synopsis

```

LC_create: {
  Name:string          (*Chart identifier, which is used by all the
                        ML functions that reference this chart. The
                        chart name must be unique within the
                        diagram. *)
  Integer:bool,        (*Specifies whether values displayed in a
                        chart are integer or real. *)
  NoOfLines:int,       (*Number of chart lines *)
  BoxSize:int,         (*Size of nodes connecting the line chart *)
  StartAtOrigin:bool, (*Lines start at the origin*)
  HorizVert:bool,     (*Specifies how the data points on the graph
                        lines are connected. True causes lines to
                        be drawn with right angles, such as in a
                        time series for hardware circuits. False
                        causes straight lines to be drawn between
                        points. *)
  XNoOfGrids:int,     (*Causes the distance between two grid lines
                        to be fixed while the numeric range between
                        the grid lines varies. *)
  XDist: 'a,          (*Causes the numeric range between two grid
                        lines to be fixed while the number of grid
                        lines varies. The number should be of the
                        type specified by Integer. *)
  XTicksOnly:bool,    (*X-axis ticks instead of lines*)
  YNoOfGrids:int,     (*Causes the distance between two grid lines
                        to be fixed while the numeric range between
                        the grid lines varies. *)
  YDist: 'a,          (*Causes the numeric range between two grid
                        lines to be fixed while the number of grid
                        lines varies. The number should be of the
                        type specified by Integer. *)
  YTicksOnly:bool,    (*Y-axis ticks instead of lines*)
  Title:string,       (*Text of line chart title*)
  Legend:bool,        (*Specifies whether or not the chart has a
                        legend identifying chart line patterns. *)
  AxisNames:bool,     (*Chart has axis labels. *)
  SaveCopy:bool,      (*Causes the chart to be copied to a new page
                        as an Aux node when the Initial State (Sim
                        menu) command is invoked*)
  KeepCont:bool,      (*Specifies whether or not the chart will be
                        initialized when the Initial State (Sim
                        menu) command is invoked. If true, the
                        chart will not be initialized when calling
                        Initial State. *)
  ReStartAtOrigin:bool, (*Specifies whether or not to start the new
                        lines at the origin or to continue from the
                        end of the previous run's lines*)
  EndOfRun:bool,      (*Specifies that the chart is to be updated at
                        the end of the simulation run. *)
  Time: 'b,           (*The number of time units between chart
                        updates for times models being simulated
                        with time. If both Time and Step are
                        specified, the chart will be updated with
                        both time and step intervals. The type is

```

Reporting Functions

```

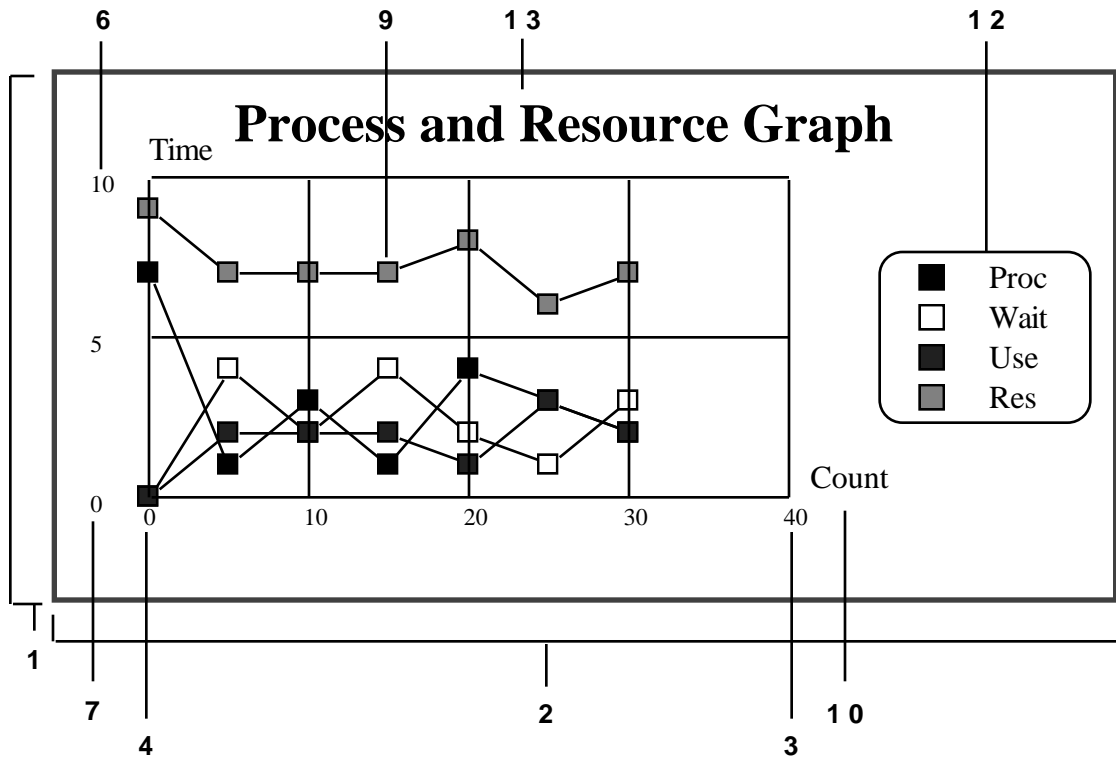
                                the kind of time specified in the Simulation
                                Code Options dialog. *)
Step:int,                        (*The number of steps between chart updates.
                                If both Step and Time are specified, the
                                chart will be updated with both time and
                                step intervals. *)
XMax:''a,                       (*Maximum value of x coordinate*)
XMin:''a,                       (*Minimum value of x coordinate*)
YMax:''a,                       (*Maximum value of y coordinate*)
YMin:''a,                       (*Minimum value of y coordinate*)
XOrigin:''a,
YOrigin:''a,                   (*Specifies where the x and y axes intersect
                                initially. x and y must be located within
                                the XMin, XMax and YMin, YMax range.
                                Specifying MoveAxis can change the location
                                of the origin. *)
RescaleAxis:bool,              (*Causes the system to automatically adjust
                                the chart's display scale to accommodate
                                changes in the data. Rescaling compresses
                                data. *)
MoveAxis:bool                  (*Causes the system to move the display scale
                                to accommodate changes in the data. The
                                display scale is fixed. If a chart will be
                                updated in both the negative and positive
                                direction, this feature should not be used.
                                MoveAxis is comparable to the history chart
                                updating method. *)
Height:int,                    (*Height of the line chart. *)
Width:int,                     (*Width of the line chart. *)
x:int,                         (*Horizontal location on the page of the upper
                                right-hand corner. *)
y:int                          (*Vertical location on the page of the upper
                                right-hand corner. *)
}
-> string
```

Description

Creates a line chart.

Arguments

The Figure Location column in the table below refers to this diagram:



Argument Type Ex. Loc. Comments

Name	string	"Line"	N/A	Chart identifier, which is used by all the ML functions that reference this chart. The chart name must be unique within the diagram.
NoOfLines	int	4	N/A	Number of chart lines.
BoxSize	int	20	9	Size of nodes connecting the line chart.
StartAtOrigin	bool	true	N/A	Lines start at the origin.
HorizVert	bool	false	N/A	Specifies how the data points on the graph lines are connected. True causes lines to be drawn with right angles, such as in a time series for hardware circuits. False causes straight lines to be drawn between points.
XNoOfGrids	int	N/A	N/A	Causes the distance between two grid lines to be fixed while the numeric range between the grid lines varies.
XDist	int/real	10	N/A	Causes the numeric range between two grid lines to be fixed while the number of grid lines varies. The number should be of the type specified by Integer.

Reporting Functions

XTicksOnly	bool	false	N/A	X-axis ticks instead of lines.
YNoOfGrids	int	false	N/A	Causes the distance between two grid lines to be fixed while the numeric range between the grid lines varies.
YDist	"a	10	N/A	Causes the numeric range between two grid lines to be fixed while the number of grid lines varies. The number should be of the type specified by Integer.
YTicksOnly	bool	false	N/A	Y-axis ticks instead of lines.
Title	bool	true	13	Specifies whether or not the chart has a title.
Legend	bool	true	12	Specifies whether or not the chart has a legend identifying chart line patterns.
AxisNames	bool	true	10	Chart has axis labels.
SaveCopy	bool	false	N/A	Causes the chart to be copied to a new page as an Aux node when the Initial State (Sim menu) command is invoked
KeepCont	bool	true	N/A	Specifies whether or not the chart will be initialized when the Initial State (Sim menu) command is invoked. If true, the chart will not be initialized when calling Initial State.
ReStartAtOrigin	bool	false	N/A	Specifies whether or not to start the new lines at the origin or to continue from the end of the previous run's lines.
EndOfRun	bool	true	N/A	Specifies that the chart is to be updated at the end of the simulation run.
Time	int/real	5	N/A	The number of time units between chart updates for times models being simulated with time. If both Time and Step are specified, the chart will be updated with both time and step intervals.
Step	int	N/A	N/A	The number of steps between chart updates. If both Step and Time are specified, the chart will be updated with both time and step intervals.
Integer	bool	true	N/A	Specifies whether values displayed in a chart are integer or real.
XMax	int/real	40	3	Maximum value of x coordinate.
XMin	int/real	0	4	Minimum value of x coordinate.
YMax	int/real	10	6	Maximum value of y coordinate.

Line Chart Functions

YMin	int/real	0	7	Minimum value of y coordinate.
XOrigin YOrigin	"a "a	0 0	N/A	Specifies where the x and y axes intersect initially. x and y must be located within the XMin, XMax and YMin, YMax range. Specifying MoveAxis can change the location of the origin.
RescaleAxis	bool	true	N/A	Causes the system to automatically adjust the chart's display scale to accommodate changes in the data. Rescaling compresses data.
MoveAxis	bool	false	N/A	Causes the system to move the display scale to accommodate changes in the data. The display scale is fixed. If a chart will be updated in both the negative and positive direction, this feature should not be used. MoveAxis is comparable to the history chart updating method.
Height	int	200	1	Height of the line chart in pixels.
Width	int	500	2	Width of the line chart in pixels.
x	int	0	N/A	Horizontal location on the page of the upper right-hand corner.
y	int	700	N/A	Vertical location on the page of the upper right-hand corner.

Return Value

Name of chart.

Exceptions

None.

Example

To create the integer line chart "Line" shown in Figure 15-1, in integer time mode, type:

```
LC_create
{Name="Line", NoOfLines=4, BoxSize=20,
StartAtOrigin=true, XDist=10, YDist=10, Title=true,
Legend=true, AxisNames=true, KeepCont=true,
EndOfRun=true, Time=5, Integer=true, XMax=40,
XMin=0, YMax=10, YMin=0, XOrigin=0, YOrigin=0,
ReScaleAxis=true, Height=200, Width=500, x=0.
y=700};
```

LC_delete

Deletes a line chart.

Synopsis

```
LC_delete : string -> unit
```

Description

Deletes a line chart.

Arguments

Reference to a line chart.

Return Value

None.

Exceptions

None.

Example

To delete the line chart "Line" shown in Figure 15-1, type:

```
LC_delete "Line";
```

LC_GetCodeSeg

Gets the ID of the CPN code segment region of a line chart.

Synopsis

```
LC_GetCodeSeg : string -> gid
```

Description

Accesses a line chart code segment region..

Arguments

Name of a line chart.

Return Value

ID of the code segment.

Exceptions

None.

Example

To access the code segment of the line chart "Line" shown in Figure 15-1, type:

```
LC_GetCodeSeg "Line";
```

LC_init_chart

Initializes a line chart.

Synopsis

```
LC_init_chart : string -> unit
```

Description

Initializes a line chart.

Arguments

Name of a line chart.

Return Value

None.

Exceptions

None.

Example

To initialize the line chart "Line" shown in Figure 15-1, type:

```
LC_init_chart "Res";
```

LC_upd_axisnames

Updates the axis names in a line chart.

Synopsis

```
LC_upd_axisnames : { lc:string, xtag:string,  
                    ytag:string};  
                  -> unit
```

Description

Establishes the axis names in a line chart. These names are used in the legend, if any.

Arguments

```
lc      String name of a line chart.  
xtag   String x-axis name.  
ytag   String y-axis name.
```

Return Value

None.

Exceptions

None.

Example

To establish the axis names shown in the legend in Figure 15-1:

```
LC_upd_axisnames {lc="Line", xtag="Time",  
                 ytag="Count"};
```

Reporting Functions

LC_upd_chart

Updates lines in a line chart.

Synopsis

```
LC_upd_chart: {lc:string, values:(int * 'a * 'a *  
bool * bool) list}  
-> unit
```

Description

Updates the lines in a line chart.

Arguments

lc	String name of a line chart.
values	List of value-specifier tuples, consisting an (integer-line-number * x value * y value * true * true) tuple for each line.

Return Value

None.

Exceptions

None.

Example

To update a line chart, named "Line", which consists of four lines with x,y values: (2,2), (2,3), (2,4), (2,8):

```
LC_upd_chart {lc="Line", values = [(1, 2, 2, true,  
true), (2, 2, 3, true, true), (3, 2, 4, true,  
true), (4, 2, 8, true, true)]};
```

LC_upd_line

Updates a line of a line chart.

Synopsis

```
LC_upd_line:{lc:string, line:int, x:''a, y:''a,  
drawline:bool}  
-> unit
```

Description

Updates a line of a line chart.

Arguments

lc	String name of a line chart.
line	Number of the line to be updated.
x	x value for the line.
y	The value must be of the type specified by the Integer option when the chart was created. y value for the line.
drawline	The value must be of the type specified by the Integer option when the chart was created. Specifies that the line is to be drawn.

Return Value

None.

Exceptions

None.

Example

To update the second line of a line chart, named "Line" with x,y value (5,9):

```
LC_upd_line {lc="Line", x=5, y=9,  
drawline=true};
```

Reporting Functions

LC_upd_linenames

Updates the line names in a line chart.

Synopsis

```
LC_upd_linenames : { lc:string, tags:string list }  
-> unit
```

Description

Establishes the line names in a line chart. These names are used in the legend, if any.

Arguments

`lc` String name of a line chart.
`tags` List of string line names.

Return Value

None.

Exceptions

None.

Example

To establish the line names shown in the legend in Figure 15-1:

```
LC_upd_linenames {lc = "Line ", tags =  
["Processing", "Awaiting Res", "Using Res", "Res  
Pool"]};
```

LC_upd_pline

Updates the fill pattern for a line terminator in a line chart.

Synopsis

```
LC_upd_pline: {lc:string, line:int, x:''a, y:''a,  
drawline:bool, pattern:int}  
-> unit
```

Description

Updates the fill pattern for a line terminator in a line chart.

Arguments

lc	String name of a line chart.
line	Number of the line to be updated.
x	x value for the line.
y	The value must be of the type specified by the Integer option when the chart was created. y value for the line.
drawline	The value must be of the type specified by the Integer option when the chart was created.
pattern	Specifies that the line is to be drawn. The number of the fill pattern.

Return Value

None.

Exceptions

None.

Example

To set the fill pattern to 3 for the second line of a line chart, named "Line" with x,y value (5,9):

```
LC_upd_pline {lc="Line", line=2, x=5, y=9,  
drawline=true, pattern=3};
```

Reporting Functions

LC_upd_title

Updates the title of a line chart.

Synopsis

```
LC_upd_title : { bc:string, title:string}
              -> unit
```

Description

Establishes the title of a line chart.

Arguments

lc	String name of a line chart
title	String title for a line chart.

Return Value

None.

Exceptions

None.

Example

To establish the title shown in Figure 15-1:

```
LC_upd_title {lc="Line", title= "Process and Resource
Graph"};
```

APPENDIX A

Version 1.9

Chart Functions

Introduction to Appendix A

Contents of Appendix A

Appendix A is divided into the following chapters:

Overview of Version 1.9 Chart Functions

Version 1.9 Bar Chart Functions

Version 1.9 History Chart Functions

Version 1.9 Line Chart Functions

Version 1.9 Matrix Chart Functions

Within each chapter, the functions are listed alphabetically by function name.

Chapter A1

Overview of Version 1.9 Chart Functions

Obsolescence of Version 1.9 Chart Functions

In Design/CPN Version 2.0, the charting capabilities provided in Version 1.9 has been replaced by a more sophisticated charting facility. For compatibility, the Version 1.9 charting functions are supported in Version 2.0.

This appendix describes the Version 1.9 charting functions. None of these functions should be used in new models. The functions are documented solely for use in models created with Version 1.9 that already use charts.

Version 1.9 and Version 2.0 charts and chart functions can be used together in the same model. However, 2.0 chart functions cannot be used with 1.9 charts, or vice versa.

Table of Version 1.9 Chart Functions

	Bar Charts	History Charts	Line Charts	Matrix Charts
Creating	BC'decint BC'create	HC'decint HC'create	LG'decint LG'create	MC'dec MC'create
Updating:				
Assigning labels	BC'upd_ltag BC'upd_rtag	HC'hist_ltag HC'hist_rtag	LG'upd_ltag LG'upd_atag	MC'upd_ltag
Appending values	BC'upd_col BC'upd_row	HC'hist_init HC'hist_row	LG'upd_line LG'upd_pline	MC'fill MC'write
Deleting	BC'delete	HC'delete	LG'delete	MC'delete

Creating a Chart Page

A newly created chart appears by default on the current page. To cause a chart to be created on a page of its own:

1. Create a page, using `DSStr_NewPage` (described in Chapter 2).
2. Make the page a CPN page, using `MakeCpnPage` (described in Chapter 8).
3. Before creating the chart, make the page current with `DSStr_SetCurPage` (described in Chapter 2).

The chart will appear on the newly created page.

Positioning a Chart on a Page

The X and Y parameters of the individual chart creation functions specify positioning of the chart on the page. To determine the X and Y coordinates that will position a newly created chart correctly:

1. Create a chart.
2. Move the chart to the desired location on the page. The Reduce menu item of the Page menu is useful in seeing and moving the chart within the page as a whole.
3. Use the Current Object menu item of the Examine menu obtain the X and Y coordinates (`xcenter` and `ycenter`) of the chart.
4. Modify the X and Y parameters of the chart creation function to specify the values obtained in Step 3.

Chapter A2

Version 1.9

Bar Chart Functions

Version 1.9 Bar Chart Example

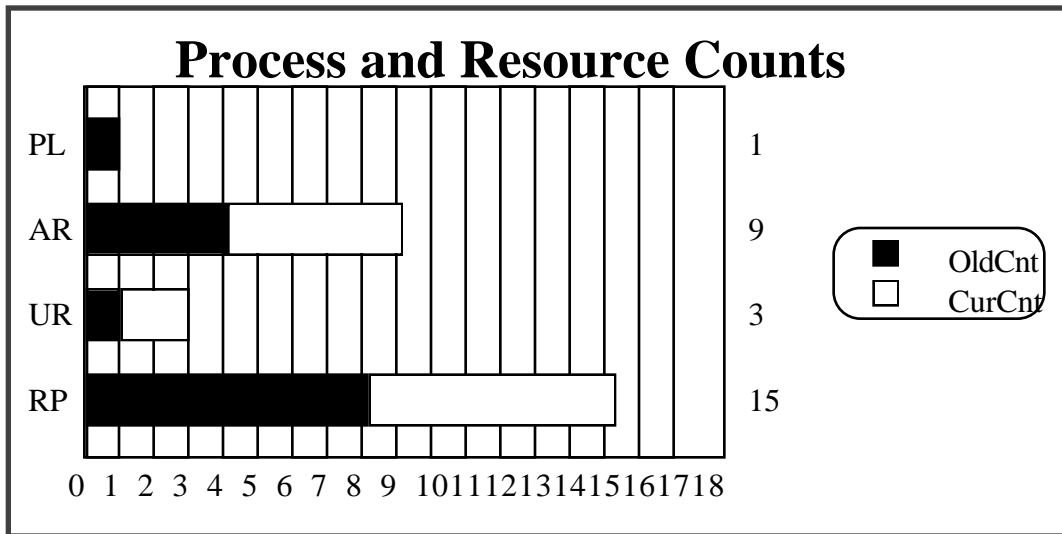


Fig. A2-1: Resource Use Model Bar Chart (bc_resmod)

Table of Version 1.9 Bar Chart Functions

Creating	BC'decint BC'create
Updating:	
Assigning labels	BC'upd_ltag BC'upd_rtag
Appending values	BC'upd_col BC'upd_row
Deleting	BC'delete

BC'create

Creates a bar chart.

Synopsis

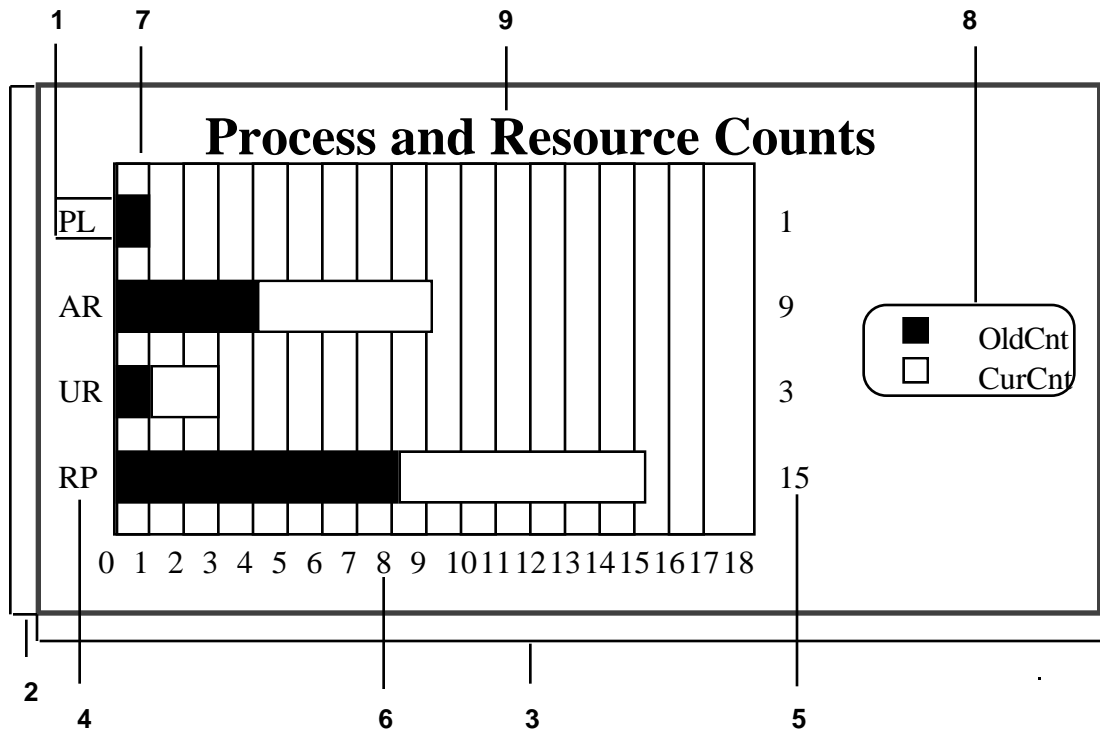
```
BC'create:
{ bar_height : int,    (* Height of the individual bars (rows) *)
  height : int,      (* Height of the chart *)
  width : int,       (* Width of the chart *)
  x : int,           (* Horizontal location on the page of the
                    upper right-hand corner *)
  y : int,           (* Vertical location on the page of the upper
                    right-hand corner *)
  columns : int,     (* Number of columns in each bar (row) *)
  row : int,         (* Number of bars (rows) in the chart *)
  tag : bool,        (* Specifies whether or not the chart has row
                    labels *)
  val_reg : bool     (* Specifies whether or not the chart has
                    value labels *)
  max_val : int,     (* Determines the range of values that are
                    displayed in the plot area *)
  resize : bool,    (* Specifies whether or not the maximum value
                    shown on the horizontal grid is adjusted to
                    be larger than the maximum bar length *)
  grid_dyn : bool,  (* Specifies whether or not the number of grid
                    lines are to be adjusted dynamically *)
  grid_no : int,    (* Number of grid lines in the chart *)
  grid_vis : bool,  (* Specifies whether or not the grid lines are
                    visible *)
  low_grid : bool,  (* Specifies whether or not the chart has grid
                    labels on the bottom of the chart *)
  up_grid : bool,   (* Specifies whether or not the chart has grid
                    labels at the top of the chart *)
  legend : bool,    (* Specifies whether or not the chart has a
                    legend identifying column patterns *)
  title : string,   (* Text of chart title *)
}
-> int BCHART
```

Description

Creates a bar chart.

Arguments

The Figure Location column in the table below refers to this diagram:



Argument **Type** **Example** **Figure Location** **Comments**

bar_height	int	20	1	Height of the individual bars (rows). The value is in pixels.
height	int	200	2	Height of the chart. The value is in pixels.
width	int	500	3	Width of the chart. The value is in pixels.
x	int	0	N/A	Horizontal location on the page of the upper right-hand corner. See Chapter A1 for a simple chart positioning technique that uses <i>x</i> and <i>y</i> .
y	int	700	N/A	Vertical location on the page of the upper right-hand corner. See Chapter A1 for a simple chart positioning technique that uses <i>x</i> and <i>y</i> .
columns	int	2	N/A	Number of columns in each bar (row). The Resource model in this appendix uses two columns in its bar chart.
row	int	4	N/A	Number of bars (rows) in the chart.
tag	bool	true	4	Specifies whether or not the chart has row labels. BC'upd_rtag determines the labels for the rows.

Version 1.9 Chart Functions

val_reg	bool	true	5	Specifies whether or not the chart has value labels. The system automatically creates the labels from the position of the left-hand end of the bar.
max_val	int	30	N/A	Determines the range of values that are displayed in the plot area. Zero is the minimum, max_val is the maximum.
resize	bool	true	N/A	Specifies whether or not the maximum value shown on the horizontal grid is adjusted to be larger than the maximum bar length. Setting resize to true helps to identify errors caused by giving a number outside of the max_val range.
grid_dyn	bool	true	N/A	Specifies whether or not the number of grid lines are to be adjusted dynamically.
grid_no	int	30	N/A	Number of grid lines in the chart. If grid_dyn is false, this number is fixed. If grid_dyn is true, this number is the maximum.
grid_vis	bool	true	N/A	Specifies whether or not the grid lines are visible.
low_grid	bool	true	6	Specifies whether or not the chart has grid labels on the bottom of the chart.
up_grid	bool	false	7	Specifies whether or not the chart has grid labels at the top of the chart. If the chart has a title, up_grid is usually false to avoid text conflict.
legend	bool	true	8	Specifies whether or not the chart has a legend identifying column patterns. BC'upd_ltag determines the labels for the patterns.
title	string	See figure	9	Text of the chart title.

Return Value

None.

Exceptions

None.

Example

To create the integer bar chart `bc_resmod` shown in Figure A2-1, type:

```
val bc_resmod = BC'decint ();
bc_resmod := BC'create
{bar_height = 20, height = 200, width =
400, x = 0, y = 350, columns = 2, row = 4,
tag = true, val_reg = true, max_val =
(MaxCount * 2), resize = true, grid_dyn =
true, grid_no = (MaxCount * 2), grid_vis =
true, low_grid = true, up_grid = false,
legend = true, title = "Process and
Resource Counts"};
BC'upd_rtag {bc = bc_resmod, tags = ["PL", "AR",
"UR", "RP"]};
```

See Also

```
BC'decint
BC'upd_rtag
BC'upd_ltag
BC'delete
```

Corresponding real function

```
BC'create: { bar_height : int, height : int, width :
int, x : int, y : int, columns : int, tag : bool,
grid_dyn : bool, grid_no : int, grid_vis : bool,
legend : bool, low_grid : bool, max_val : real,
resize : bool, row : int, title : string, up_grid :
bool, val_reg : bool } -> real BCHART
```

BC'decint

Declares an integer bar chart.

Synopsis

```
BC'decint: () -> (int BCHART) ref
```

Description

Declares an integer bar chart. This function has the effect of creating an ML value that can be used together with `BC'create` to create an integer bar chart.

Arguments

None.

Return Value

Returns reference to an integer bar chart.

Exceptions

None.

Example

To declare the integer bar chart `bc_resmod` shown in Figure A2-1, type:

```
val bc_resmod = BC'decint ();
```

See Also

`BC'create`

Corresponding real function

```
BC'dcreal = fn : () -> (real BCHART) ref
```

BC'delete

Deletes an integer bar chart.

Synopsis

```
BC'delete : ((int BCHART) ref) -> unit
```

Description

Deletes an integer bar chart.

Arguments

Reference to an integer bar chart.

Return Value

None.

Exceptions

None.

Example

To delete the integer bar chart `bc_resmod` shown in Figure A2-1, type:

```
BC'delete bc_resmod;
```

See Also

```
BC'create
```

Corresponding real function

```
BC'delete : ((real BCHART) ref) -> unit
```

BC'upd_col

Updates columns in an integer bar chart.

Synopsis

```
BC'upd_col : { bc : (int BCHART) ref, column : int,  
row_values : int list } -> unit
```

Description

Updates columns in an integer bar chart.

Arguments

bc	Reference to an integer bar chart
column	Integer column number to be updated
row_values	List of integer row values.

Return Value

None.

Exceptions

None.

Example

The Resource model used in this appendix does not update individual columns, but the function call to update the first column of all the rows would be:

```
BC'upd_col {bc = bc_resmod, column = 1, row_values  
= [(!old_ProcLoc), (!old_AwaitRes), (!old_UseRes),  
(!old_ResPool)]};
```

See Also

```
BC'upd_ltag  
BC'upd_row  
BC'upd_rtag
```

Corresponding real function

```
BC'upd_col : { bc : (real BCHART) ref, column : int,  
row_values : real list } -> unit
```

BC'upd_ltag

Updates legend labels on an integer bar chart.

Synopsis

```
BC'upd_ltag : { bc : (int BCHART) ref, tags : string  
list } -> unit
```

Description

Updates legend labels on an integer bar chart.

Arguments

bc	Reference to an integer bar chart.
tags	List of strings to use as legend labels

Return Value

None.

Exceptions

None.

Example

To create legend labels for the bar chart `bc_resmod` shown in Figure A2-1, type:

```
BC'upd_ltag {bc = bc_resmod, tags = ["OldCnt",  
"CurCnt"]};
```

See Also

```
BC'upd_col  
BC'upd_row  
BC'upd_rtag
```

Corresponding real function

```
BC'upd_ltag : { bc : (real BCHART) ref, tags : string  
list } -> unit
```

BC'upd_row

Updates rows in an integer bar chart.

Synopsis

```
BC'upd_row : { bc : (int BCHART) ref, row : int,  
column_values : int list } -> unit
```

Description

Updates rows in an integer bar chart.

Arguments

bc	Reference to an integer bar chart.
row	Integer row number to be updated
column_values	List of integer column values.

Return Value

None.

Exceptions

None.

Example

To update the second row of the integer bar chart `bc_resmod` shown in Figure A2-1, type:

```
old_AwaitRes := SV'value isv_AwaitRes;  
SV'upd (isv_AwaitRes, (!old_AwaitRes) + 1);  
BC'upd_row {bc = bc_resmod,row=2,  
column_values = [(!old_AwaitRes), SV'value  
isv_AwaitRes]};
```

See Also

BC'upd_col
BC'upd_ltag
BC'upd_rtag

Version 1.9 Chart Functions

Corresponding real function

```
BC'upd_row : { bc : (real BCHART) ref, row : int,  
column_values : real list } -> unit
```

BC'upd_rtag

Updates row labels on an integer bar chart.

Synopsis

```
BC'upd_rtag : { bc : (int BCHART) ref, tags : string  
list } -> unit
```

Description

Updates row labels on an integer bar chart.

Arguments

bc	Reference to an integer bar chart.
tags	List of strings to use as row labels

Return Value

None.

Exceptions

None.

Example

To create row labels for the integer bar chart `bc_resmod` shown in Figure A2-1, type:

```
BC'upd_rtag {bc = bc_resmod, tags = ["PL", "AR",  
"UR", "RP"]};
```

See Also

```
BC'upd_col  
BC'upd_ltag  
BC'upd_row
```

Corresponding real function

```
BC'upd_rtag : { bc : (real BCHART) ref, tags :  
string list } -> unit
```


Chapter A3

Version 1.9

History Chart Functions

Version 1.9 History Chart Example

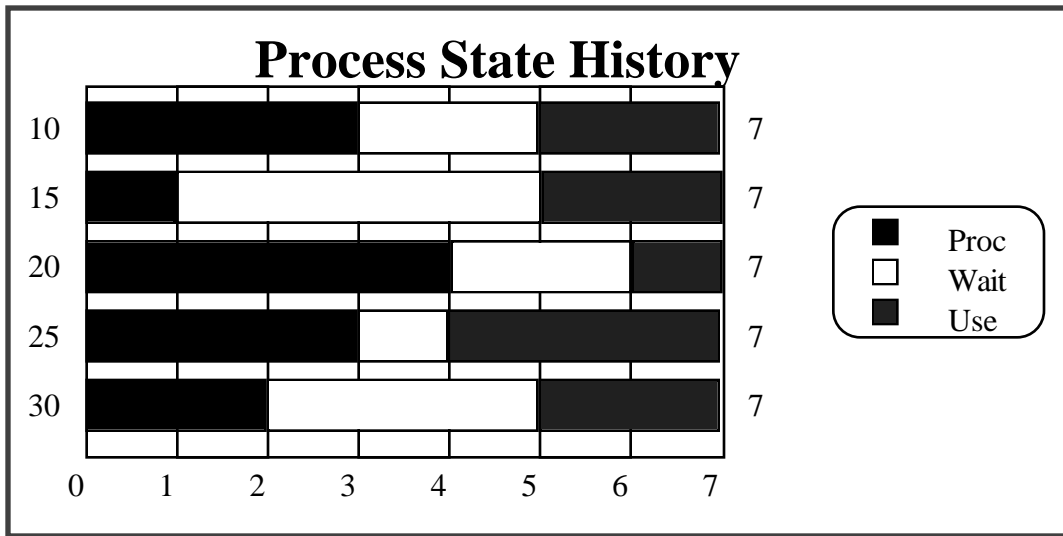


Fig. A3-1: Resource Use Model - History Chart (hc_resmod)

Table of Version 1.9 History Chart Functions

Creating	HC'decint HC'create
Updating:	
Assigning labels	HC'hist_ltag HC'hist_rtag
Appending values	HC'hist_init HC'hist_row
Deleting	HC'delete

HC'create

Creates a history bar chart.

Synopsis

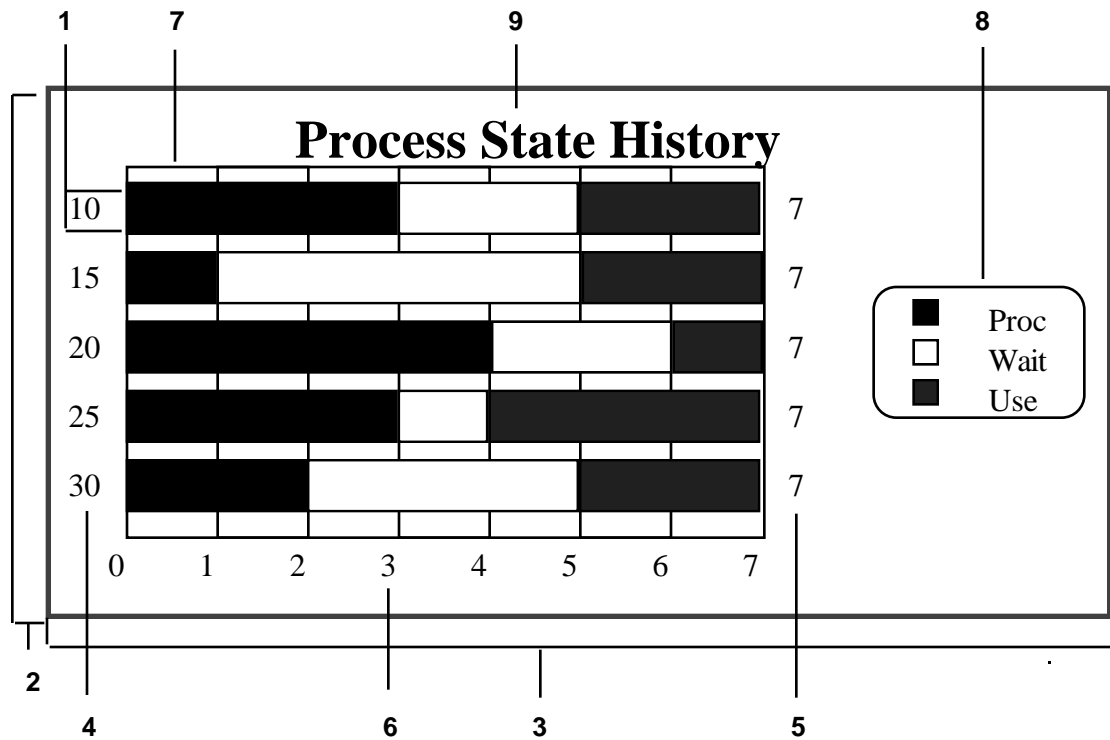
```
HC'create:
{ bar_height : int,    (* Height of the individual rows (bars) *)
  height : int,       (* Height of the chart *)
  width : int,        (* Width of the chart *)
  x : int,            (* Horizontal location on the page of the
                      upper right-hand corner *)
  y : int,            (* Vertical location on the page of the upper
                      right-hand corner *)
  columns : int,      (* Number of columns in each row *)
  row : int,          (* Number of rows in the chart *)
  tag : bool,         (* Specifies whether or not the chart has row
                      labels *)
  val_reg : bool      (* Specifies whether or not the chart has
                      value labels *)
  max_val : int,      (* Determines the range of values that are
                      displayed in the plot area *)
  resize : bool,     (* Specifies whether or not the maximum value
                      shown on the horizontal grid is adjusted to
                      be larger than the maximum bar length *)
  grid_dyn : bool,   (* Specifies whether or not the number of grid
                      lines are to be adjusted dynamically *)
  grid_no : int,     (* Number of grid lines in the chart *)
  grid_vis : bool,   (* Specifies whether or not the grid lines are
                      visible *)
  low_grid : bool,   (* Specifies whether or not the chart has grid
                      labels on the bottom of the chart *)
  up_grid : bool,    (* Specifies whether or not the chart has grid
                      labels at the top of the chart *)
  legend : bool,     (* Specifies whether or not the chart has a
                      legend identifying column patterns *)
  title : string,    (* Text of chart title *)
}
-> int HCHART
```

Description

Creates an integer history bar chart.

Arguments

The Figure Location column in the table below refers to this diagram:



Argument	Type	Example	Figure Location	Comments
bar_height	int	20	1	Height of the individual rows (bars). The value is in pixels.
height	int	200	2	Height of the chart. The value is in pixels.
width	int	500	3	Width of the chart. The value is in pixels.
x	int	0	N/A	Horizontal location on the page of the upper right-hand corner. See Chapter A1 for a simple chart positioning technique that uses x and y.
y	int	700	N/A	Vertical location on the page of the upper right-hand corner. See Chapter A1 for a simple chart positioning technique that uses x and y.
columns	int	2	N/A	Number of columns in each row. The Resource model in this appendix uses three columns in its history chart.

Version 1.9 Chart Functions

row	int	4	N/A	Number of rows in the chart. HC'hist_row generates a new row which is located underneath the previous row. New rows are added up to the maximum specified by this argument. Subsequent rows are added at the bottom and the oldest row is scrolled off the top of the chart.
tag	bool	true	4	Specifies whether or not the chart has row labels. HC'hist_rtag determines the labels for the rows.
val_reg	bool	true	5	Specifies whether or not the chart has value labels. The system automatically creates the labels from the position of the left-hand end of the bar.
max_val	int	30	N/A	Determines the range of values that are displayed in the plot area. Zero is the minimum, max_val is the maximum.
resize	bool	true	N/A	Specifies whether or not the maximum value shown on the horizontal grid is adjusted to be larger than the maximum bar length. Setting resize to true helps to identify errors caused by giving a number outside of the max_val range.
grid_dyn	bool	true	N/A	Specifies whether or not the number of grid lines are to be adjusted dynamically.
grid_no	int	30	N/A	Number of grid lines in the chart. If grid_dyn is false, this number is fixed. If grid_dyn is true, this number is the maximum.
grid_vis	bool	true	N/A	Specifies whether or not the grid lines are visible.
low_grid	bool	true	6	Specifies whether or not the chart has grid labels on the bottom of the chart.
up_grid	bool	false	7	Specifies whether or not the chart has grid labels at the top of the chart. If the chart has a title, up_grid is usually false to avoid text conflict.
legend	bool	true	8	Specifies whether or not the chart has a legend identifying column patterns. HC'hist_ltag determines the labels for the patterns.
title	string	See figure	9	Text of the chart title.

Return Value

None.

Exceptions

None.

Example

To create the integer history bar chart `hc_resmod` shown in Figure A3-1 type:

```
hc_resmod := HC'create
  {bar_height = 20, height = 200, width = 400, x =
  0, y = 350, columns = 3, row = 5, tag = true,
  val_reg = true, max_val = ProcCount, resize = true,
  grid_dyn = false, grid_no = ProcCount, grid_vis =
  true, low_grid = true, up_grid = false, legend =
  true, title = "Process State History"};
```

See Also

```
HC'decint
HC'delete
HC'hist_ltag
HC'hist_rtag
```

Corresponding real function

```
HC'create: { bar_height : int, height : int, width :
int, x : int, y : int, columns : int, tag : bool,
grid_dyn : bool, grid_no : int, grid_vis : bool,
legend : bool, low_grid : bool, max_val : real,
resize : bool, row : int, title : string, up_grid :
bool, val_reg : bool } -> real HCHART
```

HC'decint

Declares an integer history bar chart.

Synopsis

```
HC'decint: () -> (int HCHART) ref
```

Description

Declares an integer history bar chart. This function has the effect of creating an ML value that can be used together with HC'create to create an integer history chart.

Arguments

None.

Return Value

Returns reference to integer history bar chart.

Exceptions

None.

Example

To declare the integer history bar chart `hc_resmod` shown in Figure A3-1, type:

```
val hc_resmod = HC'decint ();
```

See Also

HC'create

Corresponding real function

```
HC'dcreal = fn : () -> (real HCHART) ref
```

HC'delete

Deletes an integer history bar chart.

Synopsis

```
HC'delete : ((int HCHART) ref) -> unit
```

Description

Deletes an integer history bar chart.

Arguments

Reference to an integer history chart.

Return Value

None.

Exceptions

None.

Example

To delete the integer history bar chart `hc_resmod` shown in Figure A3-1, type:

```
HC'delete hc_resmod;
```

See Also

`HC'create`

Corresponding real function

```
HC'delete : ((real HCHART) ref) -> unit
```

HC'hist_ltag

Updates column legend labels on an integer history bar chart.

Synopsis

```
HC'hist_ltag : { hc : (int HCHART) ref, tags : string  
list } -> unit
```

Description

Updates column labels on an integer history bar chart.

Arguments

hc	Reference to an integer history chart
tags	List of strings to use as column labels

Return Value

None.

Exceptions

None.

Example

To create legend labels for the integer history bar chart `hc_resmod` shown in Figure A3-1, type:

```
HC'hist_ltag {hc = hc_resmod , tags = ["Proc",  
"Wait", "Use"]};
```

See Also

```
HC'create  
HC'hist_rtag
```

Corresponding real function

```
HC'hist_ltag : { hc : (real HCHART) ref, tags : string  
list } -> unit
```

HC'hist_init

Initializes columns in an integer history bar chart.

Synopsis

```
HC'hist_col : { hc : (int HCHART) ref, init : int } ->  
unit
```

Description

Initializes columns in an integer history bar chart. This provides a method for clearing the value of individual columns.

Arguments

hc	Reference to an integer history chart
init	Integer column number to be updated

Return Value

None.

Exceptions

None.

Example

The Resource Use model used in this appendix does not initialize individual columns, but the function to clear the first column in the current row would be:

```
HC'hist_init {hc = hc_resmod, 1}
```

See Also

```
HC'create  
HC'delete  
HC'hist_row
```

Version 1.9 Chart Functions

Corresponding real function

```
HC'hist_init : { hc : (real HCHART) ref, init : int }  
-> unit
```

HC'hist_row

Creates a new row in an integer history bar chart.

Synopsis

```
HC'hist_row : { hc : (int HCHART) ref, new_row : int  
list } -> unit
```

Description

Creates a new row in an integer history bar chart. New rows are added up to the maximum specified by the HC'create function row argument. Subsequent rows are added at the bottom and the oldest row is scrolled off the top of the chart.

Arguments

hc	Reference to an integer history chart
new_row	Values for the columns in the new row

Return Value

None.

Exceptions

None.

Example

To create a new row for the integer history bar chart `hc_resmod` shown in Figure A3-1, type:

```
HC'hist_row {hc = hc_resmod, new_row = [SV'sum  
isv_ProcLoc, SV'sum isv_AwaitRes, SV'sum  
isv_UseRes]};
```

See Also

HC'create
HC'hist_rtag

Version 1.9 Chart Functions

Corresponding real function

```
HC'hist_row : { hc : (real HCHART) ref, new_row :  
real list } -> unit
```

HC'hist_rtag

Updates row labels on an integer history bar chart.

Synopsis

```
HC'hist_rtag : { hc : (int HCHART) ref, tags : string  
list } -> unit
```

Description

Updates row labels on an integer history bar chart.

Arguments

hc	Reference to an integer history chart
tags	List of strings to use as row labels

Return Value

None.

Exceptions

None.

Example

To create time-based row labels for the integer history bar chart `hc_resmod` shown in Figure A3-1, type:

```
HC'hist_rtag {hc = hc_resmod , tag = makestring  
(time ())};
```

See Also

HC'hist_row

Corresponding real function

```
HC'hist_rtag : { hc : (real HCHART) ref, tags :  
string list } -> unit
```


Chapter A4

Version 1.9

Line Graph Functions

Version 1.9 Line Graph Example

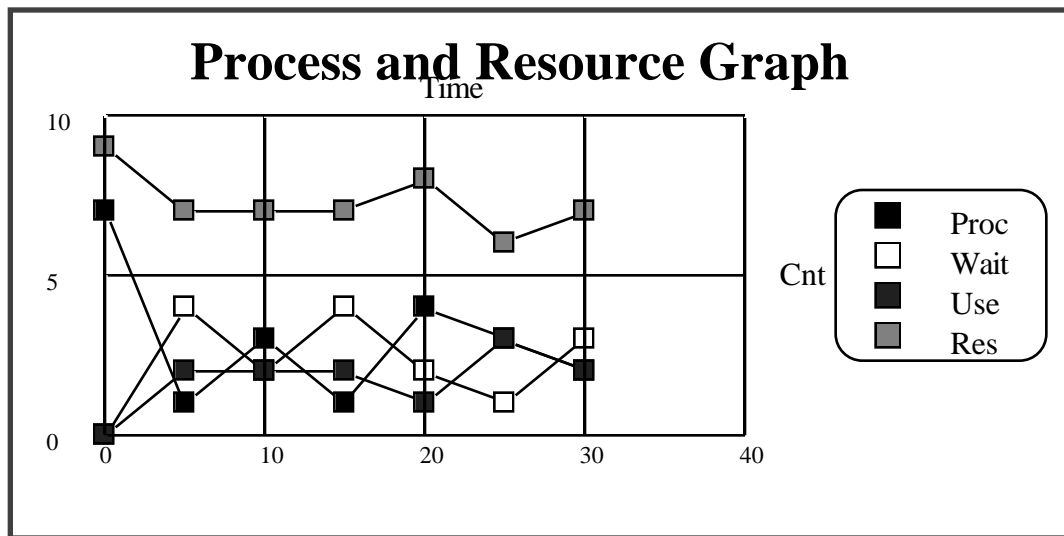


Fig. A4-1: Resource Use Model - Line Graph (lg_resmod)

Table of Version 1.9 Line Graph Functions

Creating	LG'decint LG'create
Updating:	
Assigning labels	LG'upd_ltag LG'upd_atag
Appending values	LG'upd_line LG'upd_pline
Deleting	LG'delete

LG'create

Creates a line graph.

Synopsis

```
LG'create:
  { height : int,      (* Height of the line graph *)
    width : int,      (* Width of the line graph *)
    x : int,          (* Horizontal location on the page of the
                      upper right-hand corner *)
    y : int           (* Vertical location on the page of the upper
                      right-hand corner *)

    xmax : int,      (* Maximum value of x coordinate *)
    xmin : int,      (* Minimum value of x coordinate *)
    xcent : int,     (* x value of center *)
    XMaxTick : int,  (* Number of ticks on the x coord right of
                      center *)
    XMinTick : int,  (* Number of ticks on the x coord left of
                      center *)

    ymax : int,      (* Maximum value of y coordinate *)
    ymin : int,      (* Minimum value of y coordinate *)
    ycent : int,     (* y value of center *)
    YMaxTick : int,  (* Number of ticks of the y coord right of
                      center *)
    YMinTick : int,  (* Number of ticks on the y coord left of
                      center *)

    bigticks : bool, (* Specifies whether or not ticks are as big
                      as the line graph *)
    multiple : bool, (* Specifies whether or not the graph is a
                      multiple y coordinate system *)
    nline : int,     (* Number of graph lines *)
    thick : int,     (* Thickness of graph lines *)
    boxsize : int,   (* Size of nodes connecting the line graph *)
    timing : bool,   (* Specifies whether or not the graph is a
                      time-series graph *)

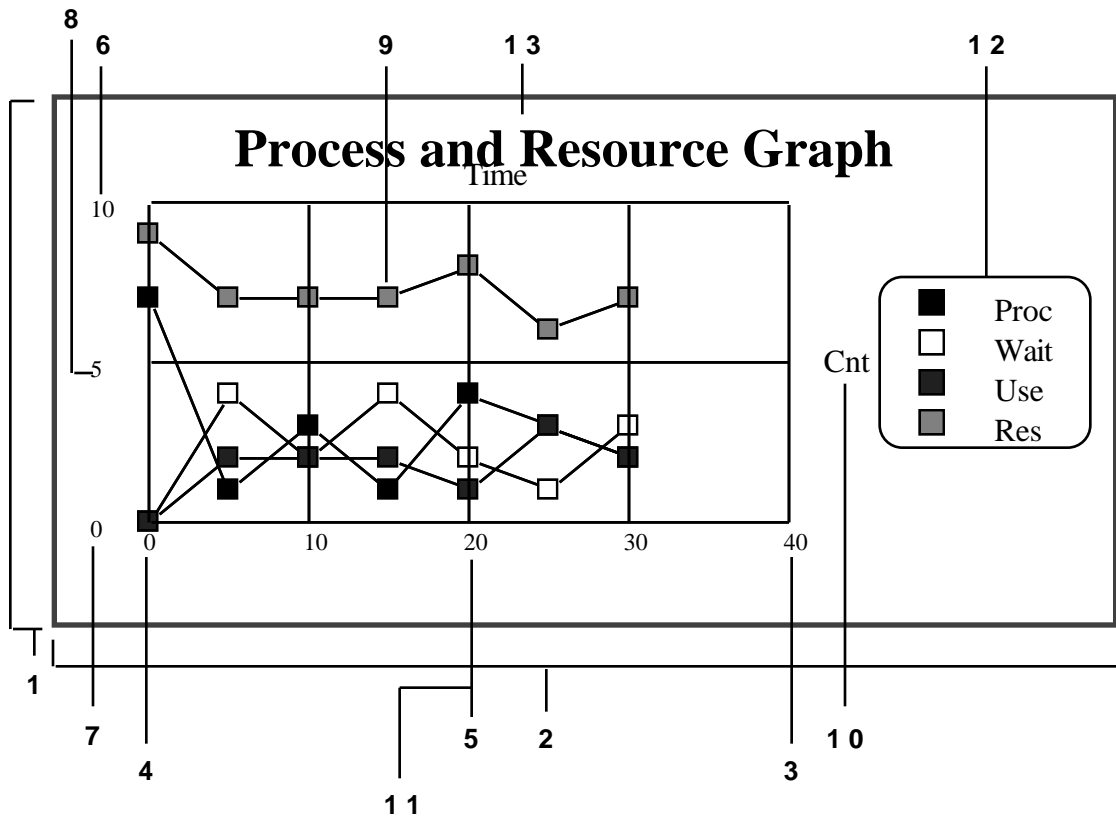
    axis : bool,     (* Specifies whether or not the graph has axis
                      labels *)
    center : bool,   (* Specifies whether or not the graph has
                      center labels *)
    legend : bool,   (* Specifies whether or not the graph has a
                      legend identifying graph line patterns *)
    title : string,  (* Text of line graph title*)
  }
-> int LINEGRAPH
```

Description

Creates an integer line graph.

Arguments

The Figure Location column in the table below refers to this diagram:



Argument **Type** **Example** **Figure Location** **Comments**

height	int	200	1	Height of the line graph. The value is in pixels.
width	int	500	2	Width of the line graph. The value is in pixels.
x	int	0	N/A	Horizontal location on the page of the upper right-hand corner. See Chapter A1 for a simple chart positioning technique that uses x and y.
y	int	700	N/A	Vertical location on the page of the upper right-hand corner. See Chapter A1 for a simple chart positioning technique that uses x and y.
xmax	int		3	Maximum value of x coordinate.
xmin	int		4	Minimum value of x coordinate.
xcent	int		5	x value of center
XMaxTick	int			Number of ticks on the x coordinate right of center
XMinTick	int			Number of ticks on the x coordinate left of center

Version 1.9 Chart Functions

y _{max}	int		6	Maximum value of y coordinate
y _{min}	int		7	Minimum value of y coordinate
y _{cent}	int		8	y value of center
Y _{MaxTick}	int			Number of ticks on the y coordinate above the center
Y _{MinTick}	int			Number of ticks on the y coordinate below the center
bigticks	bool			Specifies whether or not ticks are as big as the line graph
multiple	bool			Specifies whether or not the graph is a multiple y coordinate system
n _{line}	int			Number of graph lines
thick	int			Thickness of graph lines
boxsize	int		9	Size of nodes connecting the line graph. The boxes contain the patterns identifying the lines.
timing	bool			Specifies whether or not the graph is a time-series graph
axis	bool		10	Specifies whether or not the graph has axis labels
center	bool		11	Specifies whether or not the graph has center labels.
legend	bool		12	Specifies whether or not the graph has a legend describing graph line patterns. LG'upd_ltag determines the labels for the patterns.
title	string	See figure	13	Text of the graph title.

Return Value

None.

Exceptions

None.

Example

To create the integer line graph `lg_resmod` shown in Figure A4-1, type:

```
lg_resmod := LG'create
{height=200, width=400, x=0, y=350, xmax=40,
xmin=0, xcent=20, XMaxTick=2, XMinTick=2, ymax=10,
ymin=0, ycent=5, YMaxTick=1, YMinTick=1,
bigticks=true, multiple=false, nline=4, thick=2,
boxsize=10, timing=false, axis=true, center=true,
legend=true, title="Process and Resource Graph"};
```

See Also

```
LG'decint
LG'delete
LG'upd_atag
LG'upd_ltag
```

Corresponding real function

```
LG'create: {legend : bool, XMaxTick : int, XMinTick :
int, XMaxTick : int, XMinTick : int, axis : bool,
bigticks : bool, boxsize : int, center : bool, height
: int, multiple : bool, nline : int, thick : int,
timing : bool, title : string, width : int, x : int,
xcent : real, xmax : real, xmin : real, y : int,
ycent : real, ymax : real, ymin : real } -> int
LINEGRAPH
```

LG'decint

Declares an integer line graph.

Synopsis

```
LG'decint: () -> (int LINEGRAPH) ref
```

Description

Declares an integer line graph. This function has the effect of creating an ML value that can be used together with LG'create to create an integer line graph.

Arguments

None.

Return Value

Reference to an integer line graph

Exceptions

None.

Example

To declare the integer line graph lg_resmod shown in Figure A4-1, type:

```
val lg_resmod = LG'decint ();
```

See Also

LG'create

Corresponding real function

```
LG'dcreal: () -> (real LINEGRAPH) ref
```

LG'delete

Deletes an integer line graph.

Synopsis

```
LG'delete : ((int LINEGRAPH) ref) -> unit
```

Description

Deletes an integer line graph.

Arguments

Reference to an integer line graph

Return Value

None.

Exceptions

None.

Example

To delete the integer line graph `lg_resmod` shown in Figure A4-1, type:

```
LG'delete lg_resmod;
```

See Also

`LG'create`

Corresponding real function

```
LG'delete : ((real LINEGRAPH) ref) -> unit
```

LG'upd_atag

Updates axis labels for an integer line graph.

Synopsis

```
LG'upd_atag: { lg : (int LINEGRAPH) ref, x : string, y  
: string } -> unit
```

Description

Updates axis labels for an integer line graph.

Arguments

lg Reference to an integer line graph
x String for x-coordinate tag
y String for y-coordinate tag

Return Value

None.

Exceptions

None.

Example

To create axis labels for the integer line graph `lg_resmod` shown in Figure A4-1, type:

```
LG'upd_atag {lg = lg_resmod, x="Cnt", y="Time" };
```

See Also

LG'create
LG'upd_ltag

Corresponding real function

```
LG'upd_atag: { lg : (real LINEGRAPH) ref, x : string,  
y : string } -> unit
```

LG'upd_ltag

Updates legend labels on an integer line graph.

Synopsis

```
LG'upd_ltag : { lg : (int LINEGRAPH) ref, tags :  
string list } -> unit
```

Description

Updates labels on an integer line graph.

Arguments

lg Reference to an integer line graph
tags List of strings for lines on line graph

Return Value

None.

Exceptions

None.

Example

To create legend labels for the integer line graph `lg_resmod` shown in Figure A4-1, type:

```
LG'upd_ltag {lg = lg_resmod, tags =  
["Proc","Wait","Use", "Res"]};
```

See Also

LG'create
LG'upd_atag

Corresponding real function

```
LG'upd_ltag : { lg : (real LINEGRAPH) ref, tags :  
string list } -> unit
```

LG'upd_line

Updates an integer line graph.

Synopsis

```
LG'upd_line : { lg : (int LINEGRAPH) ref, line : int,  
x : int, y : int} -> unit
```

Description

Updates an integer line graph.

Arguments

lg	Reference to an integer line graph
line	Number of the line to be updated
x	x-coordinate of the line to be updated
y	y-coordinate of the line to be updated

Return Value

None.

Exceptions

None.

Example

To update the integer line graph `lg_resmod` shown in Figure A4-1, type:

```
LG'upd_line {lg = lg_resmod, line = 1,  
x = time (), y = SV'sum isv_ProcLoc};  
LG'upd_line {lg = lg_resmod, line = 2,  
x = time (), y = SV'sum isv_AwaitRes};  
LG'upd_line {lg = lg_resmod, line = 3,  
x = time (), y = SV'sum isv_UseRes};  
LG'upd_line {lg = lg_resmod, line = 4,  
x = time (), y = SV'sum isv_ResPool};
```

See Also

LG'create
LG'upd_pline

Corresponding real function

LG'upd_line : { lg : (**real** LINEGRAPH) ref, line : int, x : **real**, y : **real** } -> unit

LG'upd_pline

Updates the pattern of an integer line graph.

Synopsis

```
LG'upd_pline : { lg : (int LINEGRAPH) ref, line : int,  
pattern : int, x : int, y : int} -> unit
```

Description

Updates the pattern of an integer line graph.

Arguments

lg	Reference to an integer line graph
line	Number of the line to be updated
pattern	Pattern number for the line to be updated
x	x-coordinate of the line to be updated
y	y-coordinate of the line to be updated

Return Value

None.

Exceptions

None.

Example

The Resource model used in this appendix does not use LG'upd_pline.

See Also

```
LG'create  
LG'upd_line
```

Corresponding real function

```
LG'upd_pline : { lg : (real LINEGRAPH) ref, line :  
int, pattern : int, x : real, y : real} -> unit
```

Chapter A5

Version 1.9

Matrix Chart Functions

Version 1.9 Matrix Chart Example

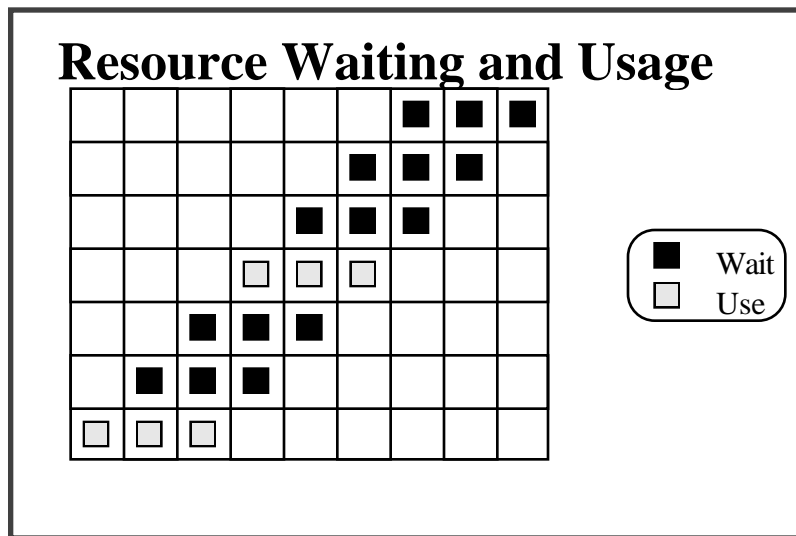


Fig. A5-1: Resource Use Model - Matrix Chart (mc_resmod)

Table of Version 1.9 Matrix Chart Functions

Creating	MC'dec MC'create
Updating:	
Assigning labels	MC'upd_ltag
Appending values	MC'fill MC'write
Deleting	MC'delete

MC'create

Creates a matrix chart.

Synopsis

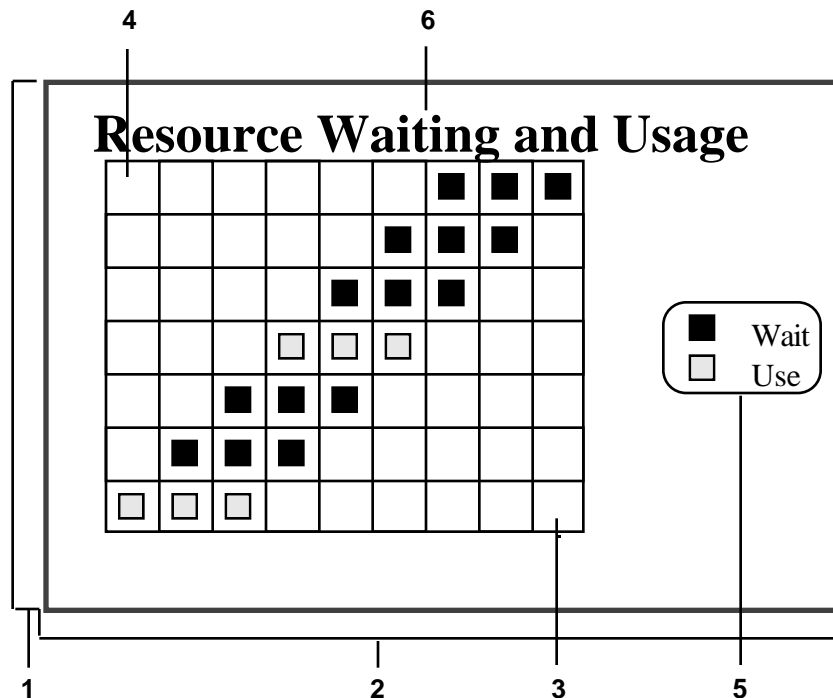
```
MC'create:
  { height : int,      (* Height of the chart *)
    width : int,      (* Width of the chart *)
    x : int,          (* Horizontal location on the page of the
                      upper right-hand corner *)
    y : int,          (* Vertical location on the page of the upper
                      right-hand corner. *)
    i : int,          (* Maximum number of x-coordinate cells *)
    j : int,          (* Maximum number of y-coordinate cells *)
    pattern : int,    (* Number of cell patterns *)
    legend : bool,    (* Specifies whether or not the chart has a
                      legend identifying line patterns *)
    title : string,   (* Text of the chart title *)
  }
-> MCHART
```

Description

Creates an integer matrix chart.

Arguments

The Figure Location column in the table below refers to this diagram:



Argument **Type** **Example** **Figure Location** **Comments**

height	int	200	1	Height of the chart. The value is in pixels.
width	int	500	2	Width of the chart. The value is in pixels.
x	int	0	N/A	Horizontal location on the page of the upper right-hand corner. See Chapter A1 for a simple chart positioning technique that uses <i>x</i> and <i>y</i> .
y	int	700	N/A	Vertical location on the page of the upper right-hand corner. See Chapter A1 for a simple chart positioning technique that uses <i>x</i> and <i>y</i> .
i	int		3	Maximum number of x-coordinate cells
j	int		4	Maximum number of y-coordinate cells
pattern	int		5	Number of cell patterns
legend	bool		5	Specifies whether or not the chart has a legend identifying line patterns. <code>MC'upd_ltag</code> determines the labels for the patterns.
title	string	See figure	6	Text of the chart title.

Version 1.9 Chart Functions

Return Value

Reference to a matrix chart

Exceptions

None.

Example

To create the integer matrix chart `mc_resmod` shown in Figure A5-1, type:

```
mc_resmod:= MC'create
{height = 200, width = 300, i = ProcCount,
 j = ResCount, legend = true, patterns = 2, x = 0,
 y = 350, title = "Resource Waiting and Usage" };
```

See Also

```
MC'dec
MC'delete
MC'upd_ltag
```

Corresponding real function

N/A

MC'dec

Declares an integer matrix chart.

Synopsis

```
MC'dec: () -> (int MCHART) ref
```

Description

Declares a matrix chart. This function has the effect of creating an ML value that can be used together with `MC'create` to create an integer matrix chart.

Arguments

None.

Return Value

Reference to a matrix chart.

Exceptions

None.

Example

To declare the integer matrix chart `mc_resmod` shown in Figure A5-1, type:

```
val mc_resmod = MC'dec ();
```

See Also

`MC'create`

Corresponding real function

N/A

MC'delete

Deletes an integer matrix chart.

Synopsis

```
MC'delete : ((int MCHART) ref) -> unit
```

Description

Deletes an integer matrix chart.

Arguments

Reference to a matrix chart

Return Value

None.

Exceptions

None.

Example

To delete the integer matrix chart `mc_resmod` shown in Figure A5-1, type:

```
MC'delete mc_resmod;
```

See Also

`MC'create`

Corresponding real function

N/A

MC'fill

Updates the fill pattern in an integer matrix chart.

Synopsis

```
MC'fill : { mc : (int MCHART) ref, pattern : int, i :  
int, j : int } -> unit
```

Description

Updates the fill pattern in an integer matrix chart.

Arguments

mc	Reference to a matrix chart
pattern	Fill pattern number
i	X coordinate of cell to be updated
j	Y coordinate of cell to be updated

Return Value

None.

Exceptions

None.

Example

To update the fill pattern for the integer matrix chart `mc_resmod` shown in Figure A5-1, type:

```
MC'fill {mc = mc_resmod, pattern = 1,  
i = (index'Res res1), j = (index'Proc pid)};  
MC'fill {mc = mc_resmod, pattern = 1,  
i = (index'Res res2), j = (index'Proc pid)};  
MC'fill {mc = mc_resmod, pattern = 1,  
i = (index'Res res3), j = (index'Proc pid)};
```

See Also

MC'create
MC'write

Version 1.9 Chart Functions

Corresponding real function

N/A

MC'upd_ltag

Updates pattern legend labels on an integer matrix chart.

Synopsis

```
MC'upd_ltag : { mc : (int MCHART) ref, tags : string  
list } -> unit
```

Description

Updates pattern legend labels on an integer matrix chart.

Arguments

mc	Reference to a matrix chart
tags	List of strings giving labels for the fill patterns

Return Value

None.

Exceptions

None.

Example

The Resource Use model used in this appendix does not use MC'upd_ltag.

See Also

MC'create

Corresponding real function

N/A

MC'write

Updates cell values in an integer matrix chart.

Synopsis

```
MC'write : { mc : (int MCHART) ref, i : int, j : int,  
text : string } -> unit
```

Description

Updates cell values in an integer matrix chart.

Arguments

mc	Reference to a matrix chart
i	X coordinate of cell to be updated
j	Y coordinate of cell to be updated
text	Text for updating cell

Return Value

None.

Exceptions

None.

Example

The Resource Use model used in this appendix does not use MC'write.

See Also

```
MC'create  
MC'fill
```

Corresponding real function

N/A

Version 1.9 Matrix Chart Functions
